[MS-IPFF]:

InfoPath Form Template Format

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.
- **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.

Revision Summary

Date	Revision History	Revision Class	Comments
4/4/2008	0.01	Major	Initial Availability
6/27/2008	1.0	Minor	Revised and edited technical content
7/13/2009	1.01	Major	Revised and edited the technical content
8/28/2009	1.02	Editorial	Revised and edited the technical content
11/6/2009	1.03	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	None	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	2.05	None	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	2.05	None	No changes to the meaning, language, or formatting of the technical content.
3/18/2011	2.05	None	No changes to the meaning, language, or formatting of the technical content.
6/10/2011	2.05	None	No changes to the meaning, language, or formatting of the technical content.
1/20/2012	2.6	Minor	Clarified the meaning of the technical content.
4/11/2012	2.6	None	No changes to the meaning, language, or formatting of the technical content.
7/16/2012	2.6	None	No changes to the meaning, language, or formatting of the technical content.
9/12/2012	2.6	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	2.6	None	No changes to the meaning, language, or formatting of the technical content.
2/11/2013	2.6	None	No changes to the meaning, language, or formatting of the technical content.
7/30/2013	2.6	None	No changes to the meaning, language, or formatting of the technical content.
11/18/2013	2.6	None	No changes to the meaning, language, or formatting of the technical content.

Date	Revision History	Revision Class	Comments
2/10/2014	2.6	None	No changes to the meaning, language, or formatting of the technical content.
4/30/2014	2.6	None	No changes to the meaning, language, or formatting of the technical content.
7/31/2014	2.6	None	No changes to the meaning, language, or formatting of the technical content.
10/30/2014	2.7	Minor	Clarified the meaning of the technical content.
6/23/2016	2.7	None	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

This document specifies the file format for InfoPath form templates. A form template (.xsn) file contains several files that are used to represent the data fields, visualization and behavior of a specific type of electronic form. For example, an expense report form template could define the data that expense report forms need to contain, such as the total amount and date filed, which data is optional, and how the data fields are be presented to the person filling out the report.

A form server understanding the format can parse and retrieve the files inside the form template (.xsn) file. The form server can then use the information to render and edit a new or existing form file in a Web browser.

In other words, a form server needs to understand the two essential components of a form:

- The form template (.xsn) file used to render and edit the data.
- The form file used to store the data.

Details on how form files are associated with a form template are described in [MS-IPFFX].

Sections 1.7 and 2 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

ActiveX Data Objects (ADO): A data access interface that connects to, retrieves, manipulates, and updates data in Object Linking and Embedding (OLE) database-compliant data sources.

ASP.NET control: A server-side component that encapsulates user interface and related functionality. An ASP.NET server control derives directly or indirectly from the System.Web.UI.Control class. The superset of ASP.NET server controls includes web server controls, HTML server controls, and ASP.NET mobile controls.

assembly: A collection of one or more files that is versioned and deployed as a unit. An assembly is the primary building block of a .NET Framework application. All managed types and resources are contained within an assembly and are marked either as accessible only within the assembly or as accessible from code in other assemblies. Assemblies also play a key role in security. The code access security system uses information about an assembly to determine the set of permissions that is granted to code in the assembly.

attachment: An external file that is included with an Internet message or associated with an item in a SharePoint list.

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].

background color: A color against which characters, patterns, and graphics are displayed. See also foreground color.

blind carbon copy (bcc) recipient: An addressee whose name is not visible to other addressees of an Internet message.

Boolean: An operation or expression that can be evaluated only as either true or false.

border formatting: A set of properties that, as a whole, specify the appearance of a border, such as color, line style, and thickness.

browser-compatible form template: A form template that is designed for publication to a protocol server that is running InfoPath Forms Services.

browser-enabled form template: A form template that is published to a protocol server that is running InfoPath Forms Services and is also activated for use on that server.

built-in control: A control that is included with Microsoft InfoPath and appears by default in the Controls pane.

business object: An object that performs a defined set of operations, such as data validation or business logic (2) rules, related to a business process or workflow.

cabinet (.cab) file: A single file that stores multiple compressed files to facilitate storage or transmission.

carbon copy (cc) recipient: In an Internet message, an addressee whose name is visible to other addressees and is not necessarily expected to take any action. The message is for informational purposes only for that addressee.

class identifier (CLSID): A **GUID** that identifies a software component; for instance, a DCOM object class (4) or a COM class.

column: A single set of data that is displayed vertically in a worksheet or a table.

connection string: A series of arguments, delimited by a semicolon, that defines the location of a database and how to connect to it.

control: A graphical user interface object that users interact with when working with applications, forms, documents, webpages, and other types of files.

custom control: A component of an InfoPath form, such as a template part or ActiveX control, that is not included with Microsoft InfoPath by default.

data adapter: Code that submits data to and retrieves data from an external data source. Also referred to as data provider.

data connection: A link between an application and a data source. Data connections can be used to query and submit data.

data connection library: A SharePoint library that contains a collection of universal data connection (.udcx) and Office data connection (.odc) files.

data source: A collection of fields and groups that define and store the data for an InfoPath form. Controls in a form are bound to the fields and groups in the data sources of the form. See also **main data source** and **secondary data source**.

digital signature: A message authenticator that is typically derived from a cryptographic operation by using an asymmetric algorithm and private key. When a symmetric algorithm is used for this purpose, the authenticator is typically referred to as a Message Authentication Code (MAC).

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

Extended Backus-Naur Form (EBNF): A modified version of Backus-Naur Form (BNF), which is commonly used to describe programming languages and formal languages. EBNF extends standard BNF to better enable the concise expression of such languages, as described in [ISO-14977].

Extensible Stylesheet Language (XSL): An **XML** vocabulary that is used to transform XML data to another form, such as **HTML**, by means of a style sheet that defines presentation rules.

file: A single, discrete unit of content.

form: A structured document with controls and spaces that are reserved for entering and displaying information. Forms can contain special coding for actions such as submitting and querying data.

form definition (.xsf) file: An **XML** file with an .xsf file name extension. The file contains information about the files and components that are used within a form, including user interface customizations, **XML schemas**, views, business logic (1), events (2), and deployment settings.

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

form library: A type of document library that is optimized for storing and displaying data in XML-based forms.

form security level: A setting that determines whether an InfoPath form can access data on other domains, or access files and settings on a user's computer. There are three security levels for forms: Restricted, Domain, and Full Trust.

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.

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.

friendly name: A name for a user or object that can be read and understood easily by a human.

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

group: An element that can contain fields and other groups in the data source for an InfoPath form. Controls that contain other controls, such as repeating tables and sections, are bound to groups.

header field: A component of a Session Initiation Protocol (SIP) message header, as described in [RFC3261].

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 Markup Language (HTML): 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 (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.

Information Rights Management (IRM): A technology that provides persistent protection to digital data by using encryption, certificates (1), and authentication (2). Authorized recipients or users acquire a license to gain access to the protected files according to the rights or business rules that are set by the content owner.

language code identifier (LCID): A 32-bit number that identifies the user interface human language dialect or variation that is supported by an application or a client computer.

left-to-right: A reading order in which characters in words are read from left to right, and words are read from left to right in sentences.

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

list view: A named collection of settings for querying and displaying items in a SharePoint list. There are two types of views: Personal, which can be used only by the user who created the view; and Public, which can be used by all users who have permission to access to the site.

locale: A collection of rules and data that are specific to a language and a geographical area. A locale can include information about sorting rules, date and time formatting, numeric and monetary conventions, and character classification.

lookup field: A field of the Lookup type that enables users to select an item from another data source.

main data connection: The primary connection between an InfoPath form and a **data source** that stores or provides data for the form. The main data connection defines the structure of the main data source of the form.

main data source: An **XML document** or **XML schema** that defines the collection of fields (1) and **groups** that store data for an InfoPath form.

MIME type: A method that is used by protocol clients to associate files of a certain type with applications that can open or access files of that type.

mobile device: A small computing device that is easily portable and can be used in various environments.

named property: A property that is identified by both a GUID and either a string name or a 32-bit identifier.

offline: The condition of not being connected to or not being on a network or the Internet. Offline can also refer to a device, such as a printer that is not connected to a computer, and files that are stored on a computer that is not connected to or not on a network or the Internet.

postback: A process in which a webpage sends data back to the server that hosts the page.

print view: A document view that displays a document as it will appear on a printed page.

query: A formalized instruction to a data source to either extract data or perform a specified action. A query can be in the form of a query expression, a method-based query, or a combination of the two. The data source can be in different forms, such as a relational database, **XML document**, or in-memory object. See also search query.

red-green-blue (RGB): A color model that describes color information in terms of the red (R), green (G), and blue (B) intensities in a color.

repeating group: A group that can occur more than once in the data source for an InfoPath form. Controls such as repeating sections and repeating tables can be bound to repeating groups. See also bind.

rich text: Text that is formatted in the Rich Text Format, as described in [MSFT-RTF].

right-to-left: A reading and display order that is optimized for right-to-left languages.

root element: The top-level element in an **XML document**. It contains all other elements and is not contained by any other element, as described in [XML].

rule: A condition or action, or a set of conditions or actions, that performs tasks automatically based on events and values.

secondary data connection: Any auxiliary connection between an InfoPath form and a data source that stores or provides data for the form.

secondary data source: An **XML** data file, a database, or a **web service** that is used to populate controls or provide values in an InfoPath form.

site: (1) A group of related webpages that is hosted by a server on the World Wide Web or an intranet. Each website has its own entry points, metadata, administration settings, and workflows. Also referred to as web site.

(2) 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.

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

SOAP message: An **XML** document consisting of a mandatory SOAP envelope, an optional SOAP header, and a mandatory SOAP body. See [SOAP1.2-1/2007] section 5 for more information.

SQL statement: A complete phrase in SQL that begins with a keyword and completely describes an action to be taken on data.

Structured Query Language (SQL): A database query and programming language that is widely used for accessing, querying, updating, and managing data in relational database systems.

submit: The process of sending data to an external data source such as a web service, database, Internet message, or SharePoint site.

symbol file: A file that contains information about an executable image, including the names and addresses of functions and variables.

toolbar: A row, column, or block of controls that represent tasks or commands within an application. A toolbar can be either a menu toolbar, which provides access to menu commands, or a basic toolbar, which contains buttons that provide shortcuts to tasks that are frequently accessed from menus.

Unicode: A character encoding standard developed by the Unicode Consortium that represents almost all of the written languages of the world. The **Unicode** standard [UNICODE5.0.0/2007] 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 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].

Universal Data Connection (.udc, .udcx) file: An **XML** file that has a .udc or .udcx file name extension that contains user credentials and other authentication information that is used to connect to a data source.

Universal Naming Convention (UNC): A string format that specifies the location of a resource. For more information, see [MS-DTYP] section 2.2.57.

user agent: An HTTP user agent, as specified in [RFC2616].

user name: A unique name that identifies a specific user account. The user name of an account is unique among the other group names and user names within its own domain or workgroup.

UTF-16: A standard for encoding Unicode characters, defined in the Unicode standard, in which the most commonly used characters are defined as double-byte characters. Unless specified otherwise, this term refers to the UTF-16 encoding form specified in [UNICODE5.0.0/2007] section 3.9.

Web Distributed Authoring and Versioning Protocol (WebDAV): The Web Distributed Authoring and Versioning Protocol, as described in [RFC2518] or [RFC4918].

web service: A unit of application logic that provides data and services to other applications and can be called by using standard Internet transport protocols such as **HTTP**, Simple Mail Transfer Protocol (SMTP), or File Transfer Protocol (FTP). Web services can perform functions that range from simple requests to complicated business processes.

Web Services Description Language (WSDL): An XML format for describing network services as a set of endpoints that operate on messages that contain either document-oriented or procedure-oriented information. The operations and messages are described abstractly and are bound to a concrete network protocol and message format in order to define an endpoint. Related concrete endpoints are combined into abstract endpoints, which describe a network service. WSDL is extensible, which allows the description of endpoints and their messages regardless of the message formats or network protocols that are used.

WebDAV server: A computer that supports **WebDAV**, as described in [RFC2518] or [RFC4918], and responds to requests from WebDAV clients.

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 element: An **XML** structure that typically consists of a start tag, an end tag, and the information between those tags. Elements can have attributes (1) and can contain other elements.

XML fragment: Lines of text that adhere to **XML** tag rules, as described in [XML], but do not have a Document Type Definition (DTD) or schema, processing instructions, or any other header information.

XML node: The smallest unit of a valid, complete structure in an **XML document**. For example, a node can represent an element, an attribute (1), or a text string.

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.

XML schema definition (XSD): The World Wide Web Consortium (W3C) standard language that is used in defining XML schemas. Schemas are useful for enforcing structure and constraining the types of data that can be used validly within other XML documents. XML schema definition refers to the fully specified and currently recommended standard for use in authoring **XML schemas**.

XML schema document: See **XML schema**.

XPath expression: An expression that searches an71 **XML document** and can extract and manipulate data in elements or attributes (1) within that document.

XSL Transformation (XSLT): A declarative, XML-based language that is used to present or transform XML data. It is designed for use as part of the **Extensible Stylesheet Language (XSL)**.

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.

[CSS-LEVEL1] Lie, H. and Bos, B., "Cascading Style Sheets: W3C Recommendation", REC CSS1-19990111, January 1999, http://www.w3.org/TR/1999/REC-CSS1-19990111

[CSS-LEVEL2] Bos, B., Celik, T., Hickson, I., and Lie, H., "Cascading Style Sheets Level 2 Revision 1 (CSS2.1) Specification: W3C Candidate Recommendation", July 2007, http://www.w3.org/TR/2007/CR-CSS21-20070719/

[HTML] World Wide Web Consortium, "HTML 4.01 Specification", December 1999, http://www.w3.org/TR/html4/

[ISO-8601] International Organization for Standardization, "Data Elements and Interchange Formats - Information Interchange - Representation of Dates and Times", ISO/IEC 8601:2004, December 2004, http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=40874&ICS1=1&ICS2=140&ICS3=30

Note There is a charge to download the specification.

[MC-MCF] Microsoft Corporation, "Microsoft Cabinet Format", http://msdn.microsoft.com/en-us/library/bb417343.aspx

[MC-NLSIP] Microsoft Corporation, "National Language Support (NLS) API Reference", http://msdn.microsoft.com/en-us/goglobal/bb896001.aspx

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

[MS-LCID] Microsoft Corporation, "Windows Language Code Identifier (LCID) Reference".

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

[MS-WSSFO] Microsoft Corporation, "Windows SharePoint Services (WSS): File Operations Database Communications Protocol".

[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

[RFC2781] Hoffman, P., and Yergeau, F., "UTF-16, an encoding of ISO 10646", RFC 2781, February 2000, http://www.rfc-editor.org/rfc/rfc2781.txt

[SOAP1.2/2] Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J., and Nielsen, H.F., "SOAP Version 1.2 Part 2: Adjuncts", W3C Recommendation, June 2003, http://www.w3.org/TR/2003/REC-soap12-part2-20030624

[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/

[W3C-XSLT] Clark, J., Ed., "XSL Transformations (XSLT) Version 1.0", W3C Recommendation, November 1999, http://www.w3.org/TR/1999/REC-xslt-19991116

[XML Namespaces] Bray, T., Hollander, D., and Layman, A., "Namespaces in XML", W3C Recommendation, January 1999, http://www.w3.org/TR/1999/REC-xml-names-19990114/

[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/

[XMLSCHEMA1] Thompson, H., Beech, D., Maloney, M., and Mendelsohn, N., Eds., "XML Schema Part 1: Structures", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

[XPATH] Clark, J., and DeRose, S., "XML Path Language (XPath), Version 1.0", W3C Recommendation, November 1999, http://www.w3.org/TR/xpath/

1.2.2 Informative References

[MSDN-XPATH] Microsoft Corporation, "Microsoft XPath Extension Functions", http://msdn.microsoft.com/en-us/library/ms256453.aspx

[MSDN-XSF] Microsoft Corporation, "InfoPath 2007 XSF Schema Reference", http://msdn.microsoft.com/en-us/library/bb265224.aspx

1.3 Structure Overview (Synopsis)

The InfoPath form template format is a **form template (.xsn) file**. A form template (.xsn) file is a **cabinet (.cab) file**, as described in [MC-MCF], that contains files used by a **form server** to render and edit new or existing **form files** in a Web browser. Form files are used to store the data of a **form** that has been filled out.

Certain files need to be included within the form template (.xsn) file for a form server to correctly open or create a form file. Other files are included only in certain scenarios.

The following figure illustrates a typical form template (.xsn) file.

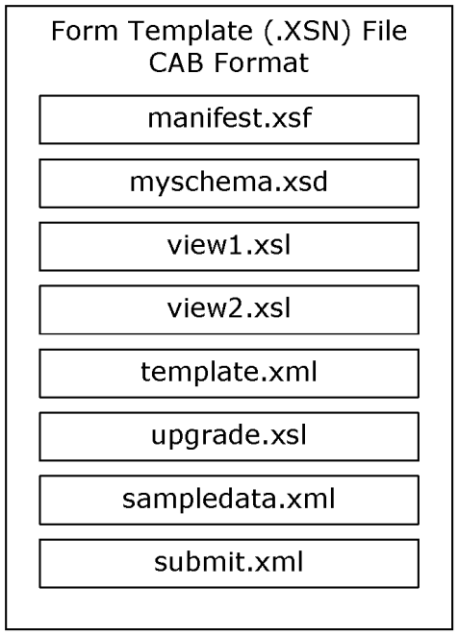


Figure 1: A typical form template (.xsn) file

The following sections describe the files that can be found in a form template (.xsn) file and how they are used to render a form.

The form template (.xsn) file is specified in section 2.1. Examples are provided in section 3.1.

1.3.1 Form Definition File (XSF)

The **form definition (.xsf) file**, manifest.xsf, specifies information about the **form template**, including the following:

- The **list** of files that comprise the form template (.xsn) file and the relationships between them.
- Properties of the form template, such as deployment information and user interface customizations.
- Properties of certain **controls** within the form template. For example, a control can have associated **rules** that perform tasks automatically based on events and values.

The form definition (.xsf) file is specified in section 2.2. Examples are provided in section 3.2.

1.3.2 XML Schema Files (XSD)

There are one or more **XML schema documents** in a form template (.xsn) file:

- Typically there is one primary XSD file, often called myschema.xsd, which specifies the **XML schema** that all form files based on the form template conform to.
- One or more secondary XSD files that specify the XML schemas for **data connections** used in the form template.

The form definition (.xsf) file specifies which XSD file is the primary one and which ones are associated with data connections.

The structure of the primary XSD file is specified in section 2.3. The function of each XSD file is specified in section 2.2.60. Examples are provided in section 3.3.

1.3.3 Form View Files (XSLT)

A **form view** is a specific visualization of a form file in a Web browser. It specifies what controls are used to represent the fields in the form and how they are laid out. It also provides information to the form server about the editing behavior for each control. For example, it provides information about what actions to take when a particular field's value changes.

A form view is represented by an **XSL Transformation (XSLT)** file, which uses the .XSL file extension. There are one or more XSLT files in the form template:

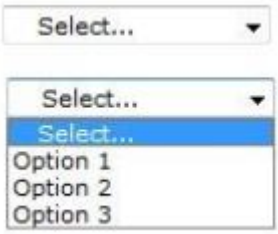
- One default XSLT file, for example view1.xsl, used to render a form file when it is first opened or created.
- Other XSLT files, for example view2.xsl, that can be used to render the form as it is edited.

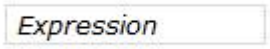
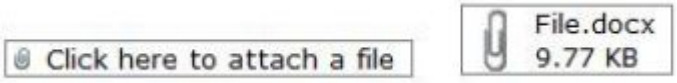
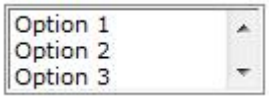
The default XSLT file is specified in the form definition (.xsf) file.

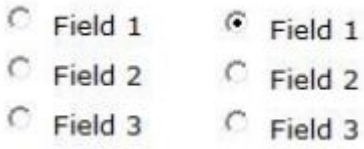
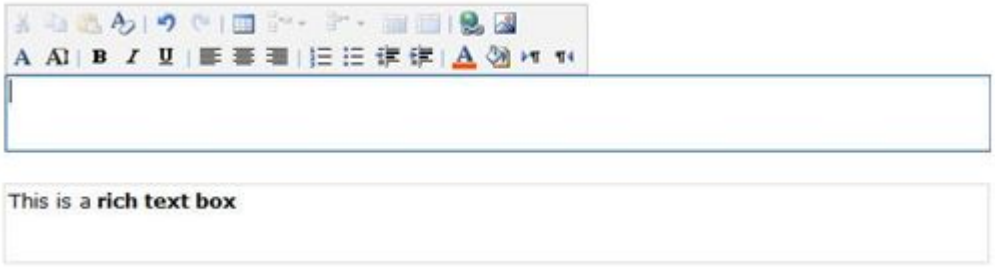
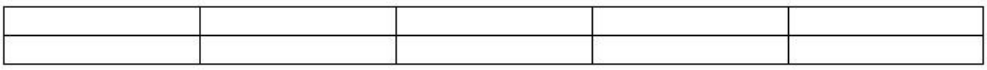
The contents of an XSLT file are used to transform the data in a form file into **HTML**, so that the form can be rendered and edited in a Web browser. The resulting HTML visualization consists of HTML used to lay out and represent the controls and informative text. The HTML for a control also contains properties specifying its behavior.

The following table lists the controls that can be used in a form. Controls are usually tied to specific fields in the form file and are used to edit such fields. Some controls, as noted in the table, are not the direct representation of any particular fields.

Control	Description and sample visual representation
Button	A control used to execute an action. It is not tied to a field. The following figure illustrates a typical representation of a button control.  **Figure 2: A button control** The visualization and properties of the control are specified in section 2.4.1.5. The relationship to the XML schema of the form template is specified in section 2.3.1.1. Behaviors affecting the control are specified in section 2.2.144 and 2.2.62.
Check Box	A control that allows users to set yes/no or true/false values by adding or removing a check mark from a small square box. The following figure illustrates a typical representation of a check box control.  **Figure 3: A set of check box controls** The visualization and properties of the control are specified in section 2.4.1.6. The relationship to the XML schema of the form template is specified in section 2.3.1.2. Behaviors affecting the control are specified in section 2.2.144 and 2.2.62.

Control	Description and sample visual representation
Contact Selector	A control that allows users to select one or more contacts from a protocol server user list. The following figure illustrates a typical representation of a contact selector control.  **Figure 4: A contact selector control** The visualization and properties of the control are specified in section 2.4.1.7. The relationship to the XML schema of the form template is specified in section 2.3.1.3. Behaviors affecting the control are specified in section 2.2.144 and 2.2.62.
Date Picker	A control that contains a box where users can type dates and a calendar button that allows users to select a date. The following figure illustrates a typical representation of a date picker control.  **Figure 5: A date picker control** The visualization and properties of the control are specified in section 2.4.1.8. The relationship to the XML schema of the form template is specified in section 2.3.1.4. Behaviors affecting the control are specified in section 2.2.144 and 2.2.62.
Drop-Down List Box	A control that presents users with a list of choices in a box. To select an item from the list (1), users click an arrow to open the list (1) of choices. The following figure illustrates a typical representation of a drop-down list box control.  **Figure 6: A drop-down list box control** The visualization and properties of the control are specified in section 2.4.1.9. The relationship to the XML schema of the form template is specified in section 2.3.1.5. Behaviors affecting the control are specified in section 2.2.144 and 2.2.62.

Control	Description and sample visual representation
Expression Box	A read-only text control used to display information. The following figure illustrates a typical representation of an expression box control.  **Figure 7: An expression box control** The visualization and properties of the control are specified in section 2.4.1.10. The relationship to the XML schema of the form template is specified in section 2.3.1.6. Behaviors affecting the control are specified in section 2.2.144 and 2.2.62.
File Attachment	A control that allows users to attach files to their form files. Each file attachment control permits one file to be attached. The following figure illustrates a typical representation of a file attachment control.  **Figure 8: A file attachment control** The visualization and properties of the control are specified in section 2.4.1.11. The relationship to the XML schema of the form template is specified in section 2.3.1.7. Behaviors affecting the control are specified in section 2.2.144 and 2.2.62.
Hyperlink	A control that can be used to link to a **Uniform Resource Locator (URL)**. The following figure illustrates a typical representation of a hyperlink.  **Figure 9: A hyperlink control** The visualization and properties of the control are specified in section 2.4.1.12. The relationship to the XML schema of the form template is specified in section 2.3.1.8. Behaviors affecting the control are specified in section 2.2.144 and 2.2.62.
List Box	A control that presents users with a list (1) of choices in a box from which users select the appropriate item. The following figure illustrates a typical representation of a list box control.  **Figure 10: A list box control** The visualization and properties of the control are specified in section 2.4.1.13. The relationship to the XML schema of the form template is specified in section 2.3.1.9. Behaviors affecting the control are specified in section 2.2.144 and 2.2.62.

Control	Description and sample visual representation
Option Button	A control that lets users select from a set of mutually exclusive choices. When one option button in a **group** is selected, the other option buttons are cleared. The following figure illustrates a typical representation of an option button control.  **Figure 11: A set of option button controls** The visualization and properties of the control are specified in section 2.4.1.14. The relationship to the XML schema of the form template is specified in section 2.3.1.10. Behaviors affecting the control are specified in section 2.2.144 and 2.2.62.
Rich Text Box	A text input control that can contain formatted text, including bold and italic text, and a variety of fonts, font sizes, and font colors. The following figure illustrates a typical representation of a rich text box control.  **Figure 12: A rich text box control, the top image shows it in the editing state** The visualization and properties of the control are specified in section 2.4.1.17. The relationship to the XML schema of the form template is specified in section 2.3.1.13. Behaviors affecting the control are specified in section 2.2.144 and 2.2.62.
Table	A table used to lay out the form. It is not tied to a field. The following figure illustrates a typical representation of a table control.  **Figure 13: A table** The visualization and properties of the control are specified in section 2.4.1.19. The relationship to the XML schema of the form template is specified in section 2.3.1.15. Behaviors affecting the control are specified in section 2.2.144 and 2.2.62.

Control	Description and sample visual representation
Text Box	A text input control that can contain any unformatted text. Text box controls cannot contain formatted text. The following figure illustrates a typical representation of a text box control.  **Figure 14: A text box control** The visualization and properties of the control are specified in section 2.4.1.20. The relationship to the XML schema of the form template is specified in section 2.3.1.16. Behaviors affecting the control are specified in section 2.2.144 and 2.2.62.
Repeating Table	A control that displays repeating information in a tabular structure. Each item appears in a new row in the repeating table control. When filling out a form, users can add or delete rows in a repeating table control as necessary. Repeating table controls can contain other controls. The following figure illustrates a typical representation of a repeating table control.  **Figure 15: A repeating table control with three text box controls per row** The visualization and properties of the control are specified in section 2.4.1.16. The relationship to the XML schema of the form template is specified in section 2.3.1.12. Behaviors affecting the control are specified in section 2.2.144 and 2.2.62.
Section, Optional Section	A control that is a container for other controls. An optional section control is the same as the section control, but does not need to be initially displayed in the form. The following figure illustrates a typical representation of a section control.  **Figure 16: A section control containing two text box controls** The visualization and properties of the control are specified in section 2.4.1.18. The relationship to the XML schema of the form template is specified in section 2.3.1.14. Behaviors affecting the control are specified in section 2.2.144 and 2.2.62.
Repeating Section	A control that is a container for other controls and is used to display repeating information. When filling out the form that includes a repeating section control, users can add additional occurrences of the repeating section control. The following figure illustrates a typical representation of a repeating section control.  **Figure 17: A repeating section control** The visualization and properties of the control are specified in section 2.4.1.15. The relationship to the XML schema of the form template is specified in section 2.3.1.11. Behaviors affecting the control are specified in section 2.2.144 and 2.2.62.

The structure of an XSLT file that defines a form view is specified in section 2.4. The role of each XSLT file is specified in section 2.2.122. Examples are provided in section 3.4.

1.3.4 Print View Files (XSLT)

Print views are visualizations of the form used for printing. A print view is defined by an XSLT file that is optimized for printing the data instead of visualizing it in a Web browser. For example, a print view could be defined to use dark text on a white background or suppress borders on the controls.

A print view is always associated with a form view so that when the form server is displaying the form view and the user chooses to print the form, the print view is sent to the printer.

The association between a form view and a print view is specified in the form definition (.xsf) file.

The structure of an XSLT file that defines a print view is specified in section 2.5. The role of each XSLT file is specified in section 2.2.122. Examples are provided in section 3.5.

1.3.5 Submit Files (XML)

Submit files are XML files, as described in [W3C-XML], that specify the information used to submit form data to a **Web service**. This information is only needed when the form has behaviors that submit data to a Web service, although there are Web services for which no submit file is needed.

This information is composed of two parts, the submit files and form definition (.xsf) file, as follows:

- The submit files contain XML templates based on the parameters required by the Web service methods.
- The form definition (.xsf) file contains references to submit files and specifies a mapping between fields and parameters to the Web service methods.

Submit files are specified in section 2.6. Examples are provided in section 3.6.

1.3.6 Template.XML File

The template.xml file is a form file based on the form template that contains it. It is used to store and load initial values of the fields when creating a new form file based on the form template. For example, it is used in creating a new expense report based on an expense report template. This file is used only when creating a new form file based on the form template.

Template.xml is specified in section 2.7. Examples are provided in section 3.7.

1.3.7 Upgrade.XSL File

The upgrade.xsl file is an XSLT file used to upgrade an existing form file if a newer version of the form template becomes available.

When upgrade.xsl is present in a form template (.xsn) file, the form server applies it to transform a form file created with an older version of the associated form template to match the latest version. Upgrade.xsl is not used when creating new form files.

When applied, the upgrade.xsl transform does the following:

1. Copies fields from the form file to the upgraded one.
2. Removes fields that are no longer used.
3. Adds new fields that have been added to the newer version of the form template.

Once the transform has been applied, the resulting form file has to be usable by the form server.

Upgrade.xsl is specified in section 2.8. Examples are provided in section 3.8.

1.3.8 Resource Files

The following files are present for the form templates:

- **Business object** files used to run form code.
- Other resource files, which can be images or file **attachments**.

The list of files in the form template is specified in sections 2.1 and 2.2.97. Examples are provided in section 3.1.

1.3.9 Unused Files

The following files are never used by a form server to render or edit a form, but are often present:

- The schema_offline.xml files are used for storing data from data connections so that they can be accessed **offline**.
- The merge.xsl file contains an XSLT, as described in [W3C-XSLT], that can be used to combine multiple form files into a single form file.
- The sampledata.xml file is an XML file, as described in [W3C-XML] containing sample data for the form.
- The script.js and script.vbs files are used for scripting events.
- The irm_template file is used for **Information Rights Management (IRM)**.
- The importerrors.xml file contain errors resulting from importing files.

The list of files in the form template is specified in sections 2.1 and 2.2.97.

1.4 Relationship to Protocols and Other Structures

The InfoPath Form Template format is an extension of the Cabinet file format, described in [MC-MCF].

All XSLT files contained in a form template (.xsn) file are XSLTs files, as described in [W3C-XSLT].

All XML schema (.xsd) files contained in a form template (.xsn) file are XSD files, as described in [XMLSCHEMA1].

Template.xml is a form file, as described in [MS-IPFFX].

1.5 Applicability Statement

This structure is used by a form server to render and edit forms based on a form template. The form is rendered and edited using a Web browser.

1.6 Versioning and Localization

This document covers versioning issues in the following areas:

Structure Versions: This structure specifies the only version of the InfoPath Form Template Format.

Localization: This structure specifies no locale-specific processes or data.

1.7 Vendor-Extensible Fields

The InfoPath Form Template Format defines vendor-extensible fields as specified by the **solutionDefinition** element, as described in section 2.2.147.8.

2 Structures

2.1 The InfoPath Form Template Format

A form template (.xsn) file MUST be a cabinet (.cab) file, as specified in [MC-MCF], containing other files used by form servers to display forms. The following subsections list files that could appear in a form template (.xsn) file.

The name of the form template (.xsn) file MUST end with the .xsn file extension and MUST contain **Unicode UTF-16** characters, as specified in [RFC2781]. The form template (.xsn) file name MUST NOT contain following characters: " # % & * : < > ? { | } ~. The form template (.xsn) file name MUST NOT contain characters that have different hexadecimal values than "0x00-0x1F and 0x7F-0x9F".

2.1.1 manifest.xsf

A form template (.xsn) file MUST include manifest.xsf, the form definition (.xsf) file, and this file MUST be the first one in the form template (.xsn) file. The form definition (.xsf) file specifies the other files that are to appear in the form template (.xsn) file. All files in the form template (.xsn) file MUST be specified in the form definition (.xsf) file other than the form definition (.xsf) file itself. See section 2.2 and section 2.2.97.

2.1.2 primaryschema.xsd

A form template (.xsn) file MUST contain *primaryschema*.xsd. *primaryschema* MUST conform to the naming conventions for files stored in a cabinet (.cab) file. See section 2.3.

2.1.3 view.xsl

A form template (.xsn) file MUST include at least one *view*.xsl. *view* MUST conform to the naming conventions for files stored in a cabinet (.cab) file. *view*.xsl files conform to the **XSL** specification in section 2.4. Also see section 2.5.

2.1.4 sampledata.xml

A form template (.xsn) file MUST include sampledata.xml. Contents of the sampledata.xml **file** MUST be ignored by the form server.

2.1.5 template.xml

A form template (.xsn) file MUST include template.xml. See section 2.7.

2.1.6 secondaryschema.xsd

A form template (.xsn) file MUST contain one or more *secondaryschema*.xsd files if there are data connections within the form template. *secondaryschema* MUST conform to the naming conventions for files stored in a cabinet (.cab) file. See section 2.3 and 2.2.147.28.

2.1.7 submitdata.xml

A form template (.xsn) file can contain one or more *submitdata*.xml files if there are data connections within the form template. *submitdata* MUST conform to the naming conventions for files stored in a cabinet (.cab) file. See section 2.6.

2.1.8 upgrade.xsl

A form template (.xsn) file SHOULD contain upgrade.xsl if there are multiple versions of the form template published. See section 2.8.

2.1.9 merge.xsl

A form template (.xsn) file MAY contain merge.xsl. merge.xsl MUST be ignored by the form server.

2.1.10 secondaryschema_offline.xml

A form template (.xsn) file MAY contain one or more *secondaryschema*_offline.xml files that are associated with a *secondaryschema*.xsd. *secondaryschema*_offline.xml files MUST be ignored by the form server.

2.1.11 Business Object

A form template (.xsn) file MUST contain a business object file if there is a business object associated with the form template. The business object file MUST conform to the naming conventions for files stored in a cabinet (.cab) file. See section 2.2.147.50.

2.1.12 script.js

A form template (.xsn) file MAY contain one or more *script*.js files. See section 2.2.33.

2.1.13 script.vbs

A form template (.xsn) file MAY contain one or more *script*.vbs files. See section 2.2.33.

2.1.14 importerrors.xml

A form template (.xsn) file MAY contain importerrors.xml. importerrors.xml MUST be ignored.

2.1.15 irm_template

A form template (.xsn) file MAY contain irm_template. irm_template MUST NOT be present.

2.1.16 Resource Files

A form template (.xsn) file MAY contain other files specified in the form definition (.xsf) file. These files MUST conform to the naming conventions for files stored in a cabinet (.cab) file. Resource files MAY include files that would cause the form server to reject the form template (.xsn) file or return an error in other circumstances. See section 2.2.97.

2.2 Form Definition File (XSF) Specification

The form definition (.xsf) file specifies the properties, content, and files of the form template. It MUST conform to the form definition (.xsf) file XML schema, as defined by the types and elements in the following table. The form definition (.xsf) file XML schema is used to validate the elements, attributes, and types in the xsf namespace:
`http://schemas.microsoft.com/office/infopath/2003/solutionDefinition`.

The **xDocumentClass** element, as defined in section 2.2.20, MUST be the root element of the form definition (.xsf) file.

The following tables list, in alphabetical order, the types and elements used in the XML schema for the form definition (.xsf) file.

The XML schema is extended by the additional types and elements specified in the Form Definition File (XSF) Extension Specification, as specified in section 2.2.147.

Type	Specified in Section
xdDesignMode	2.2.13
xdEmptyString	2.2.11
xdEnabledDisabled	2.2.5
xdErrorMessage	2.2.12
xdExpressionLiteral	2.2.7
xdFileName	2.2.8
xdHWSCaption	2.2.19
xdHWSname	2.2.18
xdManualAuto	2.2.6
xdRoleName	2.2.3
xdScriptLanguage	2.2.9
xdSignatureRelationEnum	2.2.17
xdSignedDataBlockMessage	2.2.16
xdSignedDataBlockName	2.2.15
xdSolutionVersion	2.2.10
xdTitle	2.2.1
xdTrustLevel	2.2.14
xdViewName	2.2.2
xdYesNo	2.2.4

Element	Specified in Section
action	2.2.90
adoAdapter	2.2.38
allowedActions	2.2.89
allowedControl	2.2.102
allowedTasks	2.2.91
applicationParameters	2.2.23
assignmentAction	2.2.139
attachmentFileName	2.2.57
attributeData	2.2.105
autoRecovery	2.2.31
bcc	2.2.54
button	2.2.106
button	2.2.110
calculatedField	2.2.146
calculations	2.2.145
cc	2.2.53
chooseFragment	2.2.107
closeDocumentAction	2.2.142
customCategory	2.2.95

Element	Specified in Section
customValidation	2.2.62
dataAdapters	2.2.59
dataObject	2.2.36
dataObjects	2.2.35
davAdapter	2.2.48
dialogBoxExpressionAction	2.2.137
dialogBoxMessageAction	2.2.136
documentSchema	2.2.61
documentSchemas	2.2.60
documentSignatures	2.2.125
documentVersionUpgrade	2.2.128
domEventHandler	2.2.66
domEventHandlers	2.2.65
editing	2.2.111
editWith	2.2.108
emailAdapter	2.2.51
errorCondition	2.2.63
errorMessage	2.2.64
errorMessage	2.2.75
exitRuleSet	2.2.135
exportToExcel	2.2.28
exportToWeb	2.2.27
extension	2.2.131
extensions	2.2.130
externalView	2.2.104
externalViews	2.2.103
featureRestrictions	2.2.25
field	2.2.47
field	2.2.71
fields	2.2.70
file	2.2.98
fileName	2.2.50
fileNew	2.2.93
fileProperties	2.2.99
files	2.2.97
folderURL	2.2.49
footer	2.2.117
fragmentToInsert	2.2.113
getUserNameFromData	2.2.84
group	2.2.86
header	2.2.116
hwsAdapter	2.2.40
hwsOperation	2.2.42
hwsWorkflow	2.2.87
importParameters	2.2.67
importSource	2.2.68

Element	Specified in Section
initialXmlDocument	2.2.94
input	2.2.43
intro	2.2.56
listProperties	2.2.69
location	2.2.88
mainpane	2.2.114
masterDetail	2.2.112
membership	2.2.83
menu	2.2.119
menuArea	2.2.120
message	2.2.127
onLoad	2.2.79
openNewDocumentAction	2.2.141
operation	2.2.41
override	2.2.22
package	2.2.96
partFragment	2.2.44
permissions	2.2.101
print	2.2.29
printSettings	2.2.115
property	2.2.100
query	2.2.32
query	2.2.37
queryAction	2.2.140
role	2.2.82
roles	2.2.81
rule	2.2.133
ruleSet	2.2.143
ruleSetAction	2.2.132
ruleSets	2.2.144
save	2.2.26
save	2.2.80
schemaErrorMessages	2.2.21
script	2.2.34
scripts	2.2.33
sendMail	2.2.30
sharepointListAdapter	2.2.46
signedDataBlock	2.2.126
solutionProperties	2.2.24
subject	2.2.55
submit	2.2.72
submitAction	2.2.73
submitAction	2.2.134
submitToHostAdapter	2.2.58
successMessage	2.2.74
switchViewAction	2.2.138

Element	Specified in Section
task	2.2.92
taskpane	2.2.121
to	2.2.52
toolbar	2.2.118
unboundControls	2.2.109
useHttpHandler	2.2.76
useQueryAdapter	2.2.78
userName	2.2.85
useScriptHandler	2.2.77
useTransform	2.2.129
view	2.2.123
views	2.2.122
webServiceAdapter	2.2.39
xDocumentClass	2.2.20
xmlFileAdapter	2.2.45
xmlToEdit	2.2.124

2.2.1 xdTitle

The **xdTitle** simple type specifies restrictions for a title string.

Referenced By
adoAdapter.xsfschema@name
adoAdapterExtension.xsf2.2.147.32@ref
adoAdapterExtension.xsf2.2.147.32@submitAdapterName
button.xsfschema@caption
button.xsfschema@tooltip
command.xsf2.2.147.12@caption
customCategory.xsfschema@name
dataObject.xsfschema@name
davAdapter.xsfschema@name
davAdapterExtension.xsf2.2.147.31@ref
editWith.xsfschema@caption
emailAdapter.xsfschema@name
emailAdapterExtension.xsf2.2.147.35@ref
field.xsfschema@columnName
field.xsfschema@name

Referenced By
hwsAdapter.xsfschema@name
initialXmlDocument.xsfschema@caption
inputScope.xsf2.2.147.47@caption
menu.xsfschema@caption
sharepointListAdapter.xsfschema@name
sharepointListAdapterExtension.xsf2.2.147.37@ref
submitAction.rule.xsfschema@adapter
submitAction.submit.xsf2.2.147.52@adapter
submitAction.submit.xsfschema@adapter
submitToHostAdapter.xsfschema@name
toolbar.xsfschema@caption
toolbar.xsfschema@name
viewExtension.xsf2.2.147.42@ref
webServiceAdapter.xsfschema@name
webServiceAdapterExtension.xsf2.2.147.33@ref
xmlFileAdapter.xsfschema@name
xmlFileAdapterExtension.xsf2.2.147.36@ref
xmlToEditExtension.xsf2.2.147.43@ref

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this simple type.

```
<xsd:simpleType name="xdTitle">
  <xsd:restriction base="xsd:string">
    <xsd:minLength value="1"/>
    <xsd:maxLength value="255"/>
    <xsd:pattern
value="([^\p{Z}\p{Cc}\p{Cf}\p{Cn}])(([^\p{Zl}\p{Zp}\p{Cc}])*([^\p{Z}\p{Cc}\p{Cf}\p{Cn}]))?"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.2 xdViewName

The **xdViewName** simple type specifies restrictions for specifying the name of a form view.

Referenced By
externalView.xsfschema@name

Referenced By
includedView.xsf2.2.147.23@name
role.xsfschema@name
switchViewAction.xsfschema@view
view.xsfschema@caption
view.xsfschema@name

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this simple type.

```
<xsd:simpleType name="xdViewName">
  <xsd:restriction base="xsd:string">
    <xsd:minLength value="1"/>
    <xsd:maxLength value="255"/>
    <xsd:pattern
value="([^\p{Z}\p{C}/\\#&amp;&amp;quot;&amp;gt;&amp;lt;])(([^\p{Zl}\p{Zp}\p{C}/\\#&amp;&amp;quot;&amp;gt;&amp;lt;])*([^\p{Z}\
p{C}/\\#&amp;&amp;quot;&amp;gt;&amp;lt;]))?"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.3 xdRoleName

The **xdRoleName** simple type specifies restrictions for an attribute that MUST NOT be present.

Referenced By
role.xsfschema@name

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this simple type.

```
<xsd:simpleType name="xdRoleName">
  <xsd:restriction base="xsf:xdViewName"/>
</xsd:simpleType>
```

2.2.4 xdYesNo

The **xdYesNo** simple type specifies enumeration values for specifying a "yes" or "no" value.

no: This enumeration value evaluates to "no".

yes: This enumeration value evaluates to "yes".

Referenced By
action.xsfschema@canInitiateWorkflow
adoAdapter.xsfschema@queryAllowed
adoAdapter.xsfschema@submitAllowed
autoUpdatePrompt.xsf2.2.147.45@showPrompt
calculations.xsfschema@treatBlankValueAsZero
closeDocumentAction.xsfschema@promptToSaveChanges
contentTypeTemplate.xsf2.2.147.17@browserEnable
dataObject.xsfschema@initOnLoad
davAdapter.xsfschema@overwriteAllowed
davAdapter.xsfschema@queryAllowed
davAdapter.xsfschema@submitAllowed
documentSchema.xsfschema@rootSchema
editWith.xsfschema@autoComplete
editWith.xsfschema@proofing
emailAdapter.xsfschema@queryAllowed
emailAdapter.xsfschema@submitAllowed
field.sharepointListAdapter.xsfschema@isLookup
field.xsfschema@required
field.xsfschema@viewable
fieldExtension.xsf2.2.147.27@readWrite
hwsAdapter.xsfschema@queryAllowed
hwsAdapter.xsfschema@submitAllowed
hwsWorkflow.xsfschema@taskpaneVisible
importParameters.xsfschema@enabled
importParameters.xsfschema@useScriptHandler
managedCode.xsf2.2.147.50@enabled
mergedPrintView.xsf2.2.147.21@isCustomizable
mergedPrintView.xsf2.2.147.21@isDefault
offline.xsf2.2.147.24@cacheQueries
offline.xsf2.2.147.24@openIfQueryFails
partFragment.xsfschema@sendAsString
printSettings.xsfschema@collate

Referenced By
roles.xsfschema@hideStatusBarDisplay
rule.xsfschema@isEnabled
scripts.xsfschema@enforceScriptTimeout
sendByMail.xsf2.2.147.38@disableEmailForms
server.xsf2.2.147.9@isMobileEnabled
server.xsf2.2.147.9@isPreSubmitPostBackEnabled
sharepointListAdapter.xsfschema@queryAllowed
sharepointListAdapter.xsfschema@submitAllowed
sharepointListAdapterExtension.xsf2.2.147.37@queryThisFormOnly
solutionDefinition.xsf2.2.147.8@allowClientOnlyCode
solutionDefinition.xsf2.2.147.8@verifyOnServer
solutionProperties.xsfschema@allowCustomization
solutionProperties.xsfschema@automaticallyCreateNodes
submit.xsf2.2.147.51@disableMenuItem
submit.xsf2.2.147.51@showSignatureReminder
submit.xsf2.2.147.51@showStatusDialog
submit.xsfschema@disableMenuItem
submit.xsfschema@showSignatureReminder
submit.xsfschema@showStatusDialog
submitToHostAdapter.xsfschema@queryAllowed
submitToHostAdapter.xsfschema@submitAllowed
toolbar.xsf2.2.147.10@enabledBottom
toolbar.xsf2.2.147.10@enabledTop
view.xsfschema@showMenuItem
viewExtension.xsf2.2.147.42@clientOnly
viewExtension.xsf2.2.147.42@readOnly
warning.xsf2.2.147.40@hidden
webServiceAdapter.xsfschema@queryAllowed
webServiceAdapter.xsfschema@submitAllowed
webServiceAdapter.xsfschema@useDataSet
webServiceAdapterExtension.xsf2.2.147.33@trackDataSetChanges
wss.xsf2.2.147.15@browserEnable

Referenced By
xDocumentClass.xsfschema@dataFormSolution
xDocumentClass.xsfschema@requireFullTrust
xmlToEditExtension.xsf2.2.147.43@allowLinkedImages
xmlToEditExtension.xsf2.2.147.43@excludeEmbeddedImages

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this simple type.

```
<xsd:simpleType name="xdYesNo">
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:enumeration value="yes"/>
    <xsd:enumeration value="no"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.5 xdEnabledDisabled

The **xdEnabledDisabled** simple type specifies enumeration values for specifying an "enabled" or "disabled" value.

disabled: This enumeration value evaluates to "disabled".

enabled: This enumeration value evaluates to "enabled".

Referenced By
autoRecovery.xsfschema@feature
exportToExcel.xsfschema@ui
exportToPDForXPS.xsf2.2.147.56@ui
exportToWeb.xsfschema@ui
print.xsfschema@ui
save.featureRestrictions.xsfschema@ui
sendMail.xsfschema@ui

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this simple type.

```
<xsd:simpleType name="xdEnabledDisabled">
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:enumeration value="enabled"/>
    <xsd:enumeration value="disabled"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.6 xdManualAuto

The **xdManualAuto** simple type specifies enumeration values for specifying a "manual" or "automatic" value.

automatic: This enumeration value evaluates to "automatic".

manual: This enumeration value evaluates to "manual".

Referenced By
importSource.xsfschema@authoringOfTransform
xDocumentClass.xsfschema@trustSetting

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this simple type.

```
<xsd:simpleType name="xdManualAuto">
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:enumeration value="manual"/>
    <xsd:enumeration value="automatic"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.7 xdExpressionLiteral

The **xdExpressionLiteral** simple type specifies enumeration values for specifying whether the corresponding value is an **XPath expression** or a literal string.

expression: This enumeration value specifies that the corresponding value evaluates to an XPath expression.

literal: This enumeration value specifies that the corresponding value evaluates to a literal string.

Referenced By
attachmentFileName.emailAdapter.xsfschem@valueType
bcc.emailAdapter.xsfschema@valueType
cc.emailAdapter.xsfschema@valueType
fileName.davAdapter.xsfschema@valueType
subject.emailAdapter.xsfschema@valueType
to.emailAdapter.xsfschema@valueType

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this simple type.

```
<xsd:simpleType name="xdExpressionLiteral">
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:enumeration value="expression"/>
    <xsd:enumeration value="literal"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.8 xdFileName

The **xdFileName** simple type specifies restrictions for specifying the name of a file that is part of the form template.

Referenced By
file.xsfschema@name
importSource.xsfschema@schema
importSource.xsfschema@transform
initialXmlDocument.xsfschema@href
mainpane.xsfschema@transform
script.xsfschema@src
solutionProperties.xsfschema@lastOpenView
useTransform.xsfschema@transform

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this simple type.

```
<xsd:simpleType name="xdFileName">
  <xsd:restriction base="xsd:string">
    <xsd:minLength value="1"/>
    <xsd:maxLength value="64"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.9 xdScriptLanguage

The **xdScriptLanguage** simple type specifies restrictions for the **language** and **scriptLanguage** attributes.

The **language** attribute of the **scripts** element, as defined in section 2.2.33, MUST NOT be present. The **scriptLanguage** attribute of the **solutionProperties** element, as defined in section 2.2.24, MUST be ignored.

Referenced By
scripts.xsfschema@language

Referenced By
solutionProperties.xsfschema@scriptLanguage

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this simple type.

```
<xsd:simpleType name="xdScriptLanguage">
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:pattern
value="((([Jj][Aa][Vv][Aa]|(([Jj])|([Vv][Bb])))([Ss][Cc][Rr][Ii][Pp][Tt]))([.][Ee][Nn][Cc][Oo
][Dd][Ee]))|(([Jj][Aa][Vv][Aa]|(([Jj])|([Vv][Bb])))([Ss][Cc][Rr][Ii][Pp][Tt]))|([Mm][Aa][Nn][
Aa][Gg][Ee][Dd][Cc][Oo][Dd][Ee])"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.10 xdSolutionVersion

The **xdSolutionVersion** simple type specifies restrictions for specifying the version of the form template.

Referenced By
solutionProperties.xsfschema@lastVersionNeedingTransform
useTransform.xsfschema@maxVersionToUpgrade
useTransform.xsfschema@minVersionToUpgrade
xDocumentClass.xsfschema@solutionFormatVersion
xDocumentClass.xsfschema@solutionVersion

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this simple type.

```
<xsd:simpleType name="xdSolutionVersion">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="(([0-9]{1,4}.){3}[0-9]{1,4})"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.11 xdEmptyString

The **xdEmptyString** simple type specifies restrictions for specifying an **empty string**.

Referenced By
useTransform.xsfschema@transform

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this simple type.

```
<xsd:simpleType name="xdEmptyString">
  <xsd:restriction base="xsd:string">
    <xsd:maxLength value="0"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.12 xdErrorMessage

The **xdErrorMessage** simple type specifies restrictions for specifying an error message.

Referenced By
errorMessage.xsfschema

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this simple type.

```
<xsd:simpleType name="xdErrorMessage">
  <xsd:restriction base="xsd:string">
    <xsd:maxLength value="1023"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.13 xdDesignMode

The **xdDesignMode** simple type specifies enumeration values for specifying a "normal" or "protected" value.

normal: This enumeration value evaluates to "normal".

protected: This enumeration value evaluates to "protected".

Referenced By
externalView.xsfschema@designMode
view.xsfschema@designMode

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this simple type.

```
<xsd:simpleType name="xdDesignMode">
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:enumeration value="normal"/>
    <xsd:enumeration value="protected"/>
```

```
  </xsd:restriction>
</xsd:simpleType>
```

2.2.14 xdTrustLevel

The **xdTrustLevel** simple type specifies enumeration values for specifying a "restricted" or "domain" value.

domain: This enumeration value evaluates to "domain".

restricted: This enumeration value evaluates to "restricted". This value MUST NOT be present.

Referenced By
xDocumentClass.xsfschema@trustLevel

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this simple type.

```
<xsd:simpleType name="xdTrustLevel">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="restricted"/>
    <xsd:enumeration value="domain"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.15 xdSignedDataBlockName

The **xdSignedDataBlockName** simple type specifies restrictions for specifying the name of a signed data block.

Referenced By
signedDataBlock.xsfschema@name

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this simple type.

```
<xsd:simpleType name="xdSignedDataBlockName">
  <xsd:restriction base="xsd:ID">
    <xsd:minLength value="1"/>
    <xsd:maxLength value="255"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.16 xdSignedDataBlockMessage

The **xdSignedDataBlockMessage** simple type specifies restrictions for specifying the confirmation message that is displayed when a **digital signature** is applied to the form or section of the form.

Referenced By
message.signedDataBlock.xsfschema

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this simple type.

```
<xsd:simpleType name="xdSignedDataBlockMessage">
  <xsd:restriction base="xsd:string">
    <xsd:maxLength value="255"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.17 xdSignatureRelationEnum

The **xdSignatureRelationEnum** simple type specifies enumeration values for specifying a "countersign", "cosign", or "single" value.

cosign: This enumeration value evaluates to "cosign".

countersign: This enumeration value evaluates to "countersign".

single: This enumeration value evaluates to "single".

Referenced By
signedDataBlock.xsfschema@mode

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this simple type.

```
<xsd:simpleType name="xdSignatureRelationEnum">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="countersign"/>
    <xsd:enumeration value="cosign"/>
    <xsd:enumeration value="single"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.18 xdHWSname

The **xdHWSname** simple type specifies restrictions for an attribute that MUST NOT be present.

Referenced By
action.xsfschema@name
task.xsfschema@name

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this simple type.

```
<xsd:simpleType name="xdHWSname">
  <xsd:restriction base="xsd:NCName">
    <xsd:pattern value="[^-^\.^\\^\[^\]^\|^\+^?^\*^@^\{^\}^\(^\)^&gt;^&lt;^=^;^,]*"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.19 xdHWSCaption

The **xdHWSCaption** simple type specifies restrictions for an attribute that MUST NOT be present.

Referenced By
action.xsfschema@caption
task.xsfschema@caption

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this simple type.

```
<xsd:simpleType name="xdHWSCaption">
  <xsd:restriction base="xsd:string">
    <xsd:minLength value="1"/>
    <xsd:maxLength value="255"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.20 xDocumentClass

The **xDocumentClass** element and its child elements specify the properties, appearance, content, and files of the form template. The values specified by this element determine the general behaviors, such as loading and upgrading, of all forms generated from the form template. This element MUST be the root element of the form definition (.xsf) file.

Child Elements
applicationParameters
calculations
customValidation

Child Elements
dataAdapters
dataObjects
documentSchemas
documentSignatures
documentVersionUpgrade
domEventHandlers
extensions
externalViews
featureRestrictions
fileNew
hwsWorkflow
importParameters
listProperties
onLoad
package
permissions
query
roles
ruleSets
save
schemaErrorMessages
scripts
submit
taskpane
views

Attributes:

author: The name of the author of the form template. If this attribute is not present, its value MUST be interpreted as an empty string.

dataFormSolution: Specifies whether a form template was designed based on a **main data connection** to a database or Web service. If this attribute is not present, its value MUST be interpreted as "no".

description: The description of the form template. If this attribute is not present, its value MUST be interpreted as an empty string (1).

name: An identifier for the form template in the form of a Uniform Resource Name (URN). This attribute MUST be specified.

productVersion: The version of the form designer with which the form template was created. This attribute's value SHOULD be "12.0.0.0" or "11.0.0.0".<1>

publishUrl: This attribute MUST NOT be present.

requireFullTrust: Specifies whether the form template requires an elevated **form security level**. If set to "yes", this attribute MUST override the form security level specified by the **trustLevel** attribute, and the form MUST be loaded with an elevated form security level. An elevated form security level allows cross-domain data connections and full access to business objects. If this attribute is not present, its value MUST be interpreted as "no".

solutionFormatVersion: The version number of the form template format. The attribute's value SHOULD be "2.0.0.0" or "1.100.0.0" or "1.0.0.0".<2>

solutionVersion: The version number of the form template. The value of this attribute MUST be greater than any version of the form template already published. Values are compared numerically, left-to-right.

trustLevel: The form security level. If this attribute is present, its value MUST be "domain". If this attribute is not present, its value MUST be interpreted as "domain". This is the default security context and can be overridden by the **requireFullTrust** attribute.

trustSetting: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="xDocumentClass">
  <xsd:complexType>
    <xsd:all>
      <xsd:element ref="xsf:package" minOccurs="1"/>
      <xsd:element ref="xsf:permissions" minOccurs="0"/>
      <xsd:element ref="xsf:views" minOccurs="1"/>
      <xsd:element ref="xsf:hwsWorkflow" minOccurs="0"/>
      <xsd:element ref="xsf:externalViews" minOccurs="0"/>
      <xsd:element ref="xsf:scripts" minOccurs="0"/>
      <xsd:element ref="xsf:schemaErrorMessages" minOccurs="0"/>
      <xsd:element ref="xsf:documentSchemas" minOccurs="0"/>
      <xsd:element ref="xsf:applicationParameters" minOccurs="0"/>
      <xsd:element ref="xsf:featureRestrictions" minOccurs="0"/>
      <xsd:element ref="xsf:fileNew" minOccurs="0"/>
      <xsd:element ref="xsf:customValidation" minOccurs="0"/>
      <xsd:element ref="xsf:domEventHandlers" minOccurs="0"/>
      <xsd:element ref="xsf:importParameters" minOccurs="0"/>
      <xsd:element ref="xsf:listProperties" minOccurs="0"/>
      <xsd:element ref="xsf:taskpane" minOccurs="0"/>
      <xsd:element ref="xsf:documentSignatures" minOccurs="0"/>
      <xsd:element ref="xsf:dataObjects" minOccurs="0"/>
      <xsd:element ref="xsf:dataAdapters" minOccurs="0"/>
      <xsd:element ref="xsf:query" minOccurs="0"/>
      <xsd:element ref="xsf:submit" minOccurs="0"/>
      <xsd:element ref="xsf:save" minOccurs="0"/>
      <xsd:element ref="xsf:roles" minOccurs="0"/>
      <xsd:element ref="xsf:onLoad" minOccurs="0"/>
      <xsd:element ref="xsf:documentVersionUpgrade" minOccurs="0"/>
      <xsd:element ref="xsf:extensions" minOccurs="0"/>
      <xsd:element ref="xsf:ruleSets" minOccurs="0"/>
      <xsd:element ref="xsf:calculations" minOccurs="0"/>
    </xsd:all>
    <xsd:attribute name="name" type="xsd:string" use="optional"/>
    <xsd:attribute name="author" type="xsd:string" use="optional"/>
    <xsd:attribute name="description" use="optional">
```

```
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="255"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="solutionVersion" type="xsf:xdSolutionVersion" use="optional"/>
      <xsd:attribute name="productVersion" type="xsd:string" use="optional"/>
      <xsd:attribute name="solutionFormatVersion" type="xsf:xdSolutionVersion" use="required"/>
      <xsd:attribute name="dataFormSolution" type="xsf:xdYesNo" use="optional"/>
      <xsd:attribute name="requireFullTrust" type="xsf:xdYesNo" use="optional"/>
      <xsd:attribute name="trustLevel" type="xsf:xdTrustLevel" use="optional"/>
      <xsd:attribute name="trustSetting" type="xsf:xdManualAuto" use="optional"/>
      <xsd:attribute name="publishUrl" type="xsd:string" use="optional"/>
    </xsd:complexType>
    <xsd:key name="view_name_key">
      <xsd:selector xpath="./xsf:views/xsf:view"/>
      <xsd:field xpath="@name"/>
    </xsd:key>
    <xsd:key name="externalView_name_key">
      <xsd:selector xpath="./xsf:externalViews/xsf:externalView"/>
      <xsd:field xpath="@name"/>
    </xsd:key>
    <xsd:key name="view_or_externalView_name_key">
      <xsd:selector xpath="./xsf:views/xsf:view | ./xsf:externalViews/xsf:externalView"/>
      <xsd:field xpath="@name"/>
    </xsd:key>
    <xsd:key name="ruleset_name_key">
      <xsd:selector xpath="./xsf:ruleSets/xsf:ruleSet"/>
      <xsd:field xpath="@name"/>
    </xsd:key>
    <xsd:key name="dataObject_name_key">
      <xsd:selector xpath="./xsf:dataObjects/xsf:dataObject"/>
      <xsd:field xpath="@name"/>
    </xsd:key>
    <xsd:unique name="adapter name unique">
      <xsd:selector xpath="./xsf:dataObjects/xsf:dataObject/xsf:query/* | ./xsf:query/* |
  ./xsf:dataAdapters/* | ./xsf:submit/xsf:webServiceAdapter | ./xsf:submit/xsf:davAdapter |
  ./xsf:submit/xsf:emailAdapter | ./xsf:submit/xsf:submitToHostAdapter"/>
      <xsd:field xpath="@name"/>
    </xsd:unique>
    <xsd:key name="adapter name key">
      <xsd:selector xpath="./xsf:dataAdapters/*"/>
      <xsd:field xpath="@name"/>
    </xsd:key>
    <xsd:unique name="view external name unique">
      <xsd:selector xpath="./xsf:views/xsf:view | ./xsf:externalViews/xsf:externalView"/>
      <xsd:field xpath="@name"/>
    </xsd:unique>
  </xsd:element>
```

2.2.21 schemaErrorMessages

The **schemaErrorMessages** element specifies custom error messages that are displayed for XML schema data type errors in the form file.

Parent Elements
xDocumentClass

Child Elements
override

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="schemaErrorMessages">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:override" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.22 override

The **override** element specifies the **XML node** for which the XML schema data type error message MUST be overridden.

Parent Elements
schemaErrorMessages

Child Elements
errorMessage

Attributes:

match: This attribute MUST be an XPath expression that evaluates to a single XML node.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="override">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:errorMessage"/>
    </xsd:sequence>
    <xsd:attribute name="match" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.23 applicationParameters

The **applicationParameters** element MUST be ignored.

Parent Elements
xDocumentClass

Child Elements
solutionProperties

Attributes:

application: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="applicationParameters">
  <xsd:complexType>
    <xsd:all>
      <xsd:element ref="xsf:solutionProperties" minOccurs="0"/>
    </xsd:all>
    <xsd:attribute name="application" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="InfoPath Design Mode"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
</xsd:element>
```

2.2.24 solutionProperties

The **solutionProperties** element MUST be ignored.

Parent Elements
applicationParameters

Attributes:

allowCustomization: This attribute MUST be ignored.

automaticallyCreateNodes: This attribute MUST be ignored.

fullyEditableNamespace: This attribute MUST be ignored.

lastOpenView: This attribute MUST be ignored.

lastVersionNeedingTransform: This attribute MUST be ignored.

publishSaveUrl: This attribute MUST be ignored.

scriptLanguage: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="solutionProperties">
  <xsd:complexType>
    <xsd:attribute name="allowCustomization" type="xsf:xdYesNo" use="optional"/>
    <xsd:attribute name="lastOpenView" type="xsf:xdFileName" use="optional"/>
    <xsd:attribute name="scriptLanguage" type="xsf:xdScriptLanguage" use="optional"/>
    <xsd:attribute name="automaticallyCreateNodes" type="xsf:xdYesNo" use="optional"/>
    <xsd:attribute name="lastVersionNeedingTransform" type="xsf:xdSolutionVersion"
use="optional"/>
    <xsd:attribute name="fullyEditableNamespace" type="xsd:anyURI" use="optional"/>
    <xsd:attribute name="publishSaveUrl" type="xsd:string" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.25 featureRestrictions

The **featureRestrictions** element specifies one or more of the following features that are restricted when editing the form:

- Auto-recovering the form file. MUST be ignored.
- Exporting the form file. MUST be ignored.
- Printing the form (1).
- Saving the form file.
- Sending the form file as an e-mail attachment. MUST be ignored.

Parent Elements
xDocumentClass

Child Elements
autoRecovery
exportToExcel
exportToWeb
print
save
sendMail

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="featureRestrictions">
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="save" minOccurs="0">
        <xsd:complexType>
```

```
            <xsd:attribute name="ui" type="xsf:xdEnabledDisabled" use="required"/>
          </xsd:complexType>
        </xsd:element>
        <xsd:element ref="xsf:exportToWeb" minOccurs="0"/>
        <xsd:element ref="xsf:exportToExcel" minOccurs="0"/>
        <xsd:element ref="xsf:print" minOccurs="0"/>
        <xsd:element ref="xsf:sendMail" minOccurs="0"/>
        <xsd:element ref="xsf:autoRecovery" minOccurs="0"/>
      </xsd:all>
    </xsd:complexType>
  </xsd:element>
```

2.2.26 save (1)

The **save** element specifies whether the user interface elements of the form and keyboard shortcuts for saving the form file MUST be disabled. Restricting saving the form file through this element MUST NOT disable saving the form file through the use of form code.

Parent Elements
featureRestrictions

Attributes:

ui: Specifies whether the save feature is restricted via the menus, toolbars, or keyboard shortcuts of the form (1).

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="save" minOccurs="0">
  <xsd:complexType>
    <xsd:attribute name="ui" type="xsf:xdEnabledDisabled" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.27 exportToWeb

The **exportToWeb** element MUST be ignored.

Parent Elements
featureRestrictions

Attributes:

ui: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="exportToWeb">
  <xsd:complexType>
    <xsd:attribute name="ui" type="xsf:xdEnabledDisabled" use="required"/>
```

```
  </xsd:complexType>
</xsd:element>
```

2.2.28 exportToExcel

The **exportToExcel** element MUST be ignored.

Parent Elements
featureRestrictions

Attributes:

ui: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="exportToExcel">
  <xsd:complexType>
    <xsd:attribute name="ui" type="xsf:xdEnabledDisabled" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.29 print

The **print** element specifies whether the user interface elements of the form and keyboard shortcuts for printing the form are disabled. Restricting printing through this element MUST NOT disable printing the form through use of form code.

Parent Elements
featureRestrictions

Attributes:

ui: Specifies whether the print feature is restricted via the menus, toolbars, or keyboard shortcuts of the form (1).

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="print">
  <xsd:complexType>
    <xsd:attribute name="ui" type="xsf:xdEnabledDisabled" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.30 sendMail

The **sendMail** element MUST be ignored.

Parent Elements
featureRestrictions

Attributes:

ui: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="sendMail">
  <xsd:complexType>
    <xsd:attribute name="ui" type="xsf:xdEnabledDisabled" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.31 autoRecovery

The **autoRecovery** element MUST be ignored.

Parent Elements
featureRestrictions

Attributes:

feature: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="autoRecovery">
  <xsd:complexType>
    <xsd:attribute name="feature" type="xsf:xdEnabledDisabled" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.32 query (1)

The **query** element specifies the main data connection **data adapter** that queries a **data source** for data to populate the **main data source**. The main data connection MUST specify a data connection to a database or Web service.

If the form template is designed based on a main data connection, as specified by the **dataFormSolution** attribute of the **xDocumentClass** element, as defined in section 2.2.20, the main data source XML schema is derived from the XML schema provided by the main data connection. If the form template is not designed based on a main data connection, the main data source XML schema is derived from manual modifications or an external XML schema document.

Parent Elements
xDocumentClass

Child Elements
adoAdapter
queryAction
sharepointListAdapter
webServiceAdapter
xmlFileAdapter

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="query">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element ref="xsf:queryAction"/>
      <xsd:element ref="xsf:adoAdapter"/>
      <xsd:element ref="xsf:webServiceAdapter"/>
      <xsd:element ref="xsf:xmlFileAdapter"/>
      <xsd:element ref="xsf:sharepointListAdapter"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

2.2.33 scripts

The **scripts** element MUST NOT be present.

Parent Elements
xDocumentClass

Child Elements
script

Attributes:

enforceScriptTimeout: This attribute MUST NOT be present.

language: This attribute MUST NOT be present.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="scripts">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:script" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="language" type="xsf:xdScriptLanguage" use="required"/>
    <xsd:attribute name="enforceScriptTimeout" type="xsf:xdYesNo" use="optional"
default="yes"/>
  </xsd:complexType>
</xsd:element>
```

2.2.34 script

The **script** element MUST NOT be present.

Parent Elements
scripts

Attributes:

src: This attribute MUST NOT be present.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="script">
  <xsd:complexType>
    <xsd:attribute name="src" type="xsf:xdFileName" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.35 dataObjects

The **dataObjects** element specifies all **secondary data connections** that query a **secondary data source**. Secondary data sources are used only to provide data to populate the form file or to be used for form functionality. The form file MUST NOT be submitted to a secondary data source.

If the form template contains a secondary data connection that queries a secondary data source, the secondary data source XML schema document MUST be contained in the form template (.xsn) file.

Parent Elements
xDocumentClass

Child Elements
dataObject

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="dataObjects">
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element ref="xsf:dataObject"/>
    </xsd:choice>
  </xsd:complexType>
  <xsd:unique name="dataObjects_name_unique">
    <xsd:selector xpath="./xsf:dataObject"/>
    <xsd:field xpath="@name"/>
  </xsd:unique>
</xsd:element>
```

2.2.36 dataObject

The **dataObject** element specifies the properties and behavior of a secondary data connection that queries a secondary data source for data.

Parent Elements
dataObjects

Child Elements
query

Attributes:

initOnLoad: Specifies whether the secondary data source MUST be queried when the form is loaded. If this attribute is not present, its value MUST be interpreted as "no".

name: The name for the secondary data source. The specified name MUST be unique among all secondary data sources in the form template.

schema: The name of the XML schema document associated with the secondary data source.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="dataObject">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element name="query">
        <xsd:complexType>
          <xsd:choice>
            <xsd:element ref="xsf:adoAdapter"/>
            <xsd:element ref="xsf:webServiceAdapter"/>
            <xsd:element ref="xsf:xmlFileAdapter"/>
            <xsd:element ref="xsf:sharepointListAdapter"/>
          </xsd:choice>
        </xsd:complexType>
      </xsd:element>
    </xsd:choice>
    <xsd:attribute name="name" type="xsf:xdTitle" use="required"/>
    <xsd:attribute name="schema" type="xsd:string" use="optional"/>
```

```
      <xsd:attribute name="initOnLoad" type="xsf:xdYesNo" use="optional"/>
    </xsd:complexType>
  </xsd:element>
```

2.2.37 query (2)

The **query** element specifies a data adapter that queries a secondary data source.

Parent Elements
dataObject

Child Elements
adoAdapter
sharepointListAdapter
webServiceAdapter
xmlFileAdapter

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="query">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element ref="xsf:adoAdapter"/>
      <xsd:element ref="xsf:webServiceAdapter"/>
      <xsd:element ref="xsf:xmlFileAdapter"/>
      <xsd:element ref="xsf:sharepointListAdapter"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

2.2.38 adoAdapter

The **adoAdapter** element specifies the properties of a data adapter that MUST be created to query data from a database. The **ActiveX Data Objects (ADO)** data adapter MUST NOT support submitting the form file and MUST NOT support querying an Access data source.

Parent Elements
dataAdapters
query

Attributes:

commandText: The **SQL statement** that is used for querying or submitting data to a database.

connectionString: The ADO **connection string** that is used to connect to a database. The specified value MUST NOT specify a data connection to a data source file with the extensions ".mdb", ".mde", or ".accdb".

name: The name of the data adapter. The specified name MUST be unique for all data adapters within the form template. If this attribute is not present, its value MUST be interpreted as an empty string.

queryAllowed: Specifies whether the data adapter is allowed to query the database for data. If this attribute is not present, its value MUST be interpreted as "yes".

submitAllowed: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="adoAdapter">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsf:xdTitle" use="optional"/>
    <xsd:attribute name="connectionString" type="xsd:string" use="required"/>
    <xsd:attribute name="commandText" type="xsd:string" use="required"/>
    <xsd:attribute name="queryAllowed" type="xsf:xdYesNo" use="optional"/>
    <xsd:attribute name="submitAllowed" type="xsf:xdYesNo" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.39 webServiceAdapter

The **webServiceAdapter** element specifies the properties of a data adapter that MUST be created to query and submit data to a Web service.

When a form is submitted to a Web service, a **SOAP message** containing data from the form file is sent to the Web service. The SOAP message is generated from an **XML** template file, as specified by section 2.6, which is populated by data extracted from the form file.

Parent Elements
dataAdapters
query
submit

Child Elements
operation

Attributes:

name: The name of the data adapter. The specified name MUST be unique for all data adapters within the form template. If this attribute is not present, its value MUST be interpreted as an empty string.

queryAllowed: Specifies whether the data adapter is allowed to query the Web service for data. If this attribute is not present, its value MUST be interpreted as "yes".

submitAllowed: Specifies whether the data adapter is allowed to submit data to the Web service. If this attribute is not present, its value MUST be interpreted as "yes".

useDataSet: Specifies whether the data adapter supports the ADO.Net **DataSet** type. The ADO.Net **DataSet** is used if the Web service queries data from and submits data to an internal database. If this attribute is not present, its value MUST be interpreted as "no".

wsdlUrl: The URL of the Web service. If this attribute is not present, its value MUST be interpreted as an empty string.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="webServiceAdapter">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element ref="xsf:operation"/>
    </xsd:choice>
    <xsd:attribute name="name" type="xsf:xdTitle" use="optional"/>
    <xsd:attribute name="wsdlUrl" type="xsd:string" use="required"/>
    <xsd:attribute name="queryAllowed" type="xsf:xdYesNo" use="optional"/>
    <xsd:attribute name="submitAllowed" type="xsf:xdYesNo" use="optional"/>
    <xsd:attribute name="useDataSet" type="xsf:xdYesNo" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.40 hwsAdapter

The **hwsAdapter** element MUST NOT be present.

Parent Elements
dataAdapters

Child Elements
hwsOperation

Attributes:

name: This attribute MUST NOT be present.

queryAllowed: This attribute MUST NOT be present.

submitAllowed: This attribute MUST NOT be present.

wsdlUrl: This attribute MUST NOT be present.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="hwsAdapter">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element ref="xsf:hwsOperation"/>
```

```
      </xsd:choice>
      <xsd:attribute name="name" type="xsf:xdTitle" use="required"/>
      <xsd:attribute name="wsdlUrl" type="xsd:string" use="required"/>
      <xsd:attribute name="queryAllowed" type="xsf:xdYesNo" use="optional"/>
      <xsd:attribute name="submitAllowed" type="xsf:xdYesNo" use="optional"/>
    </xsd:complexType>
  </xsd:element>
```

2.2.41 operation

The **operation** element specifies the Web service operation method that MUST be used by the Web service data adapter for querying and submitting data.

Parent Elements
webServiceAdapter

Child Elements
input

Attributes:

name: The name of the Web service method.

serviceUrl: The URL of the Web service to which the request is sent.

soapAction: The **SOAP action** of the Web service that is used for the operation. The specified value MUST match the value specified by the **SOAPAction HTTP header field**, as specified in [SOAP1.2/2], in the SOAP request message sent to the Web service.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="operation">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element ref="xsf:input" minOccurs="0"/>
    </xsd:choice>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="soapAction" type="xsd:string" use="required"/>
    <xsd:attribute name="serviceUrl" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.42 hwsOperation

The **hwsOperation** element MUST NOT be present.

Parent Elements
hwsAdapter

Child Elements
input

Attributes:

serviceUrl: This attribute MUST NOT be present.

type: This attribute MUST NOT be present.

typeID: This attribute MUST NOT be present.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="hwsOperation">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element ref="xsf:input"/>
    </xsd:choice>
    <xsd:attribute name="type" type="xsd:string" use="required"/>
    <xsd:attribute name="typeID" type="xsd:string" use="required"/>
    <xsd:attribute name="serviceUrl" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.43 input

The **input** element specifies a SOAP message that is submitted to a Web service.

Parent Elements
hwsOperation
operation

Child Elements
partFragment

Attributes:

source: This attribute specifies the name of the file containing the XML template from which the SOAP message is created. The specified file MUST exist in the form template.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="input">
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element ref="xsf:partFragment"/>
    </xsd:choice>
```

```
    <xsd:attribute name="source" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.44 partFragment

The **partFragment** element specifies substitution information for a section of a SOAP message submitted to a Web service. If this element is present, the specified part of the SOAP message MUST be substituted with the specified data from the form file.

Parent Elements
input

Attributes:

dataObject: This attribute MUST be ignored.

filter: An XPath expression that MUST evaluate to an XML subtree in the form file. This attribute MUST be present when substituting a part of the SOAP message with a subset of the form file. If this attribute is not present, its value MUST be interpreted as an empty string.

match: This attribute specifies an XPath expression that identifies the elements and attributes inside the SOAP message to be replaced.

replaceWith: This attribute specifies an XPath expression that identifies the values in the form file that will replace a part of the SOAP message. If the **filter** attribute is present, an XML subtree MUST replace a part of the SOAP message. If the **filter** attribute is not present, an XML node MUST replace a part of the SOAP message.

sendAsString: This attribute specifies whether the substituted part of the SOAP message is submitted as a **string**. If this attribute is not present, its value MUST be interpreted as "no".

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="partFragment">
  <xsd:complexType>
    <xsd:attribute name="match" type="xsd:string" use="required"/>
    <xsd:attribute name="replaceWith" type="xsd:string" use="required"/>
    <xsd:attribute name="sendAsString" type="xsf:xdYesNo" use="optional"/>
    <xsd:attribute name="dataObject" type="xsd:string" use="optional"/>
    <xsd:attribute name="filter" type="xsd:string" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.45 xmlFileAdapter

The **xmlFileAdapter** element specifies the properties of a data adapter that MUST be created to query an XML file for data. The XML file can be located either within the form template (.xsn) file or at an external location.

Parent Elements
dataAdapters
query

Attributes:

fileUrl: Either the URL of an XML file that is not contained in the form template or the name of an XML file that is contained in the form template.

name: The name of the data adapter. The specified name MUST be unique for all data adapters within the form template. If this attribute is not present, its value MUST be interpreted as an empty string.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="xmlFileAdapter">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsf:xdTitle" use="optional"/>
    <xsd:attribute name="fileUrl" type="xsd:anyURI" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.46 sharepointListAdapter

The **sharepointListAdapter** element specifies the properties of a data adapter that MUST be created to query a protocol server list. The protocol server list MUST be used as a secondary data source, and the protocol server list data adapter MUST NOT support submitting the form file.

Parent Elements
dataAdapters
query

Child Elements
field

Attributes:

infopathGroup: The name of the parent **XML element** under which all query data is saved in the form file. The data adapter MUST save each returned query data item as a child of the specified element.

name: The name of the data adapter. The specified name MUST be unique for all data adapters within the form template.

queryAllowed: Specifies whether the data adapter is allowed to query the protocol server list (1) for data. If this attribute is not present, its value MUST be interpreted as "yes".

sharepointGuid: The **GUID** of the protocol server list.

siteUrl: The URL of the parent protocol server **site (2)**.

submitAllowed: Specifies whether the data adapter is allowed to submit data to the protocol server list (1). This attribute MUST NOT be set to "yes". If this attribute is not present, its value MUST be interpreted as "no".

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="sharepointListAdapter">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="field" minOccurs="0" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:attribute name="sharepointName" type="xsd:string" use="required"/>
          <xsd:attribute name="infopathName" type="xsd:string" use="required"/>
          <xsd:attribute name="isLookup" type="xsf:xdYesNo" use="optional"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsf:xdTitle" use="required"/>
    <xsd:attribute name="siteUrl" type="xsd:string" use="required"/>
    <xsd:attribute name="sharepointGuid" type="xsd:string" use="required"/>
    <xsd:attribute name="infopathGroup" type="xsd:string" use="required"/>
    <xsd:attribute name="queryAllowed" type="xsf:xdYesNo" use="optional"/>
    <xsd:attribute name="submitAllowed" type="xsf:xdYesNo" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.47 field (1)

The **field** element specifies mapping information for a protocol server list field that is used by the protocol server list data adapter to query a protocol server list. Each protocol server list field returned from a query MUST be specified by an instance of this element.

Parent Elements
sharepointListAdapter

Attributes:

infopathName: The name of the field in the form view that corresponds to the protocol server list field name, as specified by the value of the **sharepointName** attribute.

isLookup: Specifies whether the field is considered a **lookup field**. If this attribute is not present, its value MUST be interpreted as "no".

sharepointName: The protocol server list field name that corresponds to the name of the field in the form view specified by the value of the **infopathName** attribute.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="field" minOccurs="0" maxOccurs="unbounded">
  <xsd:complexType>
    <xsd:attribute name="sharepointName" type="xsd:string" use="required"/>
    <xsd:attribute name="infopathName" type="xsd:string" use="required"/>
    <xsd:attribute name="isLookup" type="xsf:xdYesNo" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.48 davAdapter

The **davAdapter** element specifies the properties of a data adapter that MUST be created to submit a form file to a **WebDAV server**. The **Web Distributed Authoring and Versioning Protocol (WebDAV)** data adapter MUST NOT support querying a WebDAV server.

Parent Elements
dataAdapters
submit

Child Elements
fileName
folderURL

Attributes:

name: The name of the data adapter. The specified name MUST be unique for all data adapters within the form template.

overwriteAllowed: Specifies whether the data adapter can overwrite an existing file. If this attribute is not present, its value MUST be interpreted as "no".

queryAllowed: Specifies whether the data adapter is allowed to query the WebDAV server for data. This attribute MUST be set to "no".

submitAllowed: Specifies whether the data adapter is allowed to submit data to the WebDAV server. This attribute MUST be set to "yes".

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="davAdapter">
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="folderURL">
        <xsd:complexType>
          <xsd:attribute name="value" type="xsd:string" use="required"/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="fileName">
        <xsd:complexType>
          <xsd:attribute name="value" type="xsd:string" use="required"/>
          <xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="optional"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:all>
    <xsd:attribute name="name" type="xsf:xdTitle" use="required"/>
    <xsd:attribute name="overwriteAllowed" type="xsf:xdYesNo" use="optional"/>
    <xsd:attribute name="queryAllowed" type="xsf:xdYesNo" use="optional"/>
    <xsd:attribute name="submitAllowed" type="xsf:xdYesNo" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.49 folderURL

The **folderURL** element specifies the URL of a WebDAV server or protocol server to which the form file MUST be submitted.

Parent Elements
davAdapter

Attributes:

value: The server URL. The specified value MUST begin with "http://" or "https://".

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="folderURL">
  <xsd:complexType>
    <xsd:attribute name="value" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.50 fileName

The **fileName** element specifies a file name that is used when the form file is submitted using the WebDAV data adapter. The form file MUST be submitted as a file with the specified name. If the specified file name does not include a file extension, the extension ".xml" MUST be appended.

Parent Elements
davAdapter

Attributes:

value: A file name or an XPath expression that evaluates to a file name. If it is set as an XPath expression, the **valueType** attribute MUST be set to "expression". If it is set to a file name, the **valueType** attribute MUST be set to "literal".

valueType: Specifies how the value specified by the **value** attribute is interpreted. If set to "expression", the value attribute value MUST be evaluated as an **XPath** expression. If set to "literal", the value attribute value MUST be evaluated as a literal string.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="fileName">
  <xsd:complexType>
    <xsd:attribute name="value" type="xsd:string" use="required"/>
    <xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.51 emailAdapter

The **emailAdapter** element specifies the information needed to submit the form as an attachment to an e-mail with a specified set of recipients, subject, and an introduction. The e-mail MUST have a set of recipients specified by the **to**, **cc**, or **bcc** elements.

Parent Elements
dataAdapters
submit

Child Elements
attachmentFileName
bcc
cc
intro
subject
to

Attributes:

name: This attribute specifies the name of the data adapter. The specified name MUST be unique for all data adapters within the form template

queryAllowed: This attribute specifies whether the data adapter is allowed to query for data. This attribute MUST be set to "no".

submitAllowed: This attribute specifies whether the data adapter is allowed to submit data via e-mail. This attribute MUST be set to "yes".

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="emailAdapter">
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="to" minOccurs="0">
        <xsd:complexType>
          <xsd:attribute name="value" type="xsd:string" use="required"/>
          <xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="optional"/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="cc" minOccurs="0">
        <xsd:complexType>
          <xsd:attribute name="value" type="xsd:string" use="required"/>
          <xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="optional"/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="bcc" minOccurs="0">
        <xsd:complexType>
          <xsd:attribute name="value" type="xsd:string" use="required"/>
          <xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="optional"/>
```

```
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="subject" minOccurs="0">
          <xsd:complexType>
            <xsd:attribute name="value" type="xsd:string" use="required"/>
            <xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="optional"/>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="intro" minOccurs="0">
          <xsd:complexType>
            <xsd:attribute name="value" type="xsd:string" use="required"/>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="attachmentFileName" minOccurs="0">
          <xsd:complexType>
            <xsd:attribute name="value" type="xsd:string" use="required"/>
            <xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="optional"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:all>
      <xsd:attribute name="name" type="xsf:xdTitle" use="required"/>
      <xsd:attribute name="queryAllowed" type="xsf:xdYesNo" use="optional"/>
      <xsd:attribute name="submitAllowed" type="xsf:xdYesNo" use="optional"/>
    </xsd:complexType>
  </xsd:element>
```

2.2.52 to

The **to** element specifies the main recipient information for the e-mail message that is generated when the form file is submitted using the e-mail data adapter.

Parent Elements
emailAdapter

Attributes:

value: This attribute specifies either a literal string of recipient addresses or an XPath expression that evaluates to a string of recipient addresses. The recipient addresses MUST be delimited by a ";". If the specified value is a literal string, the **valueType** attribute MUST be unspecified or set to "literal". Otherwise, it MUST be set to "expression".

valueType: This attribute specifies whether the **value** attribute value MUST be interpreted as a literal string or an XPath expression. If set to "expression", the **value** attribute value MUST be evaluated as an XPath expression. If set to "literal", the **value** attribute value MUST be evaluated as a literal string.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
  <xsd:element name="to" minOccurs="0">
    <xsd:complexType>
      <xsd:attribute name="value" type="xsd:string" use="required"/>
      <xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="optional"/>
    </xsd:complexType>
  </xsd:element>
```

2.2.53 cc

The **cc** element specifies the **carbon copy (cc) recipients** for the e-mail message that is generated when the form file is submitted using the e-mail data adapter.

Parent Elements
emailAdapter

Attributes:

value: This attribute specifies either a literal string of recipient addresses or an XPath expression that evaluates to a string of recipient addresses. The recipient addresses MUST be delimited by a ";". If the specified value is a literal string, the **valueType** attribute MUST be unspecified or set to "literal". Otherwise, it MUST be set to "expression".

valueType: This attribute specifies whether the **value** attribute value MUST be interpreted as a literal string or an XPath expression. If set to "expression", the **value** attribute value MUST be evaluated as an XPath expression. If set to "literal", the **value** attribute value MUST be evaluated as a literal string.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="cc" minOccurs="0">
  <xsd:complexType>
    <xsd:attribute name="value" type="xsd:string" use="required"/>
    <xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.54 bcc

The **bcc** element specifies the **blind carbon copy (bcc) recipients** for the e-mail message that is generated when the form file is submitted using the e-mail data adapter.

Parent Elements
emailAdapter

Attributes:

value: This attribute specifies either a literal string of recipient addresses or an XPath expression that evaluates to a string of recipient addresses. The recipient addresses MUST be delimited by a ";". If the specified value is a literal string, the **valueType** attribute MUST be unspecified or set to "literal". Otherwise, it MUST be set to "expression".

valueType: This attribute specifies whether the **value** attribute value MUST be interpreted as a literal string or an XPath expression. If set to "expression", the **value** attribute value MUST be evaluated as an XPath expression. If set to "literal", the **value** attribute value MUST be evaluated as a literal string.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="bcc" minOccurs="0">
```

```
  <xsd:complexType>
    <xsd:attribute name="value" type="xsd:string" use="required"/>
    <xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.55 subject

The **subject** element specifies the subject text for the e-mail message that is generated when the form file is submitted using the e-mail data adapter. The specified subject text MUST NOT exceed 255 characters.

Parent Elements
emailAdapter

Attributes:

value: This attribute specifies either a literal string or an XPath expression that evaluates to a **string**. If the specified value is a literal string, the **valueType** attribute MUST be set to "literal". Otherwise, it MUST be set to "expression".

valueType: This attribute specifies how the **value** attribute value MUST be interpreted. If set to "expression", the **value** attribute value MUST be evaluated as an XPath expression. If set to "literal", the **value** attribute value MUST be evaluated as a literal string.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="subject" minOccurs="0">
  <xsd:complexType>
    <xsd:attribute name="value" type="xsd:string" use="required"/>
    <xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.56 intro

The **intro** element specifies the body text for the e-mail message that is generated when the form file is submitted using the e-mail data adapter.

Parent Elements
emailAdapter

Attributes:

value: This attribute specifies the body text.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="intro" minOccurs="0">
```

```
    <xsd:complexType>
      <xsd:attribute name="value" type="xsd:string" use="required"/>
    </xsd:complexType>
  </xsd:element>
```

2.2.57 attachmentFileName

The **attachmentFileName** element specifies the file name of a file attachment to be included with the e-mail message when the form file is submitted using the e-mail data adapter.

Parent Elements
emailAdapter

Attributes:

value: This attribute specifies the value of the **attachmentFileName** element.

valueType: This attribute specifies whether the **value** attribute is interpreted as an XPath expression or a string literal. If set to "expression", the **value** attribute value MUST be evaluated as an XPath expression. If set to "literal", the **value** attribute value MUST be evaluated as a literal string.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="attachmentFileName" minOccurs="0">
  <xsd:complexType>
    <xsd:attribute name="value" type="xsd:string" use="required"/>
    <xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.58 submitToHostAdapter

The **submitToHostAdapter** element specifies the properties of a data adapter that MUST be created to submit data to a hosting environment.

Parent Elements
dataAdapters
submit

Attributes:

name: This attribute specifies the name of the data adapter. The specified name MUST be unique for all data adapters within the form template.

queryAllowed: This attribute MUST be ignored.

submitAllowed: This attribute specifies whether the data adapter is allowed to submit data to the host. This attribute MUST be interpreted as "yes".

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="submitToHostAdapter">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsf:xdTitle" use="required"/>
    <xsd:attribute name="queryAllowed" type="xsf:xdYesNo" use="optional"/>
    <xsd:attribute name="submitAllowed" type="xsf:xdYesNo" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.59 dataAdapters

The **dataAdapters** element specifies the secondary data connection data adapters that submit the form file to a data source.

Parent Elements
xDocumentClass

Child Elements
adoAdapter
davAdapter
emailAdapter
hwsAdapter
sharepointListAdapter
submitToHostAdapter
webServiceAdapter
xmlFileAdapter

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="dataAdapters">
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element ref="xsf:adoAdapter"/>
      <xsd:element ref="xsf:webServiceAdapter"/>
      <xsd:element ref="xsf:xmlFileAdapter"/>
      <xsd:element ref="xsf:sharepointListAdapter"/>
      <xsd:element ref="xsf:davAdapter"/>
      <xsd:element ref="xsf:emailAdapter"/>
      <xsd:element ref="xsf:submitToHostAdapter"/>
      <xsd:element ref="xsf:hwsAdapter"/>
    </xsd:choice>
  </xsd:complexType>
```

```
</xsd:element>
```

2.2.60 documentSchemas

The **documentSchemas** element specifies the XML schemas for the form template. This element contains references to one or more XML schemas that are used to form an authoritative XML schema to which the form file MUST fully conform, as specified by [XMLSCHEMA1].

The **root element** of the authoritative XML schema is defined in the XML schema identified by the **rootSchema** attribute of the **documentSchema** element. Additional XML schemas are included by using the XML schema's **import** or **include** constructs as follows:

- If the XML schema is included using the XML schema **import** construct, as specified by [XMLSCHEMA1], a **documentSchema** element MUST exist for that imported XML schema.
- If the XML schema is included using the XML schema **include** construct, as specified by [XMLSCHEMA1], a **documentSchema** element MUST NOT exist for the included XML schema.

Parent Elements
xDocumentClass

Child Elements
documentSchema

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="documentSchemas">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:documentSchema" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.61 documentSchema

The **documentSchema** element specifies an XML schema for the form template. The specified XML schema MUST be defined by an XML schema document in the form template (.xsn) file.

Parent Elements
documentSchemas

Attributes:

location: This attribute specifies the XML schema location as either one of the following:

- The name of the XML schema document.
- Both the XML schema namespace and the name of the XML schema document, separated by a space.

The specified name of the XML schema document MUST match the name of the corresponding file in the form template (.xsn) file. If the XML schema namespace is specified, it MUST match the namespace specified by the **value** attribute of the corresponding **property** element where the **name** attribute is "namespace". All XML schema documents in the form template MUST use different namespaces.

rootSchema: This attribute specifies whether an XML schema is the top-level XML schema for form files associated with this form template. There MUST be exactly one **documentSchema** element with a **rootSchema** value of "yes" in a form template. If this attribute is not present, its value MUST be interpreted as "no".

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="documentSchema">
  <xsd:complexType>
    <xsd:attribute name="location" type="xsd:string" use="required"/>
    <xsd:attribute name="rootSchema" type="xsf:xdYesNo"/>
  </xsd:complexType>
</xsd:element>
```

2.2.62 customValidation

The **customValidation** element specifies a rule-based custom validation that is enforced in addition to the XML schema validation.

Parent Elements
xDocumentClass

Child Elements
errorCondition

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="customValidation">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:errorCondition" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.63 errorCondition

The **errorCondition** element specifies a custom validation for a set of XML nodes in the form file.

Parent Elements
customValidation

Child Elements
errorMessage

Attributes:

expression: This attribute specifies an XPath expression to validate the XML nodes returned by evaluating the **match** attribute value. If the **expressionContext** attribute is present, it specifies a context for the XPath expression.

expressionContext: This attribute specifies the XML node that provides the root context for the **expression** attribute value. If this attribute is not present, its value MUST be interpreted as an empty string.

match: This attribute specifies an XPath expression that evaluates to the XML nodes for which the custom validation applies.

showErrorOn: This attribute specifies the XML nodes on which the error MUST be displayed when the form is filled out. If this attribute is not present, its value MUST be interpreted as ".".

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="errorCondition">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:errorMessage"/>
    </xsd:sequence>
    <xsd:attribute name="match" type="xsd:string" use="required"/>
    <xsd:attribute name="expression" type="xsd:string" use="required"/>
    <xsd:attribute name="expressionContext" type="xsd:string" use="optional"/>
    <xsd:attribute name="showErrorOn" type="xsd:string" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.64 errorMessage (1)

The **errorMessage** element specifies the error message that MUST be returned if the value of the specified XML node is considered invalid according to the value specified by the **expression** attribute of the **errorCondition** element.

Parent Elements
errorCondition

Parent Elements
override

Attributes:

shortMessage: This attribute specifies the short error message that MUST be returned in case of invalid data.

type: This attribute specifies the modality of the error message. If this attribute is not present, its value MUST be interpreted as "modal". If the value is "modal", the **errorMessage** element MUST be ignored. If the value is "modeless", this **errorMessage** element MUST NOT be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="errorMessage">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsf:xdErrorMessage">
        <xsd:attribute name="type" use="optional">
          <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
              <xsd:enumeration value="modal"/>
              <xsd:enumeration value="modeless"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="shortMessage" use="required">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:maxLength value="127"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

2.2.65 domEventHandlers

The **domEventHandlers** element specifies script-based event handlers that are triggered by changes to the form file.

Parent Elements
xDocumentClass

Child Elements
domEventHandler

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="domEventHandlers">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:domEventHandler" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:unique name="domEventHandler_handlerObject_unique">
    <xsd:selector xpath="."/>
    <xsd:field xpath="@handlerObject"/>
  </xsd:unique>
</xsd:element>
```

2.2.66 domEventHandler

The **domEventHandler** element specifies a handler for events triggered when the specified XML nodes change. The **child rule** set is run when this handler is called. The various types of child rules specify supported actions to take on the form file.

Parent Elements
domEventHandlers

Child Elements
ruleSetAction

Attributes:

dataObject: This attribute specifies the name of the secondary data source that MUST be used in the event handler. The specified name MUST match the value specified by the corresponding name attribute of the **dataObject** element. If this attribute is not present, its value MUST be interpreted as an empty string.

handlerObject: This attribute specifies the name of the event handler. The specified name MUST be unique within the form template.

match: This attribute specifies the XML nodes for which the event handler is declared. The value MUST be a valid XPath expression that identifies one or more XML nodes.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="domEventHandler">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:ruleSetAction" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="dataObject" type="xsd:string" use="optional"/>
    <xsd:attribute name="match" type="xsd:string" use="required"/>
    <xsd:attribute name="handlerObject" type="xsd:string" use="optional"/>
  </xsd:complexType>
  <xsd:keyref name="domEventHandler_ruleSetAction" refer="xsf:ruleset_name_key">
    <xsd:selector xpath="./xsf:ruleSetAction"/>
```

```
      <xsd:field xpath="@ruleSet"/>
    </xsd:keyref>
  </xsd:element>
```

2.2.67 importParameters

The **importParameters** element specifies whether the form file can merge another form file. The merge feature MUST be enabled through this element.

Parent Elements
xDocumentClass

Child Elements
importSource

Attributes:

enabled: This attribute specifies whether form merging is enabled.

useScriptHandler: This attribute MUST be set to "no".

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="importParameters">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:importSource" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="enabled" type="xsf:xdYesNo" use="required"/>
    <xsd:attribute name="useScriptHandler" type="xsf:xdYesNo" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.68 importSource

The **importSource** element specifies the parameters that are used when merging a source form file of a specific XML schema into a destination form file. If this element is not present, the default XSLT file MUST be used for all XSLTs during the merge operation.

Parent Elements
importParameters

Attributes:

authoringOfTransform: This attribute specifies whether the XSLTs are automatically authored. If this attribute is not present, its value MUST be interpreted as "manual".

name: This attribute specifies the name of the source form.

schema: This attribute specifies the name of the XML schema document that is used to validate the source form file during the merge operation. The specified file MUST exist in the form template.

transform: This attribute specifies the name of the XSLT file that is used during the merge operation. The specified name MUST match the name of the corresponding file in the form template.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="importSource">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="schema" type="xsf:xdFileName" use="required"/>
    <xsd:attribute name="transform" type="xsf:xdFileName" use="required"/>
    <xsd:attribute name="authoringOfTransform" type="xsf:xdManualAuto" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.69 listProperties

The **listProperties** element specifies a collection of fields that are promoted from the form file and made available to the default **list view** of a protocol server **form library** as properties on that protocol server form library.

Parent Elements
xDocumentClass

Child Elements
fields

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="listProperties">
  <xsd:complexType>
    <xsd:all>
      <xsd:element ref="xsf:fields"/>
    </xsd:all>
  </xsd:complexType>
</xsd:element>
```

2.2.70 fields

The **fields** element specifies a collection of fields that are promoted from the form file and made available to the default list view of a protocol server form library.

Parent Elements
listProperties

Child Elements
field

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="fields">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:field" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.71 field (2)

The **field** element specifies a field that is promoted from the form file and made available to the default list view of a protocol server form library. Each promoted field MUST be specified by an instance of this element.

Parent Elements
fields

Attributes:

aggregation: This attribute specifies how a single XML node or a collection of XML nodes returned from evaluating the **node** attribute value is aggregated to obtain a value for the field. If this attribute is not present, its value MUST be interpreted as an empty string. If this attribute is present, it MUST be set to one of the following values:

- **average:** The aggregate MUST be determined by calculating the average of all XML node values. This value MUST NOT be specified if any of the XML nodes is not of an XML schema number data type.
- **count:** The aggregate value MUST be determined by calculating the number of XML nodes.
- **first:** The aggregate value MUST be determined by returning the first XML node value in the collection.
- **last:** The aggregate value MUST be determined by returning the last XML node value in the collection.
- **max:** The aggregate value MUST be determined by calculating the maximum XML node value in the collection. This value MUST NOT be specified if any of the XML nodes is not of an XML schema number data type.

- **merge:** The aggregate value MUST be determined by concatenating all XML node values in the collection separated by newline characters, or Unicode #x000D and #x000A.
- **min:** The aggregate value MUST be determined by calculating the minimum XML node value in the collection. This value MUST NOT be specified if any of the XML nodes is not of an XML schema number data type.
- **plaintext:** The aggregate value MUST be determined by returning the raw, unformatted text value of the XML node. This value MUST NOT be specified if the **node** attribute value evaluates to a collection of more than one XML node. This value MUST NOT be specified if the XML nodes is not of an XML schema rich text data type.
- **sum:** The aggregate value MUST be determined by calculating the sum of all XML node values in the collection. This value MUST NOT be specified if any of the XML nodes is not of an XML schema number data type.

columnName: This attribute specifies the internal name of the corresponding **column** in the **Structured Query Language (SQL)** database underlying the protocol server list view. The specified value MUST match the **columnName** attribute for the **fieldExtension** element.

maxLength: This attribute specifies the maximum length of the field in the number of bytes. If this attribute is not present, its value MUST be determined by the field type and protocol server site settings.

name: This attribute specifies the **friendly name** of the field used on the protocol server list view.

node: This attribute specifies the XPath expression that evaluates to the corresponding field in the form file.

required: This attribute specifies whether this field accepts null values. If this attribute is not present, its value MUST be interpreted as "no".

type: This attribute specifies the standard XML schema data type of the field.

viewable: This attribute specifies whether this field is added to the default protocol server list view. If this attribute is not present, its value MUST be interpreted as "yes".

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="field">
  <xsd:complexType>
    <xsd:attribute name="type" type="xsd:NMTOKEN" use="required"/>
    <xsd:attribute name="name" type="xsf:xdTitle" use="required"/>
    <xsd:attribute name="columnName" type="xsf:xdTitle" use="required"/>
    <xsd:attribute name="required" type="xsf:xdYesNo" use="optional"/>
    <xsd:attribute name="viewable" type="xsf:xdYesNo" use="optional"/>
    <xsd:attribute name="node" type="xsd:string" use="required"/>
    <xsd:attribute name="maxLength" type="xsd:byte"/>
    <xsd:attribute name="aggregation" use="optional">
      <xsd:simpleType>
        <xsd:restriction base="xsd:NMTOKEN">
          <xsd:enumeration value="sum"/>
          <xsd:enumeration value="count"/>
          <xsd:enumeration value="average"/>
          <xsd:enumeration value="min"/>
          <xsd:enumeration value="max"/>
          <xsd:enumeration value="first"/>
          <xsd:enumeration value="last"/>
          <xsd:enumeration value="merge"/>
          <xsd:enumeration value="plaintext"/>
        </xsd:restriction>
      </xsd:simpleType>
```

```
      </xsd:attribute>
    </xsd:complexType>
  </xsd:element>
```

2.2.72 submit

The **submit** element specifies the necessary information for configuring the submit operation for the form. This includes information regarding the method to use, user interface elements that call the operation, and related actions to perform after the submit operation is complete.

A submit element can be associated with a data adapter, rule set, script handler, or HTTP handler as follows:

- **Data adapter:** The form file is submitted using a data adapter if the target for the submit operation is a data source with an associated data adapter, as specified by the **davAdapter**, **emailAdapter**, **submitToHostAdapter**, and **webServiceAdapter** elements. The **data adapter** child element MUST have the **submitAllowed** attribute set to "yes".
- **Rule Set:** The form file is submitted by an associated collection of rules that execute associated actions as specified by the **rulesetAction** element.
- **Script Handler:** The form file is submitted by associated form code as specified by the **useScriptHandler** element.
- **HTTP Handler:** The form file is submitted using the **HTTP method** specified by the **useHttpHandler** element (section 2.2.76).

Prior to the **submit** operation being performed, the form file MUST fully conform to the XML schema specified in section 2.3 and to any custom validation defined in the form template specified by the **customValidation** element.

Parent Elements
xDocumentClass

Child Elements
davAdapter
emailAdapter
errorMessage
ruleSetAction
submitAction
submitToHostAdapter
successMessage
useHttpHandler
useQueryAdapter

Child Elements
useScriptHandler
webServiceAdapter

Attributes:

caption: This attribute specifies the name of the submit button. A corresponding button MUST appear on the form view toolbar when the form is loaded. If this attribute is not present, its value MUST be interpreted as "Submit".

disableMenuItem: This attribute specifies whether the button for submitting the form file is available. If this attribute's value is set to "yes", the button MUST be removed from the toolbar. If this attribute is not present, its value MUST be interpreted as "no".

onAfterSubmit: This attribute specifies an action that MUST be taken upon successful submission of the form file. If this attribute is not present, its value MUST be interpreted as "keepOpen". The specified value MUST be one of the following:

- **close:** The form closes.
- **keepOpen:** The form state does not change.
- **openNew:** The form resets itself to the state of a new form.

showSignatureReminder: This attribute MUST be ignored.

showStatusDialog: This attribute specifies that a dialog box MUST be shown after the form file is submitted if this attribute's value is "yes". If this attribute is not present, its value MUST be interpreted as "yes".

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="submit">
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="submitAction" minOccurs="0">
        <xsd:complexType>
          <xsd:attribute name="adapter" type="xsf:xdTitle" use="required"/>
        </xsd:complexType>
        <xsd:keyref name="submitAdapter_name_keyref" refer="xsf:adapter_name_key">
          <xsd:selector xpath="."/>
          <xsd:field xpath="@adapter"/>
        </xsd:keyref>
      </xsd:element>
      <xsd:element ref="xsf:useHttpHandler" minOccurs="0"/>
      <xsd:element ref="xsf:useScriptHandler" minOccurs="0"/>
      <xsd:element ref="xsf:ruleSetAction" minOccurs="0"/>
      <xsd:element ref="xsf:useQueryAdapter" minOccurs="0"/>
      <xsd:element ref="xsf:webServiceAdapter" minOccurs="0"/>
      <xsd:element ref="xsf:davAdapter" minOccurs="0"/>
      <xsd:element ref="xsf:emailAdapter" minOccurs="0"/>
      <xsd:element ref="xsf:submitToHostAdapter" minOccurs="0"/>
      <xsd:element name="successMessage" type="xsd:string" minOccurs="0"/>
      <xsd:element name="errorMessage" type="xsd:string" minOccurs="0"/>
    </xsd:all>
    <xsd:attribute name="caption" type="xsd:string" use="optional"/>
    <xsd:attribute name="onAfterSubmit" use="optional">
      <xsd:simpleType>
        <xsd:restriction base="xsd:NMTOKEN">
          <xsd:enumeration value="close"/>
          <xsd:enumeration value="keepOpen"/>
```

```
            <xsd:enumeration value="openNew"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="showStatusDialog" type="xsf:xdYesNo" use="optional"/>
      <xsd:attribute name="showSignatureReminder" type="xsf:xdYesNo" use="optional"/>
      <xsd:attribute name="disableMenuItem" type="xsf:xdYesNo" use="optional"/>
    </xsd:complexType>
    <xsd:keyref name="submit_ruleSetAction" refer="xsf:ruleset_name_key">
      <xsd:selector xpath="./xsf:ruleSetAction"/>
      <xsd:field xpath="@ruleSet"/>
    </xsd:keyref>
  </xsd:element>
```

2.2.73 submitAction (1)

The **submitAction** element MUST NOT be present.

Parent Elements
submit

Attributes:

adapter: This attribute MUST NOT be present.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="submitAction" minOccurs="0">
  <xsd:complexType>
    <xsd:attribute name="adapter" type="xsf:xdTitle" use="required"/>
  </xsd:complexType>
  <xsd:keyref name="submitAdapter_name_keyref" refer="xsf:adapter_name_key">
    <xsd:selector xpath="."/>
    <xsd:field xpath="@adapter"/>
  </xsd:keyref>
</xsd:element>
```

2.2.74 successMessage

The **successMessage** element specifies the **string** used to notify the user that the form was submitted successfully.

Parent Elements
submit

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="successMessage" type="xsd:string" minOccurs="0"/>
```

2.2.75 errorMessage (2)

The **errorMessage** element specifies the **string** used to notify the user that the form was not submitted successfully.

Parent Elements
submit

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="errorMessage" type="xsd:string" minOccurs="0"/>
```

2.2.76 useHttpHandler

The **useHttpHandler** element specifies that the form MUST be submitted to the specified URL using the specified HTTP method.

Parent Elements
submit

Attributes:

href: This attribute specifies the URL to which the form (1) is submitted.

method: This attribute specifies the HTTP method that is used to submit the form (1). This value MUST be "POST".

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="useHttpHandler">
  <xsd:complexType>
    <xsd:attribute name="method" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:NMTOKEN">
          <xsd:enumeration value="POST"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="href" type="xsd:anyURI" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.77 useScriptHandler

The **useScriptHandler** element specifies that the corresponding action MUST be performed using form code.

Parent Elements
documentVersionUpgrade
save
submit

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="useScriptHandler"/>
```

2.2.78 useQueryAdapter

The **useQueryAdapter** element MUST NOT be present.

Parent Elements
submit

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="useQueryAdapter"/>
```

2.2.79 onLoad

The **onLoad** element specifies a set of rules that is called when the form is loaded.

Parent Elements
xDocumentClass

Child Elements
ruleSetAction

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="onLoad">
  <xsd:complexType>
```

```
      <xsd:sequence>
        <xsd:element ref="xsf:ruleSetAction" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:keyref name="load_ruleSetAction" refer="xsf:ruleset_name_key">
      <xsd:selector xpath="./xsf:ruleSetAction"/>
      <xsd:field xpath="@ruleSet"/>
    </xsd:keyref>
  </xsd:element>
```

2.2.80 save (2)

The **save** element MUST NOT be present.

Parent Elements
xDocumentClass

Child Elements
useScriptHandler

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="save">
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="1">
      <xsd:element ref="xsf:useScriptHandler"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

2.2.81 roles

The **roles** element MUST NOT be present.

Parent Elements
xDocumentClass

Child Elements
membership
role

Attributes:

default: This attribute MUST NOT be present.

hideStatusBarDisplay: This attribute MUST NOT be present.

initiator: This attribute MUST NOT be present.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="roles">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:role" minOccurs="1" maxOccurs="unbounded"/>
      <xsd:element ref="xsf:membership" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="default" type="xsd:string" use="required"/>
    <xsd:attribute name="initiator" type="xsd:string" use="optional"/>
    <xsd:attribute name="hideStatusBarDisplay" type="xsf:xdYesNo" use="optional"/>
  </xsd:complexType>
  <!-- role names must be unique -->
  <xsd:unique name="roles name unique">
    <xsd:selector xpath="./xsf:role"/>
    <xsd:field xpath="@name"/>
  </xsd:unique>
  <!-- fields must reference existing role -->
  <xsd:key name="role name key">
    <xsd:selector xpath="./xsf:role"/>
    <xsd:field xpath="@name"/>
  </xsd:key>
  <xsd:keyref name="role_default" refer="xsf:role_name_key">
    <xsd:selector xpath="."/>
    <xsd:field xpath="@default"/>
  </xsd:keyref>
  <xsd:keyref name="role_initiator" refer="xsf:role_name_key">
    <xsd:selector xpath="."/>
    <xsd:field xpath="@initiator"/>
  </xsd:keyref>
  <xsd:keyref name="role membership" refer="xsf:role name key">
    <xsd:selector xpath="./xsf:membership/*"/>
    <xsd:field xpath="@memberOf"/>
  </xsd:keyref>
</xsd:element>
```

2.2.82 role

The **role** element MUST NOT be present.

Parent Elements
roles

Attributes:

name: This attribute MUST NOT be present.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="role">
```

```
    <xsd:complexType>
      <xsd:attribute name="name" type="xsf:xdRoleName" use="required"/>
    </xsd:complexType>
  </xsd:element>
```

2.2.83 membership

The **membership** element MUST NOT be present.

Parent Elements
roles

Child Elements
getUserNameFromData
group
userName

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="membership">
  <xsd:complexType>
    <xsd:choice minOccurs="1" maxOccurs="unbounded">
      <xsd:element ref="xsf:getUserNameFromData"/>
      <xsd:element ref="xsf:userName"/>
      <xsd:element ref="xsf:group"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

2.2.84 getUserNameFromData

The **getUserNameFromData** element MUST NOT be present.

Parent Elements
membership

Attributes:

dataObject: This attribute MUST NOT be present.

memberOf: This attribute MUST NOT be present.

select: This attribute MUST NOT be present.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="getUserNameFromData">
  <xsd:complexType>
    <xsd:attribute name="dataObject" type="xsd:string" use="optional"/>
    <xsd:attribute name="select" type="xsd:string" use="required"/>
    <xsd:attribute name="memberOf" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.85 userName

The **userName** element MUST NOT be present.

Parent Elements
membership

Attributes:

memberOf: This attribute MUST NOT be present.

name: This attribute MUST NOT be present.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="userName">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="memberOf" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.86 group

The **group** element MUST NOT be present.

Parent Elements
membership

Attributes:

memberOf: This attribute MUST NOT be present.

name: This attribute MUST NOT be present.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="group">
  <xsd:complexType>
```

```
      <xsd:attribute name="name" type="xsd:string" use="required"/>
      <xsd:attribute name="memberOf" type="xsd:string" use="required"/>
    </xsd:complexType>
  </xsd:element>
```

2.2.87 hwsWorkflow

The **hwsWorkflow** element MUST NOT be present.

Parent Elements
xDocumentClass

Child Elements
allowedActions
allowedTasks
location

Attributes:

taskpaneVisible: This attribute MUST NOT be present.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="hwsWorkflow">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:location" minOccurs="1" maxOccurs="1"/>
      <xsd:element ref="xsf:allowedActions" minOccurs="1" maxOccurs="1"/>
      <xsd:element ref="xsf:allowedTasks" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="taskpaneVisible" type="xsf:xdYesNo"/>
  </xsd:complexType>
  <xsd:unique name="hws_actiontask_name">
    <xsd:selector xpath="./xsf:allowedActions/xsf:action|./xsf:allowedTasks/xsf:task"/>
    <xsd:field xpath="@name"/>
  </xsd:unique>
</xsd:element>
```

2.2.88 location

The **location** element MUST NOT be present.

Parent Elements
hwsWorkflow

Attributes:

url: This attribute MUST NOT be present.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="location">
  <xsd:complexType>
    <xsd:attribute name="url" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.89 allowedActions

The **allowedActions** element MUST NOT be present.

Parent Elements
hwsWorkflow

Child Elements
action

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="allowedActions">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:action" minOccurs="1" maxOccurs="20"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:unique name="hws_actionTypeID_unique">
    <xsd:selector xpath="./xsf:action"/>
    <xsd:field xpath="@actionTypeID"/>
  </xsd:unique>
</xsd:element>
```

2.2.90 action

The **action** element MUST NOT be present.

Parent Elements
allowedActions

Attributes:

actionTypeID: This attribute MUST NOT be present.

canInitiateWorkflow: This attribute MUST NOT be present.

caption: This attribute MUST NOT be present.

name: This attribute MUST NOT be present.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="action">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsf:xdHWSname" use="required"/>
    <xsd:attribute name="actionTypeID" type="xsd:string" use="required"/>
    <xsd:attribute name="canInitiateWorkflow" type="xsf:xdYesNo" use="required"/>
    <xsd:attribute name="caption" type="xsf:xdHWSCaption" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.91 allowedTasks

The **allowedTasks** element MUST NOT be present.

Parent Elements
hwsWorkflow

Child Elements
task

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="allowedTasks">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:task" minOccurs="1" maxOccurs="20"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:unique name="hws_taskID_unique">
    <xsd:selector xpath="./xsf:task"/>
    <xsd:field xpath="@taskTypeID"/>
  </xsd:unique>
</xsd:element>
```

2.2.92 task

The **task** element MUST NOT be present.

Parent Elements
allowedTasks

Attributes:

caption: This attribute MUST NOT be present.

name: This attribute MUST NOT be present.

taskTypeID: This attribute MUST NOT be present.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="task">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsf:xdHWSname" use="required"/>
    <xsd:attribute name="taskTypeID" type="xsd:string" use="required"/>
    <xsd:attribute name="caption" type="xsf:xdHWSCaption" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.93 fileNew

The **fileNew** element specifies the name and location of the XML template file that contains default values for a new form based on the form template.

Parent Elements
xDocumentClass

Child Elements
initialXmlDocument

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="fileNew">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:initialXmlDocument"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.94 initialXmlDocument

The **initialXmlDocument** element specifies a reference to the template.xml file used for the creation of a new form.

When the new form is created, its field values MUST be populated with existing default values specified by the template.xml file, as defined in section 2.7.

Parent Elements
fileNew

Child Elements
customCategory

Attributes:

caption: This attribute specifies the name of the form (1).

href: This attribute specifies the name of the template.xml file. The specified file name MUST match the name of the corresponding file in the form template.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="initialXmlDocument">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:customCategory" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="caption" type="xsf:xdTitle" use="required"/>
    <xsd:attribute name="href" type="xsf:xdFileName" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.95 customCategory

The **customCategory** element specifies the form template category, which is used to group together form templates.

Parent Elements
initialXmlDocument

Attributes:

name: This attribute specifies the name of the custom category.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="customCategory">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsf:xdTitle" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.96 package

The **package** element specifies the collection of all files in the form template.

Parent Elements
xDocumentClass

Child Elements
files

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="package">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:files"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.97 files

The **files** element specifies the collection of all files in the form template, as specified in section 2.1, and their properties.

Parent Elements
package

Child Elements
file

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="files">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:file" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
```

```
</xsd:element>
```

2.2.98 file

The **file** element specifies any file other than the form definition (.xsf) file contained within the form template (.xsn) file. Every such contained file MUST be specified by an instance of this element.

Parent Elements
files

Child Elements
fileProperties

Attributes:

name: This attribute specifies the name of the file that MUST exist in the form template.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="file">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:fileProperties" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsf:xdFileName" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.99 fileProperties

The **fileProperties** element specifies a collection of properties of a file in the form template

Parent Elements
file

Child Elements
property

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="fileProperties">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:property" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.100 property

The **property** element specifies a property of a file that is part of the form template. Each file property MUST be specified by an instance of this element.

Parent Elements
fileProperties

Attributes:

name: This attribute specifies the name of the file property and MUST be set to one of the following acceptable values. Each **property** element MUST have a unique name within the set of file properties specified by each **fileProperties** element. The corresponding acceptable values of the **value** attribute are listed for each possible value of the **name** attribute:

- **componentId:** The use of this value specifies that the corresponding **value** attribute value is an identifier that uniquely identifies the form view file. A form view file, as specified in section 2.4, MUST have a file property with this name. The **value** attribute value MUST be set to a positive integer that is unique among all form view file properties in the form template.
- **dataObject:** The use of this value specifies that the parent **file** element represents an XML schema document used to validate the secondary data source. The **value** attribute value MUST be set to the value of the name attribute of the **file** element of the file that is used as a secondary data source.
- **editability:** The use of this value specifies that the corresponding **value** attribute value is the degree to which the XML schema document is editable. The file specifying the XML schema of the form file MUST have a file property with this name. The **value** attribute value MUST be set to one of the following values:
  - **full:** This value specifies that the XML schema document MUST be fully editable.
  - partial – This value specifies that the XML schema document MUST be locked for editing. XML schema documents for secondary data sources MUST have an editability property set to this value.
- **fileType:** The use of this value specifies that the corresponding **value** attribute value specifies the file type. The **value** attribute value MUST be set to one of the following values:
  - **pdb:** This value specifies that the file MUST be a **symbol file**. A form template containing business objects MAY have any number of files of this type.
  - refAssembly – This value specifies that the file MUST be a form code **assembly** other than the main form code assembly. A form template containing business objects MAY have any number of files of this type.

- **rootAssembly:** This value specifies that the file refers to the main assembly of the form (1) code. A form template containing business objects MUST have exactly one file property with a "rootAssembly" value.
- **resource:** This value specifies that the file MUST be used as a secondary data source.
- **sampleData:** This value specifies that the file MUST be a file containing sample data for the form (1).

- **lang:** The use of this value specifies that the corresponding **value** attribute value MUST be the language of the form file. A form view file, as specified in section 2.4, MUST have a file property with this name. The **value** attribute value MUST be set to the **language code identifier (LCID)**, as specified in [MS-LCID], corresponding to the user **locale**.
- **namespace:** The use of this value specifies that the corresponding **value** attribute value MUST be the namespace of the XML schema document. An XML schema document MUST have a file property with this name. The **value** attribute value MUST be set to the namespace of the XML schema document.
- **rootElement:** The use of this value specifies that the corresponding **value** attribute value MUST be the root element of the XML schema document. An XML schema document MUST have a file property with this name. The **value** attribute value MUST be set to root element of the XML schema document.
- **useOnDemandAlgorithm:** The use of this value MUST be ignored.
- **xmlToEditName:** The use of this value specifies that the corresponding **value** attribute value MUST be equal to the value of the **value** attribute for the **componentId** file property. A form view file, as specified in section 2.4, MUST have a file property with this name. The **value** attribute value MUST be set to a positive integer.

type: This attribute MUST be set to "string".

value: This attribute specifies the value of the file property and MUST be set to one of the corresponding acceptable values specified in the **name** attribute.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="property">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="value" type="xsd:string" use="required"/>
    <xsd:attribute name="type" type="xsd:QName" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.101 permissions

The **permissions** element specifies the permissions required to instantiate **custom controls** in the form view.

Parent Elements
xDocumentClass

Child Elements
allowedControl

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="permissions">
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element ref="xsf:allowedControl"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

2.2.102 allowedControl

The **allowedControl** element MUST be ignored.

Parent Elements
permissions

Attributes:

cabFile: This attribute MUST be ignored.

clsid: This attribute MUST be ignored.

version: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="allowedControl">
  <xsd:complexType>
    <xsd:attribute name="cabFile" type="xsd:string" use="optional"/>
    <xsd:attribute name="clsid" type="xsd:string" use="required"/>
    <xsd:attribute name="version" type="xsd:string" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.103 externalViews

The **externalViews** element MUST be ignored.

Parent Elements
xDocumentClass

Child Elements
externalView

Attributes:

default: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="externalViews">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:externalView" minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="default" type="xsd:string"/>
  </xsd:complexType>
  <xsd:unique name="externalViews_name_unique">
    <xsd:selector xpath="./xsf:externalView"/>
    <xsd:field xpath="@default"/>
  </xsd:unique>
  <xsd:keyref name="external views printView" refer="xsf:externalView name key">
    <xsd:selector xpath="."/>
    <xsd:field xpath="@default"/>
  </xsd:keyref>
</xsd:element>
```

2.2.104 externalView

The **externalView** element MUST be ignored.

Parent Elements
externalViews

Child Elements
mainpane

Attributes:

designMode: This attribute MUST be ignored.

name: This attribute MUST be ignored.

target: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="externalView">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:mainpane"/>
    </xsd:sequence>
```

```
    <xsd:attribute name="target" type="xsd:string"/>
    <xsd:attribute name="name" type="xsf:xdViewName" use="required"/>
    <xsd:attribute name="designMode" type="xsf:xdDesignMode"/>
  </xsd:complexType>
</xsd:element>
```

2.2.105 attributeData

The **attributeData** element specifies the name and associated value of an attribute that MUST be inserted, or MUST be modified if it already exists, by the insert action of the **xCollection** or **xOptional** controls. This element MUST be a child element of the **chooseFragment** element, as specified in section 2.2.107.

Attributes:

attribute: This attribute specifies the name of the inserted attribute.

value: This attribute specifies the value of the inserted attribute.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="attributeData">
  <xsd:complexType>
    <xsd:attribute name="attribute" type="xsd:string" use="required"/>
    <xsd:attribute name="value" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.106 button (1)

The **button** element specifies a button on a control menu and its associated action that MUST be performed when the button is pressed.

Parent Elements
menu
menuArea
toolbar

Attributes:

action: This attribute specifies the control action that is performed. This attribute MUST be specified for buttons that manipulate the following controls:

- **xCollection:** A repeating section control, as specified in section 2.3.1.11, or a repeating table control, as specified in section 2.4.1.16.
- **xOptional:** An optional section control, as specified in section 2.4.1.18.
- **xFileAttachment:** A file attachment control, as specified in section 2.3.1.7.

The specified value MUST be one of the following:

- **xCollection::insert:** This action inserts a new section after all existing sections.

- **xCollection::insertBefore:** This action inserts a new section before the current section.
- **xCollection::insertAfter:** This action inserts a new section after the current section.
- **xCollection::remove:** This action deletes the current section.
- **xCollection::removeAll:** This action deletes all existing sections.
- **xOptional::insert:** This action inserts a new section after the current section.
- **xOptional::remove:** This action deletes the current section.
- **xFileAttachment::attach:** This action attaches a file to the form file.
- **xFileAttachment::open:** This action opens an attached file.
- **xFileAttachment::saveAs:** This action saves an attached file out of the form file.
- **xFileAttachment::remove:** This action removes an attached file from the form file.

caption: This attribute specifies the caption that MUST be displayed on the button. The value MUST be defined.

icon: This attribute MUST be ignored.

name: This attribute MUST be ignored.

showIf: This attribute MUST<3> be ignored. . The specified value MUST be one of the following:

- **always:** This value is deprecated.
- **enabled:** This value is deprecated.
- **immediate:** This button SHOULD show if the menu is shown.

tooltip: This attribute MUST be ignored.

xmlToEdit: This attribute specifies the name of the control for which the button is used. The specified value MUST match the value of the **name** attribute of the corresponding **xmlToEdit** element, as specified in section 2.2.124. This attribute MUST be defined for buttons used with collection controls.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="button">
  <xsd:complexType>
    <xsd:attribute name="caption" type="xsf:xdTitle"/>
    <xsd:attribute name="icon" type="xsd:string"/>
    <xsd:attribute name="tooltip" type="xsf:xdTitle"/>
    <xsd:attribute name="name" type="xsd:NMTOKEN"/>
    <xsd:attribute name="xmlToEdit" type="xsd:NMTOKEN"/>
    <xsd:attribute name="action">
      <xsd:simpleType>
        <xsd:restriction base="xsd:NMTOKEN">
          <xsd:enumeration value="xCollection::insert"/>
          <xsd:enumeration value="xCollection::insertBefore"/>
          <xsd:enumeration value="xCollection::insertAfter"/>
          <xsd:enumeration value="xCollection::remove"/>
          <xsd:enumeration value="xCollection::refreshFilter"/>
          <xsd:enumeration value="xCollection::removeAll"/>
          <xsd:enumeration value="xOptional::insert"/>
          <xsd:enumeration value="xOptional::remove"/>
          <xsd:enumeration value="xReplace::replace"/>
          <xsd:enumeration value="xFileAttachment::attach"/>
```

```
            <xsd:enumeration value="xFileAttachment::open"/>
            <xsd:enumeration value="xFileAttachment::saveAs"/>
            <xsd:enumeration value="xFileAttachment::remove"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="showIf">
        <xsd:simpleType>
          <xsd:restriction base="xsd:NMTOKEN">
            <xsd:enumeration value="always"/>
            <xsd:enumeration value="enabled"/>
            <xsd:enumeration value="immediate"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:complexType>
  </xsd:element>
```

2.2.107 chooseFragment

The **chooseFragment** element specifies an **XML fragment**. An XML fragment is an XML subtree that is intended to represent a unit of data. It is typically used for data insertion and replacement operations.

Parent Elements
fragmentToInsert

Attributes:

followingSiblings: This attribute specifies a relative XPath expression from the parent node. The parent node specifies the XML node prior to which the insertion of the XML fragment occurs. If the node is not found, the insertion action MUST be an append. If this attribute is not present, its value MUST be interpreted as an empty string.

innerFragment: This attribute specifies a relative XPath expression from the parent node to the smallest fragment to be inserted. If this attribute is not present, its value MUST be interpreted as an empty string (1).

parent: This attribute specifies a relative XPath expression from the container node that specifies the XML node under which the XML fragment MUST be inserted. If this attribute is not present, its value MUST be interpreted as a period (".").

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="chooseFragment">
  <xsd:complexType mixed="true">
    <xsd:sequence>
      <xsd:any minOccurs="0" maxOccurs="unbounded" processContents="skip"/>
    </xsd:sequence>
    <xsd:attribute name="parent" type="xsd:string"/>
    <xsd:attribute name="followingSiblings" type="xsd:string" use="optional"/>
    <xsd:attribute name="innerFragment" type="xsd:string" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.108 editWith

The **editWith** element specifies an instance of a control that edits data in the form file.

Parent Elements
xmlToEdit

Child Elements
fragmentToInsert
masterDetail

Attributes:

allowedFileTypes: This attribute MUST be ignored.

autoComplete: This attribute specifies whether auto-completion of fields is on. If this attribute is not present, its value MUST be interpreted as "no".

caption: This attribute specifies an identifier for alternate forms of XML data to be used in the control. If this attribute is not present, its value MUST be interpreted as an empty string (1).

component: This attribute specifies the name of the control that is referenced by an instance of the **xmlToEdit** element (section 2.2.124).

component: This attribute specifies the name of the control that is referenced by an instance of the **xmlToEdit** element. The specified value MUST be one of the following:

- **xCollection:** A repeating section control, as specified in section 2.4.1.15, or a repeating table control, as specified in section 2.4.1.16.
- **xOptional:** An optional section control, as specified in section 2.4.1.18.
- **xReplace:** This value MUST be ignored.
- **xTextList:** This value MUST be ignored.
- **xField:** Maps to one of the following controls:
  - Check Box control, as specified in section 2.4.1.6.
  - Date Picker control, as specified in section 2.4.1.8.
  - Drop-down list control, as specified in section 2.4.1.9.
  - List Box control, as specified in section 2.4.1.13.
  - Option Button control, as specified in section 2.4.1.14.
  - Rich Text Box control, as specified in section 2.4.1.17.
  - Text Box control, as specified in section 2.4.1.20.
- **xImage:** This value MUST be ignored.

- **xFileAttachment:** A file attachment control, as specified in section 2.4.1.11.

field: This attribute MUST be ignored.

filterDependency: This attribute MUST be ignored.

maxLength: This attribute MUST be ignored.

proofing: This attribute MUST be ignored.

removeAncestors: This attribute MUST be ignored.

type: This attribute MUST be ignored if the specified value is not "rich". If the specified value is "rich", the following MUST be true:

- If the **xmlToEditExtension** element (section 2.2.147.43) is present, both the **excludeEmbeddedImages** and **allowLinkedImages** attributes of the **xmlToEditExtension** element MUST be set to "yes".
- Otherwise, the **clientOnly** attribute of the **viewExtension** element (section 2.2.147.42) that is parent of the **xmlToEditExtension** element (section 2.2.147.43) MUST be set to "no".

useFilter: This attribute MUST be ignored.

widgetIcon: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="editWith">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:masterDetail" minOccurs="0" maxOccurs="1"/>
      <xsd:element ref="xsf:fragmentToInsert" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="component" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:NMTOKEN">
          <xsd:enumeration value="xCollection"/>
          <xsd:enumeration value="xOptional"/>
          <xsd:enumeration value="xReplace"/>
          <xsd:enumeration value="xTextList"/>
          <xsd:enumeration value="xField"/>
          <xsd:enumeration value="xImage"/>
          <xsd:enumeration value="xFileAttachment"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="caption" type="xsf:xdTitle" use="optional"/>
    <xsd:attribute name="autoComplete" type="xsf:xdYesNo" use="optional"/>
    <xsd:attribute name="proofing" type="xsf:xdYesNo" use="optional"/>
    <xsd:attribute name="type" use="optional">
      <xsd:simpleType>
        <xsd:restriction base="xsd:NMTOKEN">
          <xsd:enumeration value="plain"/>
          <xsd:enumeration value="formatted"/>
          <xsd:enumeration value="plainMultiline"/>
          <xsd:enumeration value="formattedMultiline"/>
          <xsd:enumeration value="rich"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="useFilter" use="optional">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="yes"/>
```

```
            <xsd:enumeration value="no"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="widgetIcon" use="optional">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="standard"/>
            <xsd:enumeration value="filter"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="filterDependency" type="xsd:string" use="optional"/>
      <xsd:attribute name="field" type="xsd:string" use="optional"/>
      <xsd:attribute name="removeAncestors" type="xsd:nonNegativeInteger" use="optional"/>
      <xsd:attribute name="maxLength" use="optional">
        <xsd:simpleType>
          <xsd:restriction base="xsd:integer">
            <xsd:minInclusive value="-1"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="allowedFileTypes" type="xsd:string" use="optional"/>
      <xsd:anyAttribute namespace="http://schemas.microsoft.com/office/infopath/2003"
processContents="skip"/>
    </xsd:complexType>
</xsd:element>
```

2.2.109 unboundControls

The **unboundControls** element specifies a collection of buttons in the form view.

Parent Elements
view

Child Elements
button

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="unboundControls">
  <xsd:complexType>
    <xsd:sequence>
      <!-- button -->
      <xsd:element name="button" minOccurs="0" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="xsf:ruleSetAction" minOccurs="0" maxOccurs="1"/>
          </xsd:sequence>
          <xsd:attribute name="name" use="required">
            <xsd:simpleType>
              <!-- type of name is non qualified name, but NCName also accepts '.' and '-',
so these characters are disabled by pattern restriction -->
              <xsd:restriction base="xsd:NCName">
```

```
                <xsd:pattern value="[^\.\^-]*"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:attribute>
        </xsd:complexType>
        <xsd:keyref name="button ruleSetAction" refer="xsf:ruleset name key">
          <xsd:selector xpath="./xsf:ruleSetAction"/>
          <xsd:field xpath="@ruleSet"/>
        </xsd:keyref>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.110 button (2)

The **button** element specifies a button that MAY have an associated event handler or **ruleSetAction** (section 2.2.132).

Parent Elements
unboundControls

Child Elements
ruleSetAction

Attributes:

name: This attribute specifies the event handler identifier of the button.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="button" minOccurs="0" maxOccurs="unbounded">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:ruleSetAction" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="name" use="required">
      <xsd:simpleType>
        <!-- type of name is non qualified name, but NCName also accepts '.' and '-', so
these characters are disabled by pattern restriction -->
        <xsd:restriction base="xsd:NCName">
          <xsd:pattern value="[^\.\^-]*"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
  <xsd:keyref name="button ruleSetAction" refer="xsf:ruleset name key">
    <xsd:selector xpath="./xsf:ruleSetAction"/>
    <xsd:field xpath="@ruleSet"/>
  </xsd:keyref>
</xsd:element>
```

2.2.111 editing

The **editing** element specifies additional information about controls used in the form view to edit the form file.

Parent Elements
view

Child Elements
xmlToEdit

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="editing">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:xmlToEdit" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.112 masterDetail

The **masterDetail** element MUST NOT be present.

Parent Elements
editWith

Attributes:

detailKey: This attribute MUST NOT be present.

master: This attribute MUST NOT be present.

masterKey: This attribute MUST NOT be present.

masterViewContext: This attribute MUST NOT be present.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="masterDetail">
  <xsd:complexType>
    <xsd:attribute name="master" type="xsd:string"/>
    <xsd:attribute name="masterViewContext" type="xsd:string"/>
    <xsd:attribute name="masterKey" type="xsd:string"/>
    <xsd:attribute name="detailKey" type="xsd:string"/>
```

```
  </xsd:complexType>
</xsd:element>
```

2.2.113 fragmentToInsert

The **fragmentToInsert** element specifies alternate versions of default XML data that is inserted into an associated control.

Parent Elements
editWith

Child Elements
chooseFragment

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="fragmentToInsert">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:chooseFragment" minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.114 mainpane

The **mainpane** element specifies the XSLT file, as specified in section 2.4, included in the form template and used to represent a form view.

Parent Elements
externalView
view

Attributes:

transform: This attribute specifies the name of the XSLT file that is used to transform the form. The specified value MUST match the name of the corresponding file in the form template.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="mainpane">
  <xsd:complexType>
```

```
      <xsd:attribute name="transform" type="xsf:xdFileName" use="required"/>
    </xsd:complexType>
  </xsd:element>
```

2.2.115 printSettings

The **printSettings** element MUST be ignored.

Parent Elements
mergedPrintView
view

Child Elements
footer
header

Attributes:

bottomMargin: This attribute MUST be ignored.

collate: This attribute MUST be ignored.

copies: This attribute MUST be ignored.

footer: This attribute MUST be ignored.

header: This attribute MUST be ignored.

leftMargin: This attribute MUST be ignored.

marginUnitsType: This attribute MUST be ignored.

orientation: This attribute MUST be ignored.

pageRangeEnd: This attribute MUST be ignored.

pageRangeStart: This attribute MUST be ignored.

paperSize: This attribute MUST be ignored.

paperSource: This attribute MUST be ignored.

printerName: This attribute MUST be ignored.

printerSpecificSettings: This attribute MUST be ignored.

rightMargin: This attribute MUST be ignored.

topMargin: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="printSettings">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:header" minOccurs="0" maxOccurs="1"/>
      <xsd:element ref="xsf:footer" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="orientation">
      <xsd:simpleType>
        <xsd:restriction base="xsd:NMTOKEN">
          <xsd:enumeration value="portrait"/>
          <xsd:enumeration value="landscape"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="header">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="255"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="footer">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="255"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="marginUnitsType">
      <xsd:simpleType>
        <xsd:restriction base="xsd:NMTOKEN">
          <xsd:enumeration value="in"/>
          <xsd:enumeration value="cm"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="rightMargin">
      <xsd:simpleType>
        <xsd:restriction base="xsd:float">
          <xsd:minInclusive value="0"/>
          <xsd:maxInclusive value="100"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="leftMargin">
      <xsd:simpleType>
        <xsd:restriction base="xsd:float">
          <xsd:minInclusive value="0"/>
          <xsd:maxInclusive value="100"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="topMargin">
      <xsd:simpleType>
        <xsd:restriction base="xsd:float">
          <xsd:minInclusive value="0"/>
          <xsd:maxInclusive value="100"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="bottomMargin">
      <xsd:simpleType>
        <xsd:restriction base="xsd:float">
          <xsd:minInclusive value="0"/>
          <xsd:maxInclusive value="100"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="printerName">
```

```
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:maxLength value="255"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="paperSize">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:maxLength value="255"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="paperSource">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:maxLength value="255"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="copies">
          <xsd:simpleType>
            <xsd:restriction base="xsd:integer">
              <xsd:minInclusive value="1"/>
              <xsd:maxInclusive value="9999"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="collate" type="xsf:xdYesNo"/>
        <xsd:attribute name="pageRangeStart">
          <xsd:simpleType>
            <xsd:restriction base="xsd:integer">
              <xsd:minInclusive value="1"/>
              <xsd:maxInclusive value="32000"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="pageRangeEnd">
          <xsd:simpleType>
            <xsd:restriction base="xsd:integer">
              <xsd:minInclusive value="1"/>
              <xsd:maxInclusive value="32000"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="printerSpecificSettings">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:maxLength value="255"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
      </xsd:complexType>
    </xsd:element>
```

2.2.116 header

The **header** element MUST be ignored.

Parent Elements
printSettings

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="header">
  <xsd:complexType mixed="true">
    <xsd:sequence>
      <xsd:any minOccurs="0" maxOccurs="unbounded" processContents="skip"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.117 footer

The **footer** element MUST be ignored.

Parent Elements
printSettings

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="footer">
  <xsd:complexType mixed="true">
    <xsd:sequence>
      <xsd:any minOccurs="0" maxOccurs="unbounded" processContents="skip"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.118 toolbar

The **toolbar** element MUST be ignored.

Parent Elements
view

Child Elements
button
menu

Attributes:

caption: This attribute MUST be ignored.

name: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="toolbar">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:group ref="xsf:UIItem" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsf:xdTitle" use="required"/>
    <xsd:attribute name="caption" type="xsf:xdTitle" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.119 menu

The **menu** element specifies a custom menu that MUST be applied to the form view.

Parent Elements
menu
menuArea
toolbar
view

Child Elements
button
menu

Attributes:

caption: This attribute specifies the caption for the menu.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="menu">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:group ref="xsf:UIItem" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="caption" type="xsf:xdTitle" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.120 menuArea

The **menuArea** element specifies a custom menu area that MUST be applied to the specified control in the form view.

Parent Elements
view

Child Elements
button
menu

Attributes:

name: This attribute MUST be set to "msoStructuralEditingContextMenu". This value refers to the context menu for structural controls. For more information, see section 2.4.1.15, section 2.4.1.16, and section 2.4.1.18. This attribute MUST NOT have the following values:

- **msoEditMenu:** Refers to the Edit menu.
- **msoFileMenu:** Refers to the File menu.
- **msoHelpMenu:** Refers to the Help menu.
- **msoInsertMenu:** Refers to the Insert menu.
- **msoFormatMenu:** Refers to the Format menu.
- **msoTableMenu:** Refers to the Table menu.
- **msoToolsMenu:** Refers to the Tools menu.
- **msoViewMenu:** Refers to the View menu.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="menuArea">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:group ref="xsf:UIItem" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:NMTOKEN">
          <xsd:enumeration value="msoFileMenu"/>
          <xsd:enumeration value="msoEditMenu"/>
          <xsd:enumeration value="msoInsertMenu"/>
          <xsd:enumeration value="msoViewMenu"/>
          <xsd:enumeration value="msoFormatMenu"/>
          <xsd:enumeration value="msoToolsMenu"/>
          <xsd:enumeration value="msoTableMenu"/>
          <xsd:enumeration value="msoHelpMenu"/>
          <xsd:enumeration value="msoStructuralEditingContextMenu"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
</xsd:element>
```

2.2.121 taskpane

The **taskpane** element MUST be ignored.

Parent Elements
xDocumentClass

Attributes:

caption: This attribute MUST be ignored.

href: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="taskpane">
  <xsd:complexType>
    <xsd:attribute name="caption" type="xsd:string" use="required"/>
    <xsd:attribute name="href" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.122 views

The **views** element specifies the collection of all form views in the form template. Form views are used to render and edit form files associated with the form template. Form views are represented by XSLT files, as specified in section 2.4, which define how to convert form files into HTML. A specific form view is used by default, and other form views can be displayed.

Each **browser-compatible form template** MUST contain at least one form view where both of the following are true:

- The **clientOnly** attribute of the **viewExtension** element (section 2.2.147.42) is set to "no".
- The **designMode** attribute of the **view** element (section 2.2.123) is set "normal".

Parent Elements
xDocumentClass

Child Elements
view

Attributes:

default: This attribute specifies the name of the form view that is used to render the form file. If this attribute is present, the specified value MUST match the value specified by the **name** attribute of a **view** element. If this attribute is not present, the first form view MUST be rendered.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="views">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:view" minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="default" type="xsd:string"/>
  </xsd:complexType>
  <xsd:unique name="views_name_unique">
    <xsd:selector xpath="./xsf:view"/>
    <xsd:field xpath="@name"/>
  </xsd:unique>
  <xsd:keyref name="view_printView" refer="xsf:view_or_externalView_name_key">
    <xsd:selector xpath="./xsf:view"/>
    <xsd:field xpath="@printView"/>
  </xsd:keyref>
  <xsd:keyref name="views_default" refer="xsf:view_name_key">
    <xsd:selector xpath="."/>
    <xsd:field xpath="@default"/>
  </xsd:keyref>
</xsd:element>
```

2.2.123 view

The **view** element specifies information about a form view, which is a specific visualization of a form file. It specifies what controls are used to represent the fields in the form and how they are rendered using HTML. A form view is represented by an XSLT file, as specified in section 2.4.

A form view MUST be generated as specified by this element.

Parent Elements
views

Child Elements
editing
mainpane
menu
menuArea
printSettings
toolbar
unboundControls

Attributes:

caption: This attribute specifies the display name for the form view. If this attribute is not present, its value MUST be interpreted as an empty string.

designMode: This attribute specifies the state of the form view. A form view with a **designMode** value of "protected" MUST be ignored. There MUST be at least one form view with a **designMode** value of "normal".

name: This attribute specifies the name of the form view.

printView: This attribute specifies the name of another form view to be used for printing the form view. The specified value MUST match the **name** attribute of the **view** element of one of the **view** elements. If this attribute is not present, the current form view is used for printing.

showMenuItem: This attribute specifies whether the menu item corresponding to the form view is displayed in the menu of form views. If this attribute is not present, its value MUST be interpreted as "no".

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="view">
  <xsd:complexType>
    <xsd:group ref="xsf:ViewContent" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:attribute name="caption" type="xsf:xdViewName"/>
    <xsd:attribute name="name" type="xsf:xdViewName" use="required"/>
    <xsd:attribute name="showMenuItem" type="xsf:xdYesNo" use="optional"/>
    <xsd:attribute name="printView" type="xsd:string"/>
    <xsd:attribute name="designMode" type="xsf:xdDesignMode"/>
  </xsd:complexType>
  <xsd:unique name="toolbar name unique">
    <xsd:selector xpath="./xsf:toolbar"/>
    <xsd:field xpath="@name"/>
  </xsd:unique>
  <xsd:unique name="menuArea_name_unique">
    <xsd:selector xpath="./xsf:menuArea"/>
    <xsd:field xpath="@name"/>
  </xsd:unique>
  <xsd:unique name="xmlToEdit_name_unique">
    <xsd:selector xpath="./xsf:editing/xsf:xmlToEdit"/>
    <xsd:field xpath="@name"/>
  </xsd:unique>
  <xsd:key name="xmlToEdit_name_key">
    <xsd:selector xpath="./xsf:editing/xsf:xmlToEdit"/>
    <xsd:field xpath="@name"/>
  </xsd:key>
  <xsd:keyref name="button_xmlToEdit_reference" refer="xsf:xmlToEdit_name_key">
    <xsd:selector xpath="./xsf:menuArea/xsf:button | ./xsf:menu/xsf:button |
./xsf:toolbar/xsf:button"/>
    <xsd:field xpath="@xmlToEdit"/>
  </xsd:keyref>
</xsd:element>
```

2.2.124 xmlToEdit

The **xmlToEdit** element specifies additional properties of a control that is used in the form view to edit the form file.

Parent Elements
editing

Child Elements
editWith

Attributes:

container: This attribute specifies an XPath expression that evaluates to the context in which the control MUST be selectable and enabled.

item: This attribute specifies an XPath expression that MUST evaluate to the XML nodes to be edited with the control. The specified XPath expression MUST be unique among all **xmlToEdit** elements in the form definition (.xsf) file.

name: This attribute specifies the name of the control.

viewContext: This attribute specifies the identifier of the corresponding control in the form view. If this attribute is not present, its value MUST be interpreted as an empty string.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="xmlToEdit">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:editWith" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:NMTOKEN" use="required"/>
    <xsd:attribute name="item" type="xsd:string" use="required"/>
    <xsd:attribute name="container" type="xsd:string"/>
    <xsd:attribute name="viewContext">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:pattern value="((\.|\#|[a-zA-Z0-9_])[a-zA-Z0-9_]*)(\s((\.|\#|[a-zA-Z0-9_])[a-
zA-Z0-9_]*))*"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
</xsd:element>
```

2.2.125 documentSignatures

The **documentSignatures** element specifies the digital signatures, as specified by [MS-IPFFX], that are used to sign the form file according to [XMLDSig].

Parent Elements
xDocumentClass

Child Elements
signedDataBlock

Attributes:

signatureLocation: This attribute MUST NOT be present.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="documentSignatures">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:signedDataBlock" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="signatureLocation" type="xsd:string" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.126 signedDataBlock

The **signedDataBlock** element specifies a set of XML nodes in the form file that MUST be signed by a digital signature.

Parent Elements
documentSignatures

Child Elements
message

Attributes:

data: This attribute specifies an XPath expression that MUST evaluate to a collection of XML nodes.

mode: This attribute specifies the relationship of the digital signature (1) and MUST be set to one of the following values:

- **countersign:** The digital signature signs all previous digital signatures.
- **cosign:** The digital signature is treated independently of all previous digital signatures.
- **single:** The signed data block MUST NOT be signed by more than one digital signature.

name: This attribute specifies the name of the signed data block.

signatureLocation: This attribute specifies an XPath expression that MUST evaluate to an XML node in the form file. The specified location MUST be used to store the digital signature (1).

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="signedDataBlock">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="message" type="xsf:xdSignedDataBlockMessage" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsf:xdSignedDataBlockName" use="required"/>
    <xsd:attribute name="data" type="xsd:string" use="required"/>
    <xsd:attribute name="signatureLocation" type="xsd:string" use="required"/>
    <xsd:attribute name="mode" type="xsf:xdSignatureRelationEnum" use="required"/>
  </xsd:complexType>
```

```
    <xsd:unique name="signedDataBlock_name_unique">
      <xsd:selector xpath="."/>
      <xsd:field xpath="@name"/>
    </xsd:unique>
  </xsd:element>
```

2.2.127 message

The **message** element specifies the confirmation message that MUST be displayed before a digital signature is applied to the form or section of the form.

Parent Elements
signedDataBlock

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="message" type="xsf:xdSignedDataBlockMessage" minOccurs="0"/>
```

2.2.128 documentVersionUpgrade

The **documentVersionUpgrade** element specifies the process by which forms created based on an older version of the form template are upgraded to the latest version of the form template.

Parent Elements
xDocumentClass

Child Elements
useScriptHandler
useTransform

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="documentVersionUpgrade">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element ref="xsf:useScriptHandler"/>
      <xsd:element ref="xsf:useTransform"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

2.2.129 useTransform

The **useTransform** element specifies the XSLT file, as defined in section 2.8, and the restrictions that MUST be used to upgrade the form.

Parent Elements
documentVersionUpgrade

Attributes:

maxVersionToUpgrade: This attribute specifies the inclusive value for the maximum form template version, specified by the **solutionVersion** attribute of the **xDocumentClass** element (section 2.2.20) that MUST be upgraded. If this attribute is not present, the maximum version boundary for upgrading MUST be ignored.

minVersionToUpgrade: This attribute specifies the inclusive value for the minimum form template version, specified by the **solutionVersion** attribute of the **xDocumentClass** element, that MUST be upgraded. If this attribute is not present, the minimum version boundary for upgrading is not checked.

transform: This attribute specifies the name of the XSLT file used to upgrade the form (1). The specified file MUST exist in the form template.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="useTransform">
  <xsd:complexType>
    <xsd:attribute name="transform" use="required">
      <xsd:simpleType>
        <xsd:union memberTypes="xsf:xdFileName xsf:xdEmptyString"/>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="minVersionToUpgrade" type="xsf:xdSolutionVersion" use="required"/>
    <xsd:attribute name="maxVersionToUpgrade" type="xsf:xdSolutionVersion"/>
  </xsd:complexType>
</xsd:element>
```

2.2.130 extensions

The **extensions** element specifies the enabled extensions in the form definition (.xsf) file. Each enabled extension MUST conform to the XML schema as defined in section 2.2.147.

Parent Elements
xDocumentClass

Child Elements
extension

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="extensions">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:extension" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.131 extension

The **extension** element specifies a container for an XML schema extension.

Parent Elements
extensions

Attributes:

name: This attribute specifies the name of this XML schema extension.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="extension">
  <xsd:complexType mixed="true">
    <xsd:sequence>
      <xsd:any minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:NMTOKEN" use="required"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.132 ruleSetAction

The **ruleSetAction** element specifies the rule set, as defined by the **ruleSet** element (section 2.2.143) that MUST be called by a form or form file event.

Parent Elements
button
domEventHandler
onLoad
submit

Attributes:

ruleSet: This attribute specifies the name of the rule set that is called. The specified value MUST match the value specified by the **name** attribute of the corresponding **ruleSet** element.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="ruleSetAction">
  <xsd:complexType>
    <xsd:attribute name="ruleSet" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.133 rule

The **rule** element specifies a rule, which is composed of the rule definition and the event by which the rule is called. The rule definition is defined by the **rule** elements and the **ruleSet** elements (section 2.2.143). The event is defined by the **button** (section 2.2.110), **domEventHandler** (section 2.2.66), **onLoad** (section 2.2.79), and **submit** (section 2.2.72) elements and their associated **ruleSetAction** element (section 2.2.132).

A rule (1) consists of the following:

- A set of one or more actions.
- A condition that determines whether the actions are executed.

If the associated condition of the rule evaluates positively with the **true** function specified in [XPATH] section 4.3, the actions associated with the rule are processed sequentially in the order in which they are listed in the **rule** element.

Rules are grouped together as a rule set, as specified by the **ruleSet** element, containing one or more rules. A rule set is bound to one of the following events with the **ruleSetAction** element:

- A form file change, such as a change in the value of an XML node.
- A form action, such as submitting the form file.
- An unbound control event, such as a button click event.

Each rule set is processed sequentially in the order in which they are listed within the **ruleSets** element (section 2.2.144).

Parent Elements
ruleSet

Child Elements
assignmentAction
closeDocumentAction
dialogBoxExpressionAction
dialogBoxMessageAction
exitRuleSet

Child Elements
openNewDocumentAction
queryAction
submitAction
switchViewAction

Attributes:

caption: This attribute specifies the name of the rule (1).

condition: This attribute specifies an XPath expression that MUST evaluate to either "true()" or "false()". If it evaluates to "true()", the associated actions MUST be executed. If this attribute is not present, its value MUST be interpreted as "true()".

isEnabled: This attribute specifies if the rule (1) MUST be enabled for the form (1). If this attribute is not present, its value MUST be interpreted as "yes".

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="rule">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element ref="xsf:dialogBoxMessageAction"/>
        <xsd:element ref="xsf:dialogBoxExpressionAction"/>
        <xsd:element ref="xsf:switchViewAction"/>
        <xsd:element ref="xsf:assignmentAction"/>
        <xsd:element ref="xsf:queryAction"/>
        <xsd:element name="submitAction">
          <xsd:complexType>
            <xsd:attribute name="adapter" type="xsf:xdTitle" use="required"/>
          </xsd:complexType>
        </xsd:element>
        <xsd:element ref="xsf:openNewDocumentAction"/>
        <xsd:element ref="xsf:closeDocumentAction"/>
      </xsd:choice>
      <xsd:element name="exitRuleSet" minOccurs="0">
        <xsd:complexType/>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="caption" type="xsd:string" use="required"/>
    <xsd:attribute name="condition" type="xsd:string" use="optional"/>
    <xsd:attribute name="isEnabled" type="xsf:xdYesNo" use="optional" default="yes"/>
  </xsd:complexType>
</xsd:element>
```

2.2.134 submitAction (2)

The **submitAction** element specifies the data adapter that MUST submit the form file when called by a form action.

Parent Elements
rule

Attributes:

adapter: This attribute specifies the name of the corresponding data adapter that is used to submit the form file. The specified name MUST match the name of an existing data adapter that allows submission of the form file.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="submitAction">
  <xsd:complexType>
    <xsd:attribute name="adapter" type="xsf:xdTitle" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.135 exitRuleSet

The **exitRuleSet** element specifies that rule processing MUST stop for the entire rule set.

Parent Elements
rule

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="exitRuleSet" minOccurs="0">
  <xsd:complexType/>
</xsd:element>
```

2.2.136 dialogBoxMessageAction

The **dialogBoxMessageAction** element MUST be ignored.

Parent Elements
rule

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="dialogBoxMessageAction">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="1024"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

2.2.137 dialogBoxExpressionAction

The **dialogBoxExpressionAction** element MUST be ignored.

Parent Elements
rule

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.2.138 switchViewAction

The **switchViewAction** element specifies the form view that MUST be shown when called by a form event.

Parent Elements
rule

Attributes:

view: This attribute specifies the name of the form view to be shown. The specified name MUST match an existing **name** attribute of the **view** element (section 2.2.123).

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="switchViewAction">
  <xsd:complexType>
    <xsd:attribute name="view" type="xsf:xdViewName" use="required"/>
  </xsd:complexType>
  <xsd:keyref name="switchViewAction_view_keyref" refer="xsf:view_name_key">
    <xsd:selector xpath="."/>
    <xsd:field xpath="@view"/>
  </xsd:keyref>
</xsd:element>
```

2.2.139 assignmentAction

The **assignmentAction** element specifies an action that MUST set the value of a field.

Parent Elements
rule

Attributes:

expression: This attribute specifies an XPath expression to populate the value of the **targetField** attribute.

targetField: This attribute specifies an XPath expression that MUST evaluate to the target XML node.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="assignmentAction">
  <xsd:complexType>
    <xsd:attribute name="targetField" type="xsd:string" use="required"/>
    <xsd:attribute name="expression" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.140 queryAction

The **queryAction** element specifies a data adapter that MUST query its data source when called by a form action.

Parent Elements
query
rule

Attributes:

adapter: This attribute specifies the name of the data adapter that MUST query its data source (2).

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="queryAction">
  <xsd:complexType>
    <xsd:attribute name="adapter" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.141 openNewDocumentAction

The **openNewDocumentAction** element MUST NOT be present.

Parent Elements
rule

Attributes:

solutionURI: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="openNewDocumentAction">
  <xsd:complexType>
    <xsd:attribute name="solutionURI" type="xsd:anyURI" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.142 closeDocumentAction

The **closeDocumentAction** element specifies an action to close the form.

Parent Elements
rule

Attributes:

promptToSaveChanges: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="closeDocumentAction">
  <xsd:complexType>
    <xsd:attribute name="promptToSaveChanges" type="xsf:xdYesNo" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.143 ruleSet

The **ruleSet** element specifies a set of one or more rules for the form.

Parent Elements
ruleSets

Child Elements
rule

Attributes:

name: This attribute specifies the name of the set of rules (1).

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="ruleSet">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:rule" minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
```

```
      <xsd:attribute name="name" type="xsd:string" use="required"/>
    </xsd:complexType>
  </xsd:element>
```

2.2.144 ruleSets

The **ruleSets** element specifies the rule sets for the form.

Parent Elements
xDocumentClass

Child Elements
ruleSet

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="ruleSets">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:ruleSet" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:unique name="ruleSets_name_unique">
    <xsd:selector xpath="./xsf:ruleSet"/>
    <xsd:field xpath="@name"/>
  </xsd:unique>
</xsd:element>
```

2.2.145 calculations

The **calculations** element specifies definitions for the calculations performed in the form and how blank values are handled.

Parent Elements
xDocumentClass

Child Elements
calculatedField

Attributes:

treatBlankValueAsZero: This attribute specifies whether an empty string is equivalent to the integer zero. If this attribute is not present, its value MUST be interpreted as "yes".

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="calculations">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:calculatedField" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="treatBlankValueAsZero" type="xsf:xdYesNo" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.146 calculatedField

The **calculatedField** element specifies an individual calculation, when the calculation is to be performed, and where the result is stored.

Parent Elements
calculations

Attributes:

expression: This attribute specifies the formula, as an XPath expression, to be evaluated. The result MUST be stored in the **target** attribute.

refresh: This attribute specifies when the expression MUST be evaluated. The value MUST be one of the following values:

- **onInit:** The value is evaluated when the node is initialized.
- **onChange:** The value is evaluated when a parameter of the expression changes.

target: This attribute specifies the XPath expression location where the result of evaluating the **expression** attribute MUST be stored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="calculatedField">
  <xsd:complexType>
    <xsd:attribute name="target" type="xsd:string" use="required"/>
    <xsd:attribute name="expression" type="xsd:string" use="required"/>
    <xsd:attribute name="refresh" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147 Form Definition File (XSF) Extension Specification

The extensions to the form definition (.xsf) file specify extensions to the properties and content of the form template. The extensions MUST conform to the XML schema defined by the elements in the following table. The XML schema for the form definition (.xsf) file extensions is used to validate the

elements, attributes, and types in the xsf2 namespace (http://schemas.microsoft.com/office/infopath/2006/solutionDefinition/extensions).

The following tables list, in alphabetical order, the types and elements used in the XML schema for the extensions to the form definition (.xsf) file.

Type	Specified in Section
compatibilityModesType	2.2.147.3
emailAttachmentType	2.2.147.2
formDescriptionType	2.2.147.5
formLocaleType	2.2.147.6
managedCodeType	2.2.147.7
serverCommandActionType	2.2.147.1
solutionType	2.2.147.4

Element	Specified in Section
admin	2.2.147.20
adoAdapterExtension	2.2.147.32
autoUpdatePrompt	2.2.147.45
command	2.2.147.12
commands	2.2.147.11
connectoid	2.2.147.30
contentType	2.2.147.16
contentTypeTemplate	2.2.147.17
dataConnections	2.2.147.28
davAdapterExtension	2.2.147.31
emailAdapterExtension	2.2.147.35
errorMessage	2.2.147.54
exportToPDForXPS	2.2.147.56
featureRestrictionsExtension	2.2.147.55
fieldExtension	2.2.147.27
fieldsExtension	2.2.147.26
includedView	2.2.147.23
includedViews	2.2.147.22
inputScope	2.2.147.47
inputScopes	2.2.147.46
install	2.2.147.14
listPropertiesExtension	2.2.147.25
mail	2.2.147.19
managedCode	2.2.147.50
mergedPrintView	2.2.147.21
offline	2.2.147.24
preview	2.2.147.44
relativeQuery	2.2.147.34
sendByMail	2.2.147.38
server	2.2.147.9
share	2.2.147.18
sharepointListAdapterExtension	2.2.147.37

Element	Specified in Section
solutionDefinition	2.2.147.8
solutionPropertiesExtension	2.2.147.13
submit	2.2.147.51
submitAction	2.2.147.52
successMessage	2.2.147.53
toolbar	2.2.147.10
useHttpHandlerExtension	2.2.147.29
viewExtension	2.2.147.42
viewsExtension	2.2.147.41
warning	2.2.147.40
warnings	2.2.147.39
webServiceAdapterExtension	2.2.147.33
word	2.2.147.49
words	2.2.147.48
wss	2.2.147.15
xmlFileAdapterExtension	2.2.147.36
xmlToEditExtension	2.2.147.43

2.2.147.1 serverCommandActionType

The **serverCommandActionType** simple type specifies restrictions for specifying a form action on the form **toolbar**.

Referenced By
command.xsf2.2.147.12@action

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this simple type.

```
<xsd:simpleType name="serverCommandActionType">
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:pattern value="[a-zA-Z0-9_]*"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.147.2 emailAttachmentType

The **emailAttachmentType** simple type specifies restrictions for specifying the file format of an attached form or form template when it is sent in e-mail.

Referenced By
emailAdapterExtension.xsf2.2.147.35@emailAttachmentType

Referenced By
sendByMail.xsf2.2.147.38@emailAttachmentType

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this simple type.

```
<xsd:simpleType name="emailAttachmentType">
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:pattern value="[a-zA-Z0-9_]*"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.147.3 compatibilityModesType

The **compatibilityModesType** simple type specifies restrictions for specifying the compatibility mode for the form template.

Referenced By
solutionDefinition.xsf2.2.147.8@runtimeCompatibility

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this simple type.

```
<xsd:simpleType name="compatibilityModesType">
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:pattern value="[a-zA-Z0-9 ]*"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.147.4 solutionType

The **solutionType** simple type specifies restrictions for an attribute that, if present, MUST be ignored.

Referenced By
solutionDefinition.xsf2.2.147.8@solutionType

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this simple type.

```
<xsd:simpleType name="solutionType">
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:pattern value="[a-zA-Z0-9]*"/>
  </xsd:restriction>
```

```
</xsd:simpleType>
```

2.2.147.5 formDescriptionType

The **formDescriptionType** simple type specifies restrictions for specifying the form template description.

Referenced By
solutionDefinition.xsf2.2.147.8@description

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this simple type.

```
<xsd:simpleType name="formDescriptionType">
  <xsd:restriction base="xsd:string">
    <xsd:maxLength value="1024"/>
    <xsd:minLength value="1"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.147.6 formLocaleType

The **formLocaleType** simple type specifies restrictions for specifying the locale of the form template.

Referenced By
server.xsf2.2.147.9@formLocale

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this simple type.

```
<xsd:simpleType name="formLocaleType">
  <xsd:restriction base="xsd:token">
    <xsd:minLength value="1"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.147.7 managedCodeType

The **managedCodeType** simple type specifies restrictions for specifying the business objects programming language used in the form template.

Referenced By
managedCode.xsf2.2.147.50@language

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this simple type.

```
<xsd:simpleType name="managedCodeType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[a-zA-Z0-9\.]*"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.147.8 solutionDefinition

The **solutionDefinition** element specifies the extended properties and features of the form template. This element also enables extending the form definition (.xsf) file with custom attributes not specified in section 2.2 or section 2.2.147. Custom attributes MUST NOT be defined under the **xsf** or **xsf2** namespace. Any specified custom attribute MUST be ignored. This element MUST be the root element of all extensions to the form template.

Child Elements
autoUpdatePrompt
dataConnections
featureRestrictionsExtension
inputScopes
listPropertiesExtension
managedCode
mergedPrintView
offline
preview
sendByMail
server
solutionPropertiesExtension
submit
viewsExtension
warnings

Attributes:

allowClientOnlyCode: This attribute specifies whether a browser-compatible form template is designed to enable the inclusion of client-specific object model code that is not compatible with the

protocol server. If this attribute is set to "yes", a warning MUST be generated when browser-enabling a form template containing incompatible code. If this attribute is set to "no", an error MUST be generated when browser-enabling a form template containing incompatible code.

description: This attribute specifies the description of the form template.

runtimeCompatibility: This attribute MUST be set to "client server".

runtimeCompatibilityURL: This attribute MUST be ignored.

solutionType: This attribute MUST be ignored.

verifyOnServer: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="solutionDefinition">
  <xsd:complexType>
    <xsd:all>
      <xsd:element ref="xsf2:server" minOccurs="0"/>
      <xsd:element ref="xsf2:solutionPropertiesExtension" minOccurs="0"/>
      <xsd:element ref="xsf2:mergedPrintView" minOccurs="0"/>
      <xsd:element ref="xsf2:offline" minOccurs="0"/>
      <xsd:element ref="xsf2:listPropertiesExtension" minOccurs="0"/>
      <xsd:element ref="xsf2:dataConnections" minOccurs="0"/>
      <xsd:element ref="xsf2:sendByMail" minOccurs="0"/>
      <xsd:element ref="xsf2:warnings" minOccurs="0"/>
      <xsd:element ref="xsf2:viewsExtension" minOccurs="0"/>
      <xsd:element ref="xsf2:preview" minOccurs="0"/>
      <xsd:element ref="xsf2:autoUpdatePrompt" minOccurs="0"/>
      <xsd:element ref="xsf2:inputScopes" minOccurs="0"/>
      <xsd:element ref="xsf2:managedCode" minOccurs="0"/>
      <xsd:element ref="xsf2:submit" minOccurs="0"/>
      <xsd:element ref="xsf2:featureRestrictionsExtension" minOccurs="0"/>
    </xsd:all>
    <xsd:attribute name="runtimeCompatibility" use="required">
      <xsd:simpleType>
        <xsd:list itemType="xsf2:compatibilityModesType"/>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="solutionType" type="xsf2.2.147.4:solutionType" use="optional"/>
    <xsd:attribute name="description" type="xsf2.2.147.5:formDescriptionType"
use="optional"/>
    <xsd:attribute name="allowClientOnlyCode" type="xsf:xdYesNo" use="optional"
default="no"/>
    <xsd:attribute name="runtimeCompatibilityURL" type="xsd:string" use="optional"/>
    <xsd:attribute name="verifyOnServer" type="xsf:xdYesNo" use="optional"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.9 server

The **server** element specifies display and functional properties for the form template.

Parent Elements
solutionDefinition

Child Elements
toolbar

Attributes:

formLocale: This attribute specifies the locale in which to render the form template. The specified value MUST be a valid locale, as defined by [MS-LCID].

isMobileEnabled: This attribute specifies whether the form template is a **browser-enabled form template** that can be rendered on a mobile device. This attribute MUST be set to "yes" for the form to be loaded in a mobile Web browser.

isPreSubmitPostBackEnabled: This attribute specifies whether the Web browser MUST **postback** the form prior to submitting the form file. If the form (1) will be post backed, the user MUST be notified that the form file will have to be submitted after the postback. If this attribute is not present, its value MUST be interpreted as "no".

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="server">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:toolbar" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="formLocale" type="xsf2.2.147.6:formLocaleType" use="required"/>
    <xsd:attribute name="isPreSubmitPostBackEnabled" type="xsf:xdYesNo" use="optional"/>
    <xsd:attribute name="isMobileEnabled" type="xsf:xdYesNo" use="optional"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.10 toolbar

The **toolbar** element specifies information about the toolbar that is displayed when a form is loaded.

Parent Elements
server

Child Elements
commands

Attributes:

enabledBottom: This attribute specifies whether the toolbar MUST be displayed at the bottom of the form (1).

enabledTop: This attribute specifies whether the toolbar MUST be displayed at the top of the form (1).

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="toolbar">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:commands" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="enabledTop" type="xsf:xdYesNo" use="optional" default="no"/>
    <xsd:attribute name="enabledBottom" type="xsf:xdYesNo" use="optional" default="no"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.11 commands

The **commands** element contains commands that are displayed on visible toolbars when a form is loaded.

Parent Elements
toolbar

Child Elements
command

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="commands">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:command" maxOccurs="unbounded" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.147.12 command

The **command** element specifies a command that MUST be displayed on the toolbar when a form is opened.

Parent Elements
commands

Attributes:

action: This attribute specifies an action that MUST be performed when the toolbar button is clicked. The value MUST be one of the following acceptable values:

- "submit"

- "print"
- "view"
- "save"
- "saveAs"
- "close"
- "refresh"

caption: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="command">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="action" type="xsf2.2.147.1:serverCommandActionType" use="required"/>
    <xsd:attribute name="caption" type="xsf:xdTitle" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.13 solutionPropertiesExtension

The **solutionPropertiesExtension** element MUST be ignored.

Parent Elements
solutionDefinition

Child Elements
admin
contentType
contentTypeTemplate
install
mail
share
wss

Attributes:

branch: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="solutionPropertiesExtension">
  <xsd:complexType>
    <xsd:all>
      <xsd:element ref="xsf2:install" minOccurs="0"/>
      <xsd:element ref="xsf2:wss" minOccurs="0"/>
      <xsd:element ref="xsf2:contentType" minOccurs="0"/>
      <xsd:element ref="xsf2:share" minOccurs="0"/>
      <xsd:element ref="xsf2:mail" minOccurs="0"/>
      <xsd:element ref="xsf2:admin" minOccurs="0"/>
      <xsd:element ref="xsf2:contentTypeTemplate" minOccurs="0"/>
    </xsd:all>
    <xsd:attribute name="branch" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="install"/>
          <xsd:enumeration value="wss"/>
          <xsd:enumeration value="contentType"/>
          <xsd:enumeration value="share"/>
          <xsd:enumeration value="mail"/>
          <xsd:enumeration value="admin"/>
          <xsd:enumeration value="contentTypeTemplate"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.14 install

The **install** element MUST be ignored.

Parent Elements
solutionPropertiesExtension

Attributes:

companyName: This attribute MUST be ignored.

language: This attribute MUST be ignored.

path: This attribute MUST be ignored.

updatePath: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="install">
  <xsd:complexType>
    <xsd:attribute name="companyName" type="xsd:string" use="required"/>
    <xsd:attribute name="language" type="xsd:string" use="required"/>
    <xsd:attribute name="path" type="xsd:string" use="required"/>
    <xsd:attribute name="updatePath" type="xsd:string" use="optional"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.15 wss

The **wss** element MUST be ignored.

Parent Elements
solutionPropertiesExtension

Attributes:

browserEnable: This attribute MUST be ignored.

description: This attribute MUST be ignored.

name: This attribute MUST be ignored.

path: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="wss">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="path" type="xsd:string" use="required"/>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="description" type="xsd:string" use="required"/>
    <xsd:attribute name="browserEnable" type="xsf:xdYesNo" use="optional"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.16 contentType

The **contentType** element MUST be ignored.

Parent Elements
solutionPropertiesExtension

Attributes:

path: This attribute MUST be ignored.

sharepointContentTypeId: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="contentType">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="path" type="xsd:string" use="required"/>
    <xsd:attribute name="sharepointContentTypeId" type="xsd:string" use="required"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
```

```
</xsd:element>
```

2.2.147.17 contentTypeTemplate

The **contentTypeTemplate** element MUST be ignored.

Parent Elements
solutionPropertiesExtension

Attributes:

browserEnable: This attribute MUST be ignored.

description: This attribute MUST be ignored.

name: This attribute MUST be ignored.

path: This attribute MUST be ignored.

site: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="contentTypeTemplate">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="site" type="xsd:string" use="required"/>
    <xsd:attribute name="path" type="xsd:string" use="required"/>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="description" type="xsd:string" use="required"/>
    <xsd:attribute name="browserEnable" type="xsf:xdYesNo" use="optional"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.18 share

The **share** element MUST be ignored.

Parent Elements
solutionPropertiesExtension

Attributes:

accessPath: This attribute MUST be ignored.

formName: This attribute MUST be ignored.

path: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="share">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="formName" type="xsd:string" use="required"/>
    <xsd:attribute name="path" type="xsd:string" use="required"/>
    <xsd:attribute name="accessPath" type="xsd:string" use="required"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.19 mail

The **mail** element MUST be ignored.

Parent Elements
solutionPropertiesExtension

Attributes:

formName: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="mail">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="formName" type="xsd:string" use="required"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.20 admin

The **admin** element MUST be ignored.

Parent Elements
solutionPropertiesExtension

Attributes:

path: This attribute MUST be ignored.

site: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="admin">
```

```
    <xsd:complexType>
      <xsd:sequence/>
      <xsd:attribute name="path" type="xsd:string" use="required"/>
      <xsd:attribute name="site" type="xsd:string" use="required"/>
      <xsd:anyAttribute processContents="skip"/>
    </xsd:complexType>
  </xsd:element>
```

2.2.147.21 mergedPrintView

The **mergedPrintView** element MUST be ignored.

Parent Elements
solutionDefinition

Child Elements
includedViews
printSettings

Attributes:

isCustomizable: This attribute MUST be ignored.

isDefault: This attribute MUST be ignored.

viewBreak: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="mergedPrintView">
  <xsd:complexType>
    <xsd:all>
      <xsd:element ref="xsf:printSettings" minOccurs="0"/>
      <xsd:element ref="xsf2:includedViews" minOccurs="0"/>
    </xsd:all>
    <xsd:attribute name="isDefault" type="xsf:xdYesNo" use="optional" default="no"/>
    <xsd:attribute name="isCustomizable" type="xsf:xdYesNo" use="optional" default="no"/>
    <xsd:attribute name="viewBreak" type="xsd:string" use="required"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.22 includedViews

The **includedViews** element MUST be ignored.

Parent Elements
mergedPrintView

Child Elements
includedView

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="includedViews">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:includedView" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.147.23 includedView

The **includedView** element MUST be ignored.

Parent Elements
includedViews

Attributes:

name: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="includedView">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="name" type="xsf:xdViewName" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.24 offline

The **offline** element MUST be ignored.

Parent Elements
solutionDefinition

Attributes:

cacheQueries: This attribute MUST be ignored.

expirationTime: This attribute MUST be ignored.

openIfQueryFails: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="offline">
  <xsd:complexType>
    <xsd:attribute name="openIfQueryFails" type="xsf:xdYesNo" default="no" use="optional"/>
    <xsd:attribute name="cacheQueries" type="xsf:xdYesNo" default="no" use="optional"/>
    <xsd:attribute name="expirationTime" type="xsd:nonNegativeInteger" use="optional"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.25 listPropertiesExtension

The **listPropertiesExtension** element MUST be ignored.

Parent Elements
solutionDefinition

Child Elements
fieldsExtension

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="listPropertiesExtension">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:fieldsExtension" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.147.26 fieldsExtension

The **fieldsExtension** element MUST be ignored.

Parent Elements
listPropertiesExtension

Child Elements
fieldExtension

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="fieldsExtension">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:fieldExtension" maxOccurs="unbounded" minOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.147.27 fieldExtension

The **fieldExtension** element MUST be ignored.

Parent Elements
fieldsExtension

Attributes:

columnId: This attribute MUST be ignored.

columnName: This attribute MUST be ignored.

readWrite: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="fieldExtension">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="columnName" type="xsd:string" use="required"/>
    <xsd:attribute name="readWrite" type="xsf:xdYesNo" use="optional" default="no"/>
    <xsd:attribute name="columnId" type="xsd:string" use="optional"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.28 dataConnections

The **dataConnections** element contains elements that specify extensions to data adapter connection settings.

Parent Elements
solutionDefinition

Child Elements
adoAdapterExtension
davAdapterExtension
emailAdapterExtension
sharepointListAdapterExtension
useHttpHandlerExtension
webServiceAdapterExtension
xmlFileAdapterExtension

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="dataConnections">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:useHttpHandlerExtension" minOccurs="0"/>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element ref="xsf2:davAdapterExtension" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="xsf2:adoAdapterExtension" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="xsf2:webServiceAdapterExtension" minOccurs="0"
maxOccurs="unbounded"/>
        <xsd:element ref="xsf2:emailAdapterExtension" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="xsf2:xmlFileAdapterExtension" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="xsf2:sharepointListAdapterExtension" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:choice>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.147.29 useHttpHandlerExtension

The **useHttpHandlerExtension** element specifies extended information for **useHttpHandler** (section 2.2.76).

Parent Elements
dataConnections

Child Elements
connectoid

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="useHttpHandlerExtension">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:connectoid" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.147.30 connectoid

The **connectoid** element specifies the location of a **Universal Data Connection (.udc, .udcx) file**, as specified by [MS-UDCX], containing data adapter connection settings that MUST override the connection settings specified in the form definition (.xsf) file. A .udc file provides the following benefits:

- Allows a form template to be published on multiple form servers and have different connection settings for each form server without modifying the connection settings in the form template.
- Allows multiple form templates to be published on multiple form servers and share the same connection settings.
- Allows a form template without an elevated form security level, as specified by the **requireFullTrust** attribute of the **xDocumentClass** element, as defined in section 2.2.20, to access specific data sources in a different domain.
- Allows the data adapter connection settings for a form template to be changed without modifying the form template (.xsn) file.

Parent Elements
adoAdapterExtension
davAdapterExtension
sharepointListAdapterExtension
useHttpHandlerExtension
webServiceAdapterExtension
xmlFileAdapterExtension

Attributes:

connectionLinkType: This attribute specifies the location context of the .udc file. The value MUST be set to one of the following values:

- **relative:** This value specifies that the file is located in the current **site collection**.

- **store:** This value specifies that the file is located in the global **data connection library**.

name: This attribute MUST be ignored.

siteCollection: This attribute specifies the URL to the root of the site collection where the original .udc file is located. This value MUST be identical for all **connectoid** elements in the form template.

source: This attribute specifies a URL that specifies the relative path from the **siteCollection** attribute to the .udc file.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="connectoid">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="siteCollection" type="xsd:string" use="required"/>
    <xsd:attribute name="source" type="xsd:string" use="required"/>
    <xsd:attribute name="connectionLinkType" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.31 davAdapterExtension

The **davAdaptorExtension** element specifies the extended information for **davAdapter** (section 2.2.48).

Parent Elements
dataConnections

Child Elements
connectoid

Attributes:

ref: This attribute specifies the associated **davAdapter** element that is being extended. The specified value SHOULD match the value specified by the **name** attribute of the corresponding **davAdapter** element.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="davAdapterExtension">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:connectoid" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="ref" type="xsf:xdTitle" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.32 adoAdapterExtension

The **adoAdapterExtension** element specifies extended information for **adoAdapter** (section 2.2.38).

Parent Elements
dataConnections

Child Elements
connectoid

Attributes:

queryFile: This attribute MUST be ignored.

queryKey: This attribute MUST be ignored.

ref: This attribute specifies the associated **adoAdapter** element that is being extended. The specified value SHOULD match the value specified by the **name** attribute of the corresponding **adoAdapter** element.

submitAdapterName: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="adoAdapterExtension">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:connectoid" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="ref" type="xsf:xdTitle" use="required"/>
    <xsd:attribute name="submitAdapterName" type="xsf:xdTitle" use="optional"/>
    <xsd:attributeGroup ref="xsf2:queryKeyFile"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.33 webServiceAdapterExtension

The **webServiceAdapterExtension** element specifies extended information for **webServiceAdapter** (section 2.2.39). This element MUST NOT have both a **connectoid** (section 2.2.147.30) child element and a **relativeQuery** (section 2.2.147.34) child element present.

Parent Elements
dataConnections

Child Elements
connectoid
relativeQuery

Attributes:

queryFile: This attribute MUST be ignored.

queryKey: This attribute MUST be ignored.

ref: This attribute specifies the associated **webServiceAdapter** element that is being extended. The specified value SHOULD match the value specified by the **name** attribute of the corresponding **webServiceAdapter** element.

trackDataSetChanges: This attribute MUST be set to "no".

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="webServiceAdapterExtension">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:connectoid" minOccurs="0"/>
      <xsd:element ref="xsf2:relativeQuery" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="trackDataSetChanges" type="xsf:xdYesNo" use="optional"
default="no"/>
    <xsd:attribute name="ref" type="xsf:xdTitle" use="required"/>
    <xsd:attributeGroup ref="xsf2:queryKeyFile"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.34 relativeQuery

The **relativeQuery** element specifies a substring of the specified Web service URL that is replaced at run time by a different substring to create a new Web service URL. This element is used when hosting a form that is published to multiple site collections that have different absolute root URLs to the site collection but the same relative paths to a Web service. The specified Web service URL substring MUST be replaced by the value specified by an implementation-specific **ASP.NET control** hosting the form.

Parent Elements
webServiceAdapterExtension

Attributes:

replace: This attribute specifies the substring of the Web service URL that is replaced at run time. The specified URL MUST be an absolute path and MUST NOT be a local or **Universal Naming Convention (UNC)** path. The specified URL MUST match the beginning of the value specified by the **serviceUrl** attribute of the **operation** element (section 2.2.41).

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="relativeQuery">
```

```
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="replace" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.35 emailAdapterExtension

The **emailAdapterExtension** element specifies the method of submitting the form file using the e-mail data adapter.

Parent Elements
dataConnections

Attributes:

emailAttachmentType: This attribute specifies the file type an attachment MUST be sent as when the form file is submitted using the e-mail data adapter. This attribute MUST be set to one of the following values:

- **none:** This value specifies that the form file is sent in the body of the e-mail.
- **xml:** This value specifies that the form file is attached to the e-mail as an XML file.
- **xmlXsn:** This value specifies that the form file and the form template (.xsn) file are both attached as two separate attachments to the e-mail.

ref: This attribute specifies the associated **emailAdapter** element (section 2.2.51) that is being extended. The specified value SHOULD match the value specified by the **name** attribute of the corresponding **emailAdapter** element.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="emailAdapterExtension">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="ref" type="xsf:xdTitle" use="required"/>
    <xsd:attribute name="emailAttachmentType" type="xsf2.2.147.2:emailAttachmentType"
use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.36 xmlFileAdapterExtension

The **xmlFileAdapterExtension** element specifies extended information for **xmlFileAdapter** (section 2.2.45).

Parent Elements
dataConnections

Child Elements
connectoid

Attributes:

queryFile: This attribute MUST be ignored.

queryKey: This attribute MUST be ignored.

ref: This attribute specifies the associated **xmlFileAdapter** element that is being extended. The specified value SHOULD match the value specified by the **name** attribute of the corresponding **xmlFileAdapter** element.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="xmlFileAdapterExtension">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:connectoid" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="ref" type="xsf:xdTitle" use="required"/>
    <xsd:attributeGroup ref="xsf2:queryKeyFile"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.37 sharepointListAdapterExtension

The **sharepointListAdapterExtension** element specifies extended information for **sharepointListAdapter** (section 2.2.46).

Parent Elements
dataConnections

Child Elements
connectoid

Attributes:

queryFile: This attribute MUST be ignored.

queryKey: This attribute MUST be ignored.

queryThisFormOnly: This attribute specifies whether the protocol server list data adapter MUST query the protocol server list for values applicable only to the current form.

ref: This attribute specifies the associated **sharepointListAdapter** element that is being extended. The specified value SHOULD match the value specified by the **name** attribute of the corresponding **sharepointListAdapter** element.

sharepointWebGuid: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="sharepointListAdapterExtension">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:connectoid" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="ref" type="xsf:xdTitle" use="required"/>
    <xsd:attribute name="queryThisFormOnly" type="xsf:xdYesNo" use="optional" default="no"/>
    <xsd:attribute name="sharepointWebGuid" type="xsd:string" use="optional"/>
    <xsd:attributeGroup ref="xsf2:queryKeyFile"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.38 sendByMail

The **sendByMail** element specifies whether the form file or form template is attached to the e-mail generated by the e-mail data adapter as a control-specific **MIME type**.

Parent Elements
solutionDefinition

Attributes:

disableEmailForms: This attribute specifies the MIME type of the attached form file or form template. If this attribute is not present, its value MUST be interpreted as "no".

- **no:** A form file MUST be attached as "application/x-microsoft-InfoPathForm" MIME type and a form template MUST be attached as an "application/x-microsoft-InfoPathFormTemplate" MIME Type.
- **yes:** A form file or form template MUST be attached as a "text/xml" MIME type.

emailAttachmentType: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="sendByMail">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="emailAttachmentType" type="xsf2.2.147.2:emailAttachmentType"
use="optional"/>
    <xsd:attribute name="disableEmailForms" type="xsf:xdYesNo" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.39 warnings

The **warnings** element MUST be ignored.

Parent Elements
solutionDefinition

Child Elements
warning

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="warnings">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:warning" maxOccurs="unbounded" minOccurs="0"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.40 warning

The **warning** element MUST be ignored.

Parent Elements
warnings

Attributes:

hidden: This attribute MUST be ignored.

source: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="warning">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="source" type="xsd:string" use="required"/>
    <xsd:attribute name="hidden" type="xsf:xdYesNo" use="optional" default="no"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.41 viewsExtension

The **viewsExtension** element contains extended information for the form views in this form template.

Parent Elements
solutionDefinition

Child Elements
viewExtension

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="viewsExtension">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:viewExtension" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.147.42 viewExtension

The **viewExtension** element specifies extended information for **view** (section 2.2.123).

Parent Elements
viewsExtension

Child Elements
xmlToEditExtension

Attributes:

clientOnly: This attribute specifies whether the form view contains features that are not compatible with the protocol server. If this attribute is set to "yes", the form view MUST NOT be rendered and MUST NOT be present in the menu of form views. If this attribute is not present, its value MUST be interpreted as "no".

designMode: This attribute MUST be ignored.

readOnly: This attribute MUST be ignored.

ref: This attribute specifies the name of the corresponding form view and MUST match the corresponding the **name** attribute of the **view** element.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="viewExtension">
  <xsd:complexType>
```

```
    <xsd:sequence>
      <xsd:element ref="xsf2:xmlToEditExtension" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="ref" type="xsf:xdTitle" use="required"/>
    <xsd:attribute name="designMode" type="xsd:string" use="optional"/>
    <xsd:attribute name="readOnly" type="xsf:xdYesNo" use="optional"/>
    <xsd:attribute name="clientOnly" type="xsf:xdYesNo" use="optional" default="no"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.43 xmlToEditExtension

The **xmlToEditExtension** element specifies extended information for **xmlToEdit** (section 2.2.124).

Parent Elements
viewExtension

Attributes:

allowLinkedImages: This attribute specifies whether hyperlink references to images are allowed in a rich text box control. This attribute MUST be set to "yes" if the corresponding **xmlToEdit** element refers to a rich text box control.

excludeEmbeddedImages: This attribute specifies whether embedded images are excluded in a rich text box control. This attribute MUST be set to "yes" if the corresponding **xmlToEdit** element refers to a rich text box control.

ref: This attribute specifies the name of the corresponding control and MUST match the corresponding **name** attribute of the **xmlToEdit** element.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="xmlToEditExtension">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="ref" type="xsf:xdTitle" use="required"/>
    <xsd:attribute name="excludeEmbeddedImages" type="xsf:xdYesNo" use="optional"
default="no"/>
    <xsd:attribute name="allowLinkedImages" type="xsf:xdYesNo" use="optional" default="no"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.44 preview

The **preview** element MUST be ignored.

Parent Elements
solutionDefinition

Attributes:

domain: This attribute MUST be ignored.

sampleData: This attribute MUST be ignored.

userRole: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="preview">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="sampleData" type="xsd:string" use="optional"/>
    <xsd:attribute name="domain" type="xsd:string" use="optional"/>
    <xsd:attribute name="userRole" type="xsd:string" use="optional"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.45 autoUpdatePrompt

The **autoUpdatePrompt** element MUST be ignored.

Parent Elements
solutionDefinition

Attributes:

showPrompt: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="autoUpdatePrompt">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="showPrompt" type="xsf:xdYesNo" use="optional"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.46 inputScopes

The **inputScopes** element MUST be ignored.

Parent Elements
solutionDefinition

Child Elements
inputScope

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="inputScopes">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:inputScope" maxOccurs="unbounded" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.147.47 inputScope

The **inputScope** element MUST be ignored.

Parent Elements
inputScopes

Child Elements
words

Attributes:

caption: This attribute MUST be ignored.

expression: This attribute MUST be ignored.

name: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="inputScope">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:words" maxOccurs="unbounded" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="caption" type="xsf:xdTitle" use="optional"/>
    <xsd:attribute name="expression" type="xsd:string" use="optional"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.48 words

The **words** element MUST be ignored.

Parent Elements
inputScope

Child Elements
word

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="words">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="word" maxOccurs="unbounded" minOccurs="1">
        <xsd:complexType>
          <xsd:sequence/>
          <xsd:attribute name="value" type="xsd:string" use="optional" default=""/>
          <xsd:anyAttribute processContents="skip"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.49 word

The **word** element MUST be ignored.

Parent Elements
words

Attributes:

value: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="word" maxOccurs="unbounded" minOccurs="1">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="value" type="xsd:string" use="optional" default=""/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
```

```
</xsd:element>
```

2.2.147.50 managedCode

The **managedCode** element specifies settings for business objects in the form template (.xsn) file. Business objects are loaded when a form template (.xsn) file is published. Platform specifics determine the success of loading business objects.

Parent Elements
solutionDefinition

Attributes:

enabled: This attribute MUST be ignored.

language: This attribute MUST be ignored.

projectPath: This attribute MUST be ignored.

version: This attribute specifies which version of the platform with which the business objects were compiled. The specified value MUST match a supported version of the platform installed on the protocol server.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="managedCode">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="projectPath" type="xsd:string" use="optional"/>
    <xsd:attribute name="language" type="xsf2.2.147.7:managedCodeType" use="required"/>
    <xsd:attribute name="version" type="xsd:string" use="required"/>
    <xsd:attribute name="enabled" type="xsf:xdYesNo" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.51 submit

The **submit** element MUST be ignored.

Parent Elements
solutionDefinition

Child Elements
errorMessage
submitAction

Child Elements
successMessage

Attributes:

caption: This attribute MUST be ignored.

disableMenuItem: This attribute MUST be ignored.

onAfterSubmit: This attribute MUST be ignored.

showSignatureReminder: This attribute MUST be ignored.

showStatusDialog: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="submit">
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="submitAction" minOccurs="0">
        <xsd:complexType>
          <xsd:sequence/>
          <xsd:attribute name="adapter" type="xsf:xdTitle" use="required"/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="successMessage" type="xsd:string" minOccurs="0"/>
      <xsd:element name="errorMessage" type="xsd:string" minOccurs="0"/>
    </xsd:all>
    <xsd:attribute name="caption" type="xsd:string" use="optional"/>
    <xsd:attribute name="onAfterSubmit" use="optional">
      <xsd:simpleType>
        <xsd:restriction base="xsd:NMTOKEN">
          <xsd:enumeration value="close"/>
          <xsd:enumeration value="keepOpen"/>
          <xsd:enumeration value="openNew"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="showStatusDialog" type="xsf:xdYesNo" use="optional"/>
    <xsd:attribute name="showSignatureReminder" type="xsf:xdYesNo" use="optional"/>
    <xsd:attribute name="disableMenuItem" type="xsf:xdYesNo" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.52 submitAction

The **submitAction** element MUST be ignored.

Parent Elements
submit

Attributes:

adapter: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="submitAction" minOccurs="0">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="adapter" type="xsf:xdTitle" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.147.53 successMessage

The **successMessage** element MUST be ignored.

Parent Elements
submit

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="successMessage" type="xsd:string" minOccurs="0"/>
```

2.2.147.54 errorMessage

The **errorMessage** element MUST be ignored.

Parent Elements
submit

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="errorMessage" type="xsd:string" minOccurs="0"/>
```

2.2.147.55 featureRestrictionsExtension

The **featureRestrictionsExtension** element MUST be ignored.

Parent Elements
solutionDefinition

Child Elements
exportToPDForXPS

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="featureRestrictionsExtension">
  <xsd:complexType>
    <xsd:all>
      <xsd:element ref="xsf2:exportToPDForXPS" minOccurs="0"/>
    </xsd:all>
  </xsd:complexType>
</xsd:element>
```

2.2.147.56 exportToPDForXPS

The **exportToPDForXPS** element MUST be ignored.

Parent Elements
featureRestrictionsExtension

Attributes:

ui: This attribute MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="exportToPDForXPS">
  <xsd:complexType>
    <xsd:attribute name="ui" type="xsf:xdEnabledDisabled" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.3 XML Schema Files (XSD) Specification

The XML schema documents in the form template (.xsn) file MUST be conformant to [XMLSCHEMA1].

This section specifies the XML schema documents representing the XML schema for the data of any form files based on the form template. The specific XML schema documents affected are specified by the **documentSchema** element (section 2.2.61) of the form definition (.xsf) file.

2.3.1 Control Representation

Each control that is bound to a field or group has a particular representation in the **XML schema definition (XSD)** language.

The following table lists the sections that specify what data types each control can be bound to in the XML schema document:

Section	Description
Button Control (section 2.3.1.1)	Specifies the representation of a button control in the XML schema document.
Check Box Control (section 2.3.1.2)	Specifies the representation of a check box control in the XML schema document.
Contact Selector Control (section 2.3.1.3)	Specifies the representation of a contact selector control in the XML schema document.
Date Picker Control (section 2.3.1.4)	Specifies the representation of a date picker control in the XML schema document.
Drop-Down List Control (section 2.3.1.5)	Specifies the representation of a drop-down control in the XML schema document.
Expression Box Control (section 2.3.1.6)	Specifies the representation of an expression box control in the XML schema document.
File Attachment Control (section 2.3.1.7)	Specifies the representation of a file attachment control in the XML schema document.
Hyperlink Control (section 2.3.1.8)	Specifies the representation of a hyperlink control in the XML schema document.
List Box Control (section 2.3.1.9)	Specifies the representation of a list box control in the XML schema document.
Option Button Control (section 2.3.1.10)	Specifies the representation of an option button control in the XML schema document.
Repeating Section Control (section 2.3.1.11)	Specifies the representation of a repeating section control in the XML schema document.
Repeating Table Control (section 2.3.1.12)	Specifies the representation of a repeating table control in the XML schema document.
Rich Text Box Control (section 2.3.1.13)	Specifies the representation of a rich text box control in the XML schema document.
Section Control and Optional Section Control section (2.3.1.14)	Specifies the representation of a section control in the XML schema document.
Table Control (section 2.3.1.15)	Specifies the representation of a table control in the XML schema document.
Text Box Control (section 2.3.1.16)	Specifies the representation of a text box control in the XML schema document.

2.3.1.1 Button Control

A button control MUST be unbound. It has no XSD representation.

2.3.1.2 Check Box Control

A check box control that is not required to contain data, for example that is bound to an XML element, named "field1" with data type set to "boolean" for which no constraining facets, as specified in [XMLSCHEMA1], have been set, has the following XSD:

```
<xsd:element name="field1" nillable="true" type="xsd:boolean"/>
```

A check box control that is required to contain data, for example that is bound to an XML element named "field1" with data type set to "boolean" for which no constraining facets, as specified in [XMLSCHEMA1], have been set, has the following XSD:

```
<xsd:element name="field1" type="xsd:boolean"/>
```

A check box control SHOULD be bound to a field with one of the following XSD data types, as specified in [XMLSCHEMA1], for which any valid constraining facets, as specified in [XMLSCHEMA1], MAY also be set:

- string
- integer
- double
- boolean
- anyURI
- date
- time
- dateTime

2.3.1.3 Contact Selector Control

A contact selector control bound to a group MUST have a complex type XSD, as specified in [XMLSCHEMA1]. The following example is the XSD in which the group (1) is named "group1":

```
<xsd:element name="group1">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="my:Person" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="Person">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="my:DisplayName" minOccurs="0"/>
            <xsd:element ref="my:AccountId" minOccurs="0"/>
            <xsd:element ref="my:AccountType" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="DisplayName" type="xsd:string"/>
<xsd:element name="AccountId" type="xsd:string"/>
<xsd:element name="AccountType" type="xsd:string"/>
```

2.3.1.4 Date Picker Control

A date picker control that is not required to contain data, for example that is bound to an XML element, named "field1" with data type set to "date" for which no constraining facets, as specified in [XMLSCHEMA1], have been set has an XSD as follows:

```
<xsd:element name="field1" nillable="true" type="xsd:date"/>
```

A date picker control that is required to contain data, for example that is bound to an XML element, named "field1" with data type set to "date" for which no constraining facets, as specified in [XMLSCHEMA1], have been set has the following XSD:

```
<xsd:element name="field1" type="xsd:date"/>
```

A date picker control SHOULD be bound to a field with one of the following XSD data types for which any valid constraining facets, as specified in [XMLSCHEMA1], MAY also be set:

- string
- date
- dateTime

2.3.1.5 Drop-Down List Control

A drop-down list control that is not required to contain data, for example that is bound to an XML element named "field1" with data type set to "string" for which no constraining facets, as specified in [XMLSCHEMA1], have been set, has the following XSD:

```
<xsd:element name="field1" type="xsd:string"/>
```

A drop-down list control that is required to contain data, for example that is bound to an XML element named "field1" with data type set to "string" for which a **xsd:minLength** constraining facet, as specified in [XMLSCHEMA1], has been set has the following XSD:

```
<xsd:element name="field1" type="my:requiredString"/>
<xsd:simpleType name="requiredString">
    <xsd:restriction base="xsd:string">
        <xsd:minLength value="1"/>
    </xsd:restriction>
</xsd:simpleType>
```

A drop-down list control SHOULD be bound to a field with one of the following XSD data types, as specified in [XMLSCHEMA1], for which any valid constraining facets, as specified in [XMLSCHEMA1], MAY also be set:

- string
- integer
- double
- boolean
- anyURI
- date
- time
- dateTime

2.3.1.6 Expression Box Control

An expression box control MAY be bound to a field to retrieve the value that it displays. If bound, an expression box control MUST NOT change the data in the field to which it is bound, though the data in that field MAY be changed by other relevant events within the form.

2.3.1.7 File Attachment Control

A file attachment control that is not required to contain data, which is bound to an XML element, for example named "field1", with data type set to "base64Binary" for which no constraining facets, as specified in [XMLSCHEMA1], have been set, has the following XSD:

```
<xsd:element name="field1" nillable="true" type="xsd:base64Binary"/>
```

A file attachment control that is required to contain data, for example that is bound to an XML element named "field1", with data type set to "xsd:base64Binary" for which an **xsd:minLength** constraining facet, as specified in [XMLSCHEMA1], has been set, has the following XSD:

```
<xsd:element name="field1" type="my:requiredBase64Binary"/>
<xsd:simpleType name="requiredBase64Binary">
    <xsd:restriction base="xsd:base64Binary">
        <xsd:minLength value="1"/>
    </xsd:restriction>
</xsd:simpleType>
```

A file attachment control SHOULD be bound to a field with the XSD data types **base64Binary**. Any valid constraining facets, as specified in [XMLSCHEMA1], MAY also be set.

2.3.1.8 Hyperlink Control

A hyperlink control MAY be bound to up to two fields; one field to retrieve the target of the hyperlink and another field to retrieve the value of the hyperlink's display text. If bound, a hyperlink control MUST NOT change the data in either field to which it is bound, though the data in those fields MAY be changed by other relevant events within the form.

2.3.1.9 List Box Control

A list box control that is not required to contain data, which is bound to an XML element named "field1", with data type set to "string" for which no constraining facets, as specified in [XMLSCHEMA1], have been set has the following XSD:

```
<xsd:element name="field1" type="xsd:string"/>
```

A list box control that is required to contain data, for example that is bound to an XML element, named "field1", with data type set to "string" for which an **xsd:minLength** constraining facet, as specified in [XMLSCHEMA1], has been set has the following XSD:

```
<xsd:element name="field1" type="my:requiredString"/>
<xsd:simpleType name="requiredString">
    <xsd:restriction base="xsd:string">
        <xsd:minLength value="1"/>
    </xsd:restriction>
</xsd:simpleType>
```

A list box control SHOULD be bound to a field with one of the following XSD data types, as specified in [XMLSCHEMA1], for which any valid constraining facets, as specified in [XMLSCHEMA1], MAY also be set:

- string
- integer
- double
- boolean
- anyURI
- date
- time
- dateTime

2.3.1.10 Option Button Control

An option button control that is not required to contain data, for example that is bound to an XML element named "field1", with data type set to "string" for which no constraining facets, as specified in [XMLSCHEMA1], have been set has the following XSD:

```
<xsd:element name="field1" type="xsd:string"/>
```

An option button control that is required to contain data, for example that is bound to an XML element named "field1", with data type set to "string" for which a **minLength** constraining facet, as specified in [XMLSCHEMA1], has been set has the following XSD:

```
<xsd:element name="field1" type="my:requiredString"/>
<xsd:simpleType name="requiredString">
    <xsd:restriction base="xsd:string">
        <xsd:minLength value="1"/>
    </xsd:restriction>
</xsd:simpleType>
```

An option button control SHOULD be bound to a field with one of the following XSD data types, as specified in [XMLSCHEMA1], for which any valid constraining facets, as specified in [XMLSCHEMA1], MAY also be set:

- string
- integer
- double
- boolean
- anyURI
- date
- time
- dateTime

2.3.1.11 Repeating Section Control

A repeating section control bound to a **repeating group**, for example named "group2" and containing no other bound controls, has the following complex type XSD:

```
<xsd:element name="group1">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="my:group2" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="group2">
    <xsd:complexType>
        <xsd:sequence/>
    </xsd:complexType>
</xsd:element>
```

A repeating section control bound to a repeating group, for example named "group2", and containing a text box control that is not required to contain data, which is bound to an XML element named "field1", with data type set to "string" for which no constraining facets, as specified in [XMLSCHEMA1], have been set has the following XSD:

```
<xsd:element name="group1">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="my:group2" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="group2">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="my:field1" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="field1" type="xsd:string"/>
```

2.3.1.12 Repeating Table Control

A repeating table control bound to a repeating group, for example named "group2", and containing three text box controls that are not required to contain data, which are bound to XML elements named "field1", "field2", and "field3" with data types set to "string" for which no constraining facets, as specified in [XMLSCHEMA1], have been set, has the following XSD:

```
<xsd:element name="group1">
    <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="my:group2" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
    <xsd:element name="group2">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="my:field1" minOccurs="0"/>
                <xsd:element ref="my:field2" minOccurs="0"/>
                <xsd:element ref="my:field3" minOccurs="0"/>
            </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

```
    <xsd:element name="field1" type="xsd:string"/>
    <xsd:element name="field2" type="xsd:string"/>
    <xsd:element name="field3" type="xsd:string"/>
```

2.3.1.13 Rich Text Box Control

A rich text box control, which is bound to an XML element, for example named "field1", has the following complex type XSD:

```
<xsd:element name="field1">
    <xsd:complexType mixed="true">
        <xsd:sequence>
            <xsd:any minOccurs="0" maxOccurs="unbounded"
namespace="http://www.w3.org/1999/xhtml" processContents="lax"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

2.3.1.14 Section Control and Optional Section Control

A section control or optional section control bound to a group, for example named "group1" and containing no other bound controls, has the following complex type XSD:

```
<xsd:element name="group1">
    <xsd:complexType>
        <xsd:sequence/>
    </xsd:complexType>
</xsd:element>
```

A section control or optional section control bound to a group, for example named "group1", and containing a text box control that is not required to contain data, which is bound to an XML element, named "field1", with data type set to "string" for which no constraining facets, as specified in [XMLSCHEMA1], have been set, has the following XSD:

```
<xsd:element name="group1">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="my:field1" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
    </xsd:element>
<xsd:element name="field1" type="xsd:string"/>
```

2.3.1.15 Table Control

A table control MUST be unbound. It has no XSD representation.

2.3.1.16 Text Box Control

A text box control that is not required to contain data, for example that is bound to an XML element named "field1", with data type set to "string" for which no constraining facets, as specified in [XMLSCHEMA1], have been set, has the following XSD:

```
    <xsd:element name="field1" type="xsd:string"/>
```

A text box control that is required to contain data, which is bound to an XML element, for example named "field1", with data type set to "string" for which an **xsd:minLength** constraining facet, as specified in [XMLSCHEMA1], has been set, has the following XSD:

```
<xsd:element name="field1" type="my:requiredString"/>
<xsd:simpleType name="requiredString">
    <xsd:restriction base="xsd:string">
        <xsd:minLength value="1"/>
    </xsd:restriction>
</xsd:simpleType>
```

A text box control SHOULD be bound to a field with one of the following XSD data types, as specified in [XMLSCHEMA1], for which any valid constraining facets, as specified in [XMLSCHEMA1], MAY also be set:

- string
- integer
- double
- boolean
- anyURI
- date
- time
- dateTime

2.4 Form View Files (XSLT) Specification

The view XSL file MUST be an XSLT valid transformation, as specified in [W3C-XSLT], which MUST produce a valid HTML document, as specified in [HTML]. The HTML document MUST be a valid **XML document**, as specified in [W3C-XML]. The XSL file uses constructs with a specific pattern. The following sections specify the pattern of valid XSLT transformations.

The topic is divided into the following sections:

- **View Representation** (section 2.4.1)**.** This section specifies a valid XSLT transformation of the form data into HTML.
- **Control-specific Attributes** (section 2.4.2)**.** This section specifies the HTML representation of certain control properties and behaviors.
- **XSL Function Extensions** (section 2.4.3). This section specifies extensions to XSLT used in transforming the form data to HTML.

2.4.1 View Representation

Each control MUST have the representation specified in the following sections. Simple or complex XSD values, as specified in [XMLSCHEMA1], MUST be rendered using controls.

The following table lists the sections that specify the different constructs used to represent the XSLT file.

Section	Description
View Syntax (section 2.4.1.1)	Specifies the syntax used to represent the XSLT file.

Section	Description
XSL Root Template (section 2.4.1.2)	Specifies the starting element that contains the XSLT file.
XSL Root Template Style Sheets (section 2.4.1.3)	Specifies the representation for the style sheet.
Control Data Formatting (section 2.4.1.4)	Specifies the representation of data formatting for multiple controls.
Button Control (section 2.4.1.5)	Specifies the representation of a button control.
Check Box Control (section 2.3.1.2)	Specifies the representation of a check box control.
Contact Selector Control (section 2.4.1.7)	Specifies the representation of a contact selector control.
Date Picker Control (section 2.4.1.8)	Specifies the representation of a date picker control.
Drop-Down List Control (section 2.4.1.9)	Specifies the representation of a drop-down list control.
Expression Box Control (section 3.4.1.6)	Specifies the representation of an expression box control.
File Attachment Control (section 2.4.1.11)	Specifies the representation of a file attachment control.
Hyperlink Control (section 2.4.1.12)	Specifies the representation of a hyperlink control.
List Box Control (section 2.4.1.13)	Specifies the representation of a list box control.
Option Button Control (section 2.4.1.14)	Specifies the representation of an option button control.
Repeating Section Control (section 2.4.1.15)	Specifies the representation of a repeating section control.
Repeating Table Control (section 2.4.1.16)	Specifies the representation of a table control.
Rich Text Box Control (section 2.3.1.13)	Specifies the representation of a rich text box control.
Section Control and Optional Section Control (section 2.4.1.18)	Specifies the representation of a section control.
Table Control (section 2.4.1.19)	Specifies the representation of a table control.
Text Box Control (section 2.4.1.20)	Specifies the representation of a text box control.
Ignored Controls (section 2.4.1.21)	Specifies a list of controls that are ignored.
Invalid Controls (section 2.4.1.22)	Specifies a list of controls that are invalid.
Invalid Constructs (section 2.4.1.23)	Specifies a list of invalid constructs for the XSLT.

2.4.1.1 View Syntax

The formal grammar of the view XSL file is given in this specification using an **Extended Backus-Naur Form (EBNF)** notation. The EBNF notation is used instead of **Augmented Backus-Naur Form (ABNF)** or XML schema definition (XSD) to enhance the clarity of the constructs used in the XSLT transformation.

Notation

Each rule in the grammar defines one symbol, in the form.

SYMBOL::= expression

Symbols are written with capital and bold letters (SYMBOL). If a symbol has a subscript in a rule (1), the symbol MUST be expanded to the same yield in all places inside the rule (1).

#xN -where N is a hexadecimal integer, the expression matches the character whose number, or code point, in [ISO-10646] is "N". The number of leading zeros in the #xN form (1) is insignificant.

[a-zA-Z], [#xN-#xN]: Matches any character with a value in the range(s) indicated, inclusive.

[abc], [#xN#xN#xN]: Matches any character with a value among the characters enumerated. Enumerations and ranges can be mixed in one set of brackets.

[^a-z], [^#xN-#xN]: Matches any character with a value outside the range indicated.

[^abc], [^#xN#xN#xN]: Matches any character with a value not among the characters given. Enumerations and ranges of forbidden values can be mixed in one set of brackets.

"string": Matches a literal string matching that given inside the double quotes.

'string': Matches a literal string matching that given inside the single quotes.

To match more complex patterns, these symbols MUST be combined as follows, where A and B represent simple expressions:

(expression): Expression is treated as a unit and MUST be combined as specified in this list.

semicolon-delimited list((expression)(, expression)*): Matches a semicolon-delimited list of expressions.

A?: Matches A or nothing; optional A.

A B: Matches A followed by B. This operator has higher precedence than alternation; thus A B | C D is identical to (A B) | (C D).

A | B: Matches A or B.

A – B: Matches any string that matches A but does not match B.

A+: Matches one or more occurrences of A. Concatenation has higher precedence than alternation; thus A+ | B+ is identical to (A+) | (B+).

A*: Matches zero or more occurrences of A. Concatenation has higher precedence than alternation; thus A* | B* is identical to (A*) | (B*).

text: The text that does not match any production specified earlier MUST be interpreted as a literal. Additionally, any construct that has the same semantics in the target language ([HTML], [CSS-LEVEL1], [W3C-XML],[XMLSCHEMA1],[XPATH], and [W3C-XSLT]) can substitute the literal text.

The order and value of element attributes in the EBNF rules (1) MUST be interpreted in accordance with the target language.

For example, in HTML as a target language, the following constructs are semantically equivalent:

```
<span  class="xdTextBox xdBehavior_Formatting"/>
<span CLASS="xdTextBox xdBehavior_Formatting"/>
<span Class="xdTextBox      xdBehavior_Formatting"/>
```

This is true because the **class** attribute is specified in HTML and the syntax of the attribute is case-insensitive. Also the number of white spaces or tabs between **xdTextBox** and **xdBehavior_Formatting** is not important as long as there is at least one. The syntax for the values of **class** attributes MUST be as specified in [HTML] section 7.5.2.

The following productions MUST be used for the controls representation:

- ISO_646_DIGIT: [#x0030-#x0039]
- LATIN_CHARACTER: [#x0041-#x005A] | [#x0061- #x007a]
- SINGLE_CHARACTER: LATIN_ CHARACTER | ISO_646_DIGIT
- BUTTON_POSTBACKMODEL: always | auto
- POSTBACKMODEL: never | BUTTON_POSTBACKMODEL

The semantics of the post back model values MUST be as specified in section 2.4.2.29.

CONTROL_ID: MUST be (LATIN_CHARACTER) (LATIN_CHARACTER|ISO_646_DIGIT|_)*. The value of **CONTROL_ID** MUST be a valid value for type **xsf:xdTitle**.

TEMPLATE_MODE_ID: MUST be _(ISO_646_DIGIT)*.

ANY_STRING: MUST be a value of **Reference**, as specified in [W3C-XML] section 4.1.

NON_EMPTY_STRING: MUST be a value of **Reference,** as specified in [W3C-XML] section 4.1 that contains at least one **char**.

XML_NAMESPACE: Values MUST be as specified in [XML Namespaces].

INPUT_SCOPE_ID: MUST be **ANY_STRING**.

INPUT_SCOPE_NAME: MUST be **ANY_STRING.**

INPUT_SCOPE: MUST be the following:

```
xd:inputScopeId="INPUT_SCOPE_ID" (xd:inputScope="INPUT_SCOPE_NAME")?
(xd:allowNonMatching="yes")?
```

The semantics of the input scope attributes MUST be as specified in section 2.4.2.21, section 2.4.2.20, and section 2.4.2.2.

ALIGN: MUST be "left" or "right".

VALIGN: MUST be one of the following values:

- "middle"
- "baseline"
- "bottom"
- "top"

SIZE: Values MUST be as specified in [HTML] section 17.4.

ANCHOR_TEXT: Values MUST be as specified in [HTML] section 12.2. MUST NOT contain an anchor tag, as specified in [HTML] section 12.2.

XML_TO_EDIT_NAME: Nmtoken, as specified in [W3C-XML], and MUST match the name of a corresponding **xmlToEdit** (section 2.2.124) entry in the XSF file.

TAB_INDEX<4>: MUST be "-1" or as specified in [HTML] section 17.11.

HEIGHT: Value pairs MUST be as specified in [HTML] section 13.7.1.

MIN_HEIGHT: Values MUST be as specified in [CSS-LEVEL2] section 10.7.

WIDTH: Value pairs MUST be as specified in [HTML] section 13.7.1.

COLSPAN: Value pairs MUST be as specified in [HTML] section 11.2.6.

ROWSPAN: Value pairs MUST be as specified in [HTML] section 11.2.6.

STYLE_SIZE: Value pairs MUST be as specified in [HTML] section 17.4 and [CSS-LEVEL1] sections 5.5.23 and 5.5.24.

STYLE_WIDTH: Value pairs MUST be as specified in [CSS-LEVEL1] sections 5.5.23.

STYLE_HEIGHT: Value pairs MUST be as specified in [CSS-LEVEL1] sections 5.5.24.

CSS1_STYLE: Values MUST be as specified in [CSS-LEVEL1].

STYLE_DISPLAY_NONE: MUST be "DISPLAY: none", as specified in [CSS-LEVEL1] section 5.6.1.

STYLE_MARGIN: Value pairs MUST be as specified in [CSS-LEVEL1] sections 5.5.1, 5.5.2, 5.5.3, 5.5.4, 5.5.5.

STYLE_PADDING: Value pairs MUST be as specified in [CSS-LEVEL1] sections 5.5.6, 5.5.7, 5.5.8, 5.5.9, 5.5.10.

STYLE_TEXT_DECORATION: Value pairs MUST be as specified in [CSS-LEVEL1] section 5.4.3.

STYLE_BACKGROUND_COLOR: Value pairs MUST be as specified in [CSS-LEVEL1] section 5.3.2.

STYLE_BORDER: Value pairs MUST be as specified in [CSS-LEVEL1] sections 5.5.11, 5.5.12, 5.5.13, 5.5.14, 5.5.15, 5.5.16, 5.5.17, 5.5.18, 5.5.19, 5.5.20, 5.5.21, 5.5.22.

STYLE_BORDER_STYLE: Value pairs MUST be as specified in [CSS-LEVEL2] section 8.5.3.

STYLE_BORDER_COLLAPSE: Value pairs MUST be as specified in [CSS-LEVEL2] section 17.6.

STYLE_FONT: Value pairs MUST be as specified in [CSS-LEVEL1] section 5.2.

STYLE_FONT_STYLE: Value pairs MUST be as specified in [CSS-LEVEL1] section 5.2.3.

STYLE_COLOR: Value pairs MUST be as specified in [CSS-LEVEL1] section 5.3.1.

STYLE_FONT_WEIGHT: Value pairs MUST be as specified in [CSS-LEVEL1] section 5.2.5.

STYLE_TEXT_ALIGN: Value pairs MUST be as specified in [CSS-LEVEL1] section 5.4.6.

STYLE_WRAP: WHITE-SPACE: normal | WHITE-SPACE: nowrap; WORD-WRAP: normal

STYLE_OVERFLOW: Value pairs MUST be as specified in [CSS-LEVEL2] section 11.1.1.

STYLE_VERTICAL_ALIGN: VERTICAL-ALIGN: (sub | super)

STYLE_DIRECTION: (DIRECTION: ltr) | (DIRECTION: rtl)

STYLE_DISABLE_CHILD_XML_TO_EDIT:

msos-(xOptional|xCollection)-XML_TO_EDIT_NAME-editing:disabled;

LEAF_CONTROL_CONDITIONAL_FORMATTING_CAPTION: STYLE_DISPLAY_NONE | semicolon delimited list of (STYLE_FONT?, STYLE_COLOR?, STYLE_BACKGROUND_COLOR?, STYLE_TEXT_DECORATION?)

AUX_DOM_SOURCE_NAME: (LATIN_CHARACTER |_) (LATIN_CHARACTER|ISO_646_DIGIT|_)*. The value of **AUX_DOM_SOURCE_NAME** MUST be a valid **name** attribute of **dataObject** (section 2.2.36).

LEAF_XPATH: MUST be an extended location XPath expression, which MUST NOT contain XPath predicates, as specified in [XPATH] section 2.4, and MUST use only the **child** and **attribute** XPath axes, as specified in [XPATH] section 2.2. The generated node-set MUST have only 1 element. The XPath expression MUST be a relative or absolute XPath expression. To alter the context of the XPath expression evaluation, the **xdXDocument:GetDOM** function (section 2.4.3.9.2) MUST be used before the first XPath expression step.

GROUP_XPATH: MUST be an extended location XPath expression, which MUST NOT contain XPath predicates, as specified in [XPATH] section 2.4, and MUST use only the child XPath axes, as specified

in [XPATH] section 2.2. There are no restrictions for the number of elements in the generated node-set. The XPath expression MUST be a relative or absolute XPath expression. To alter the context of the XPath evaluation, the **xdXDocument:GetDOM** function (section 2.4.3.9.2) MUST be used before the first XPath expression step.

RELATIVE_REPEATING_GROUP_XPATH: MUST be a location XPath expression that MUST NOT contain XPath predicates, as specified in [XPATH] section 2.4, and MUST use only the child XPath axes, as specified in [XPATH] section 2.2. There are no restrictions for the number of elements in the generated node-set. The XPath expression MUST be a relative XPath expression.

RELATIVE_GROUP_XPATH: MUST be an extended location XPath expression that MUST NOT contain XPath predicates, as specified in [XPATH] section 2.4, and MUST use only the child XPath axes, as specified in [XPATH] section 2.2. There are no restrictions for the number of elements in the generated node-set. The XPath expression MUST be a relative XPath expression.

RELATIVE_LEAF_XPATH: MUST be a location XPath expression that MUST NOT contain XPath predicates, as specified in [XPATH] section 2.4, and MUST use only the child and attribute XPath axes, as specified in [XPATH] section 2.2. The XPath expression MUST be a relative XPath expression.

BOOLEAN_XPATH_EXPRESSION: MUST be an XPath expression that yields an object that MUST be a **Boolean** basic type. Boolean is specified in [XPATH] section 3.4. To extend the syntax of the XPath expression, the XSL function extensions specified in section 2.4.3 MUST be used. The XPath expression MUST be less than 100 in depth. It MUST NOT use the **position** and **last** functions specified in [XPATH] section 4.1. It MUST NOT use XPath predicates, as specified in [XPATH] section 2.4.

STRING_XPATH_EXPRESSION: MUST be an XPath expression that yields an object that has a **String** basic type. **String** is specified in [XPATH] section 3.6. To extend the syntax of XPath expressions, the XSL function extensions specified in section 2.4.3 MUST be used. The XPath expression MUST be less than 100 in depth. It MUST NOT use the **position** and **last** functions specified in [XPATH] section 4.1. It MUST NOT use XPath predicates, as specified in [XPATH] section 2.4.

CHECK_FOR_GETDOM_BEGIN: (<xsl:if test="function-available('xdXDocument:GetDOM')">)?

CHECK_FOR_GETDOM_END: (</xsl:if>)?

CHECK_FOR_GETDOM_BEGIN and **CHECK_FOR_GETDOM_END** symbols always appear in pairs in the EBNF rules in the following sections. Subscripts are used to mark the pairs.

If the yield of **CHECK_FOR_GETDOM_BEGIN** in one production is empty, the yield of the pairing **CHECK_FOR_GETDOM_END** MUST be empty.

If the yield of **CHECK_FOR_GETDOM_END** in one production is empty, the yield of the pairing **CHECK_FOR_GETDOM_BEGIN** MUST be empty.

2.4.1.2 XSL Root Template

The starting element for the EBNF notation is **XSL_STYLE_SHEET**.

```
XSL_STYLE_SHEET ::=
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    (XML NAMESPACE)*
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    (xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance")?
    (xmlns:xd="http://schemas.microsoft.com/office/infopath/2003")?
    (xmlns:msxsl="urn:schemas-microsoft-com:xslt")?
    (xmlns:x="urn:schemas-microsoft-com:office:excel")?
    (xmlns:xdExtension="http://schemas.microsoft.com/office/infopath/2003/xslt/extension")?
    (xmlns:xdXDocument="http://schemas.microsoft.com/office/infopath/2003/xslt/xDocument")?
```

```
    (xmlns:xdSolution="http://schemas.microsoft.com/office/infopath/2003/xslt/solution")?
    (xmlns:xdFormatting="http://schemas.microsoft.com/office/infopath/2003/xslt/formatting")?
    (xmlns:xdImage="http://schemas.microsoft.com/office/infopath/2003/xslt/xImage")?
    (xmlns:xdUtil="http://schemas.microsoft.com/office/infopath/2003/xslt/Util")?
    (xmlns:xdMath="http://schemas.microsoft.com/office/infopath/2003/xslt/Math")?
    (xmlns:xdDate="http://schemas.microsoft.com/office/infopath/2003/xslt/Date")?
    (xmlns:sig="http://www.w3.org/2000/09/xmldsig#")?
(xmlns:xdSignatureProperties="http://schemas.microsoft.com/office/infopath/2003/SignatureProp
erties")?
    (xmlns:ipApp="http://schemas.microsoft.com/office/infopath/2006/XPathExtension/ipApp")?

(xmlns:xdEnvironment="http://schemas.microsoft.com/office/infopath/2006/xslt/environment")?
    (xmlns:dfs="http://schemas.microsoft.com/office/infopath/2003/dataFormSolution")?
    (xmlns:xsf="http://schemas.microsoft.com/office/infopath/2003/solutionDefinition")?

(xmlns:xsf2="http://schemas.microsoft.com/office/infopath/2006/solutionDefinition/extensions"
)?
    (xmlns:xsd="http://www.w3.org/2001/XMLSchema")?
    (xmlns:xhtml="http://www.w3.org/1999/xhtml")?
    (xmlns:xdUser="http://schemas.microsoft.com/office/infopath/2006/xslt/User")? >
    <xsl:output method="html" indent="no"/>
    <xsl:template match="GROUP_XPATH">
        <html (dir="HTML DIR")?>
            <head>
               (HTML_COMMENTS)?
               <meta http-equiv="Content-Type" content="text/html"></meta>
               (CONTROL_STYLE)?
               TABLE_STYLE
               LANGUAGE STYLE
               (THEME_STYLE)?
            </head>
            <body (style="CSS1_STYLE")? (background="IMAGE_FILE")?
                (scroll="auto")?>MAIN_BODY</body>
        </html>
    </xsl:template>
    (SECTION BODY | REPEATING SECTION BODY)*
</xsl:stylesheet>

MAIN_BODY ::= XML_HTML_4_1_WITH_CONTROLS |
<span>
    <xsl:attribute name="style">
        (<xsl:if test="BOOLEAN XPATH EXPRESSION">STYLE DISABLE CHILD XML TO EDIT</xsl:if>)+
    </xsl:attribute>
    XML_HTML_4_1_WITH_CONTROLS
</span>
```

XML_HTML_4_1_WITH_CONTROLS: MUST be an HTML 4.1 fragment, as specified in [HTML], valid under the **BODY** element that is also a valid XML 1.0 fragment, as specified in [W3C-XML]. If an element inside the fragment contains the **xd:xctname** attribute, it MUST conform to one of the control productions specified for controls in section 2.4.1.5 to section 2.4.1.20. If the fragment contains an XSL element with the syntax of **SECTION_CALL**, it MUST be located only in the locations where a **<DIV/>** element, as specified in [HTML] section 7.5.4, could also be placed.

SECTION_CALL: SIMPLE_SECTION_CALL | OPTIONAL_SECTION_CALL | REPEATING_SECTION_CALL

HTML_COMMENTS: MUST be a concatenation of one or more HTML 4.1 comments, as specified in [HTML] section 3.2.4.

HTML_DIR: The values MUST be as specified in [HTML] section 8.2.

IMAGE_FILE: MUST be the name of an image file, as specified in [CSS-LEVEL1] section 5.3.7. The image file MUST be present in the form template. See section 2.2.98.

2.4.1.3 XSL Root Template Style Sheets

The following rules specify the CSS1 style sheets, as specified in [CSS-LEVEL1], used in the **head** element.

CONTROL_STYLE yields are associated with a client-only feature and MUST be ignored by the form server.

CONTROL_STYLE:

```
<style controlStyle="controlStyle">
    @media screen
    {
BODY{margin-left:21px;background-position:21px 0px;}
    }
    BODY{color:windowtext;background-color:window;layout-grid:none;}
    .xdListItem {display:inline-block;width:100%;vertical-align:text-top;}
    .xdListBox,.xdComboBox{margin:1px;}
    .xdInlinePicture{margin:1px; BEHAVIOR: url(#default#urn::xdPicture) }
    .xdLinkedPicture{margin:1px; BEHAVIOR: url(#default#urn::xdPicture)
url(#default#urn::controls/Binder) }
    .xdSection{border:1pt solid #FFFFFF;margin:6px 0px 6px 0px;padding:1px 1px 1px 5px;}
    .xdRepeatingSection{border:1pt solid #FFFFFF;margin:6px 0px 6px 0px;padding:1px 1px 1px
5px;}
    .xdMultiSelectList{margin:1px;display:inline-block; border:1pt solid #dcdcdc; padding:1px
1px 1px 5px; text-indent:0; color:windowtext; background-color:window; overflow:auto;
behavior: url(#default#DataBindingUI) url(#default#urn::controls/Binder)
url(#default#MultiSelectHelper) url(#default#ScrollableRegion);}
    .xdMultiSelectListItem{display:block;white-space:nowrap}
    .xdMultiSelectFillIn{display:inline-block;white-space:nowrap;text-
overflow:ellipsis;;padding:1px;margin:1px;border: 1pt solid #dcdcdc;overflow:hidden;text-
align:left;}
    .xdBehavior Formatting {BEHAVIOR: url(#default#urn::controls/Binder)
url(#default#Formatting);}
    .xdBehavior_FormattingNoBUI{BEHAVIOR: url(#default#CalPopup)
url(#default#urn::controls/Binder) url(#default#Formatting);}
    .xdExpressionBox{margin: 1px;padding:1px;word-wrap: break-word;text-overflow:
ellipsis;overflow-x:hidden;}
    .xdBehavior GhostedText,
    .xdBehavior_GhostedTextNoBUI{BEHAVIOR: url(#default#urn::controls/Binder)
url(#default#TextField) url(#default#GhostedText);}
    .xdBehavior GTFormatting{BEHAVIOR: url(#default#urn::controls/Binder)
url(#default#Formatting) url(#default#GhostedText);}
    .xdBehavior GTFormattingNoBUI{BEHAVIOR: url(#default#CalPopup)
url(#default#urn::controls/Binder) url(#default#Formatting) url(#default#GhostedText);}
    .xdBehavior_Boolean{BEHAVIOR: url(#default#urn::controls/Binder)
url(#default#BooleanHelper);}
    .xdBehavior Select{BEHAVIOR: url(#default#urn::controls/Binder)
url(#default#SelectHelper);}
    .xdBehavior ComboBox{BEHAVIOR: url(#default#ComboBox)}
    .xdBehavior_ComboBoxTextField{BEHAVIOR: url(#default#ComboBoxTextField);}
    .xdRepeatingTable{BORDER-TOP-STYLE: none; BORDER-RIGHT-STYLE: none; BORDER-LEFT-STYLE:
none; BORDER-BOTTOM-STYLE: none; BORDER-COLLAPSE: collapse; WORD-WRAP: break-word;}
    .xdScrollableRegion{BEHAVIOR: url(#default#ScrollableRegion);}
    .xdLayoutRegion{display:inline-block;}
    .xdMaster{BEHAVIOR: url(#default#MasterHelper);}
    .xdActiveX{margin:1px; BEHAVIOR: url(#default#ActiveX);}
    .xdFileAttachment{display:inline-
block;margin:1px;BEHAVIOR:url(#default#urn::xdFileAttachment);}
    .xdPageBreak{display: none;}
    BODY{margin-right:21px;}
    .xdTextBoxRTL{display:inline-block;white-space:nowrap;text-
overflow:ellipsis;;padding:1px;margin:1px;border: 1pt solid
#dcdcdc;color:windowtext;background-color:window;overflow:hidden;text-align:right;word-
wrap:normal;}
    .xdRichTextBoxRTL{display:inline-block;;padding:1px;margin:1px;border: 1pt solid
#dcdcdc;color:windowtext;background-color:window;overflow-x:hidden;word-wrap:break-word;text-
```

```
overflow:ellipsis;text-align:right;font-weight:normal;font-style:normal;text-
decoration:none;vertical-align:baseline;}
    .xdDTTextRTL{height:100%;width:100%;margin-left:22px;overflow:hidden;padding:0px;white-
space:nowrap;}
    .xdDTButtonRTL{margin-right:-21px;height:18px;width:20px;behavior:
url(#default#DTPicker);}
    .xdMultiSelectFillinRTL{display:inline-block;white-space:nowrap;text-
overflow:ellipsis;;padding:1px;margin:1px;border: 1pt solid #dcdcdc;overflow:hidden;text-
align:right;}
    .xdTextBox{display:inline-block;white-space:nowrap;text-
overflow:ellipsis;;padding:1px;margin:1px;border: 1pt solid
#dcdcdc;color:windowtext;background-color:window;overflow:hidden;text-align:left;word-
wrap:normal;}
    .xdRichTextBox{display:inline-block;;padding:1px;margin:1px;border: 1pt solid
#dcdcdc;color:windowtext;background-color:window;overflow-x:hidden;word-wrap:break-word;text-
overflow:ellipsis;text-align:left;font-weight:normal;font-style:normal;text-
decoration:none;vertical-align:baseline;}
    .xdDTPicker{;display:inline;margin:1px;margin-bottom: 2px;border: 1pt solid
#dcdcdc;color:windowtext;background-color:window;overflow:hidden;text-indent:0}
    .xdDTText{height:100%;width:100%;margin-right:22px;overflow:hidden;padding:0px;white-
space:nowrap;}
    .xdDTButton{margin-left:-21px;height:18px;width:20px;behavior: url(#default#DTPicker);}
    .xdRepeatingTable TD {VERTICAL-ALIGN: top;}
</style>
|
<style controlStyle="controlStyle">
BODY{margin-left:21px;color:windowtext;background-color:window;layout-grid:none;}
.xdListItem {display:inline-block;width:100%%;vertical-align:text-top;}
.xdListBox,.xdComboBox{margin:1px;}
.xdInlinePicture{margin:1px; BEHAVIOR: url(#default#urn::xdPicture) }
.xdLinkedPicture{margin:1px; BEHAVIOR: url(#default#urn::xdPicture)
url(#default#urn::controls/Binder) }
.xdSection{border:1pt solid #FFFFFF;margin:6px 0px 6px 0px;padding:1px 1px 1px 5px;}
.xdRepeatingSection{border:1pt solid #FFFFFF;margin:6px 0px 6px 0px;padding:1px 1px 1px 5px;}
.xdBehavior_Formatting {BEHAVIOR: url(#default#urn::controls/Binder)
url(#default#Formatting);}
.xdBehavior_FormattingNoBUI{BEHAVIOR: url(#default#CalPopup)
url(#default#urn::controls/Binder) url(#default#Formatting);}
.xdExpressionBox{margin: 1px;padding:1px;word-wrap: break-word;text-overflow:
ellipsis;overflow-x:hidden;}
.xdBehavior_GhostedText,
.xdBehavior_GhostedTextNoBUI{BEHAVIOR: url(#default#urn::controls/Binder)
url(#default#TextField) url(#default#GhostedText);}
.xdBehavior_GTFormatting{BEHAVIOR: url(#default#urn::controls/Binder)
url(#default#Formatting) url(#default#GhostedText);}
.xdBehavior_GTFormattingNoBUI{BEHAVIOR: url(#default#CalPopup)
url(#default#urn::controls/Binder) url(#default#Formatting) url(#default#GhostedText);}
.xdBehavior_Boolean{BEHAVIOR: url(#default#urn::controls/Binder)
url(#default#BooleanHelper);}
.xdBehavior Select{BEHAVIOR: url(#default#urn::controls/Binder) url(#default#SelectHelper);}
.xdRepeatingTable{BORDER-TOP-STYLE: none; BORDER-RIGHT-STYLE: none; BORDER-LEFT-STYLE: none;
BORDER-BOTTOM-STYLE: none; BORDER-COLLAPSE: collapse; WORD-WRAP: break-word;}
.xdTextBox{display:inline-block;white-space:nowrap;text-
overflow:ellipsis;;padding:1px;margin:1px;border: 1pt solid
#dcdcdc;color:windowtext;background-color:window;overflow:hidden;text-align:left;} /*
 locID css@text-align="left"  locComment="for Arabic and Hebrew SKU, the text-align value
needs to be set to right {L=!1025,1037}{ValidStrings=left,right}" */
.xdRichTextBox{display:inline-block;;padding:1px;margin:1px;border: 1pt solid
#dcdcdc;color:windowtext;background-color:window;overflow-x:hidden;word-wrap:break-word;text-
overflow:ellipsis;text-align:left;font-weight:normal;font-style:normal;text-
decoration:none;vertical-align:baseline;} /* _locID_css@text-align="left" _locComment="for
Arabic and Hebrew SKU, the text-align value needs to be set to right
{L=!1025,1037}{ValidStrings=left,right}" */
.xdDTPicker{;display:inline;margin:1px;margin-bottom: 2px;border: 1pt solid
#dcdcdc;color:windowtext;background-color:window;overflow:hidden;}
.xdDTText{height:100%%;width:100%%;margin-right:22px;overflow:hidden;padding:0px;white-
space:nowrap;}
.xdDTButton{margin-left:-21px;height:18px;width:20px;behavior: url(#default#DTPicker);}
.xdRepeatingTable TD {VERTICAL-ALIGN: top;}</style>
|
```

```
<style controlStyle="controlStyle">
BODY{margin-left:21px;color:windowtext;background-color:window;layout-grid:none;}
.xdListItem {display:inline-block;width:100%%;vertical-align:text-top;}
.xdListBox,.xdComboBox{margin:1px;}
.xdInlinePicture{margin:1px; BEHAVIOR: url(#default#urn::xdPicture) }
.xdLinkedPicture{margin:1px; BEHAVIOR: url(#default#urn::xdPicture)
url(#default#urn::controls/Binder) }
.xdSection{border:1pt solid #FFFFFF;margin:6px 0px 6px 0px;padding:1px 1px 1px 5px;}
.xdRepeatingSection{border:1pt solid #FFFFFF;margin:6px 0px 6px 0px;padding:1px 1px 1px 5px;}
.xdBehavior_Formatting {BEHAVIOR: url(#default#urn::controls/Binder)
url(#default#Formatting);}
.xdBehavior_FormattingNoBUI{BEHAVIOR: url(#default#CalPopup)
url(#default#urn::controls/Binder) url(#default#Formatting);}
.xdExpressionBox{margin: 1px;padding:1px;word-wrap: break-word;text-overflow:
ellipsis;overflow-x:hidden;}
.xdBehavior_GhostedText,.xdBehavior_GhostedTextNoBUI{BEHAVIOR:
url(#default#urn::controls/Binder) url(#default#TextField) url(#default#GhostedText);}
.xdBehavior_GTFormatting{BEHAVIOR: url(#default#urn::controls/Binder)
url(#default#Formatting) url(#default#GhostedText);}
.xdBehavior_GTFormattingNoBUI{BEHAVIOR: url(#default#CalPopup)
url(#default#urn::controls/Binder) url(#default#Formatting) url(#default#GhostedText);}
.xdBehavior_Boolean{BEHAVIOR: url(#default#urn::controls/Binder)
url(#default#BooleanHelper);}
.xdBehavior_Select{BEHAVIOR: url(#default#urn::controls/Binder) url(#default#SelectHelper);}
.xdRepeatingTable{BORDER-TOP-STYLE: none; BORDER-RIGHT-STYLE: none; BORDER-LEFT-STYLE: none;
BORDER-BOTTOM-STYLE: none; BORDER-COLLAPSE: collapse; WORD-WRAP: break-
word;}.xdTextBox{display:inline-block;white-space:nowrap;text-
overflow:ellipsis;;padding:1px;margin:1px;border: 1pt solid
#dcdcdc;color:windowtext;background-color:window;overflow:hidden;text-align:left;}/*
_locID_css@text-align="left" _locComment="for Arabic and Hebrew SKU, the text-align value
needs to be set to right {L=!1025,1037}{ValidStrings=left,right}" */
.xdRichTextBox{display:inline-block;;padding:1px;margin:1px;border: 1pt solid
#dcdcdc;color:windowtext;background-color:window;overflow:hidden;word-wrap:break-word;text-
overflow:ellipsis;text-align:left;font-weight:normal;font-style:normal;text-
decoration:none;vertical-align:baseline;}/* _locID_css@text-align="left" _locComment="for
Arabic and Hebrew SKU, the text-align value needs to be set to right
{L=!1025,1037}{ValidStrings=left,right}" */
.xdDTPicker{;display:inline;margin:1px;margin-bottom: 2px;border: 1pt solid
#dcdcdc;color:windowtext;background-color:window;overflow:hidden;}
.xdDTText{height:100%%;width:100%%;margin-right:22px;overflow:hidden;padding:0px;white-
space:nowrap;}
.xdDTButton{margin-left:-21px;height:18px;width:20px;behavior: url(#default#DTPicker);}
.xdRepeatingTable TD {VERTICAL-ALIGN: top;}</style>
|
<style controlStyle="controlStyle">
BODY{margin-left:21px;color:windowtext;background-color:window;layout-grid:none;}
.xdListItem {display:inline-block;width:100%%;vertical-align:text-top;}
.xdListBox,.xdComboBox{margin:1px;}
.xdInlinePicture{margin:1px; BEHAVIOR: url(#default#urn::xdPicture) }
.xdLinkedPicture{margin:1px; BEHAVIOR: url(#default#urn::xdPicture)
url(#default#urn::controls/Binder) }
.xdSection{border:1pt solid #FFFFFF;margin:6px 0px 6px 0px;padding:1px 1px 1px 5px;}
.xdRepeatingSection{border:1pt solid #FFFFFF;margin:6px 0px 6px 0px;padding:1px 1px 1px 5px;}
.xdBehavior Formatting {BEHAVIOR: url(#default#urn::controls/Binder)
url(#default#Formatting);}
.xdBehavior_FormattingNoBUI{BEHAVIOR: url(#default#CalPopup)
url(#default#urn::controls/Binder) url(#default#Formatting);}
.xdExpressionBox{margin: 1px;padding:1px;word-wrap: break-word;text-overflow:
ellipsis;overflow-x:hidden;}
.xdBehavior_GhostedText,.xdBehavior_GhostedTextNoBUI{BEHAVIOR:
url(#default#urn::controls/Binder) url(#default#TextField) url(#default#GhostedText);}
.xdBehavior_GTFormatting{BEHAVIOR: url(#default#urn::controls/Binder)
url(#default#Formatting) url(#default#GhostedText);}
.xdBehavior_GTFormattingNoBUI{BEHAVIOR: url(#default#CalPopup)
url(#default#urn::controls/Binder) url(#default#Formatting) url(#default#GhostedText);}
.xdBehavior_Boolean{BEHAVIOR: url(#default#urn::controls/Binder)
url(#default#BooleanHelper);}
.xdBehavior_Select{BEHAVIOR: url(#default#urn::controls/Binder) url(#default#SelectHelper);}
```

```
.xdRepeatingTable{BORDER-TOP-STYLE: none; BORDER-RIGHT-STYLE: none; BORDER-LEFT-STYLE: none;
BORDER-BOTTOM-STYLE: none; BORDER-COLLAPSE: collapse; WORD-WRAP: break-
word;}.xdTextBox{display:inline-block;white-space:nowrap;text-
overflow:ellipsis;;padding:1px;margin:1px;border: 1pt solid
#dcdcdc;color:windowtext;background-color:window;overflow:hidden;text-align:left;}/*
 locID css@text-align="right"  locComment="for Arabic and Hebrew SKU, the text-align value
needs to be set to right {L=!1025,1037}{ValidStrings=left,right}" */
.xdRichTextBox{display:inline-block;;padding:1px;margin:1px;border: 1pt solid
#dcdcdc;color:windowtext;background-color:window;overflow:hidden;word-wrap:break-word;text-
overflow:ellipsis;text-align:left;font-weight:normal;font-style:normal;text-
decoration:none;vertical-align:baseline;}/*  locID css@text-align="left"  locComment="for
Arabic and Hebrew SKU, the text-align value needs to be set to right
{L=!1025,1037}{ValidStrings=left,right}" */
.xdDTPicker{;display:inline;margin:1px;margin-bottom: 2px;border: 1pt solid
#dcdcdc;color:windowtext;background-color:window;overflow:hidden;}
.xdDTText{height:100%%;width:100%%;margin-right:22px;overflow:hidden;padding:0px;white-
space:nowrap;}
.xdDTButton{margin-left:-21px;height:18px;width:20px;behavior: url(#default#DTPicker);}
.xdRepeatingTable TD {VERTICAL-ALIGN: top;}</style>
```

TABLE_STYLE yields are associated with a client-only feature and MUST be ignored by the form server.

TABLE_STYLE:

```
<style tableEditor="TableStyleRulesID">
    TABLE.xdLayout TD {BORDER-RIGHT: medium none; BORDER-TOP: medium none; BORDER-LEFT:
medium none; BORDER-BOTTOM: medium none}
    TABLE.msoUcTable TD {BORDER-RIGHT: 1pt solid; BORDER-TOP: 1pt solid; BORDER-LEFT: 1pt
solid; BORDER-BOTTOM: 1pt solid}
    TABLE {BEHAVIOR: url (#default#urn::tables/NDTable)}
</style>
```

LANGUAGE_STYLE yields are associated with a client-only feature and MUST be ignored by the form server.

LANGUAGE_STYLE:

```
<style languageStyle="languageStyle">
    BODY {FONT-SIZE: 10pt; FONT-FAMILY: Verdana}
    TABLE {FONT-SIZE: 10pt; FONT-FAMILY: Verdana}
    SELECT {FONT-SIZE: 10pt; FONT-FAMILY: Verdana}
    .optionalPlaceholder {PADDING-LEFT: 20px; FONT-WEIGHT: normal; FONT-SIZE: xx-small;
BEHAVIOR: url(#default#xOptional); COLOR: #333333; FONT-STYLE: normal; FONT-FAMILY: Verdana;
TEXT-DECORATION: none}
    .langFont {FONT-FAMILY: Verdana}
    .defaultInDocUI {FONT-SIZE: xx-small; FONT-FAMILY: Verdana}
    .optionalPlaceholder {PADDING-RIGHT: 20px}
</style>
|
 <style languageStyle="languageStyle">body, table,
select{font-family:CSS FONT FAMILY;font-size:CSS FONT SIZE}
.optionalPlaceholder{font-family:CSS_FONT_FAMILY;font-size:CSS_FONT_SIZE;color:#333333;font-
weight:normal;font-style:normal;text-decoration:none;padding-
left:20px;BEHAVIOR:url(#default#xOptional)}
.langFont{font-family:CSS_FONT_FAMILY;}
.defaultInDocUI{font-family:CSS_FONT_FAMILY;font-size:CSS_FONT_SIZE;}0
.optionalPlaceholder{padding-right:20px}
</style>
```

CSS_FONT_FAMILY: Values MUST be as specified in [CSS-LEVEL1] section 5.2.2.

CSS_FONT_SIZE: Values MUST be as specified in [CSS-LEVEL1] section 5.2.6.

THEME_STYLE:

```
<style themeStyle="urn:office.microsoft.com:themeBlackWhite">BODY {
COLOR: black; BACKGROUND-COLOR: white
}
TABLE {
BORDER-RIGHT: medium none; BORDER-TOP: medium none; BORDER-LEFT: medium none; BORDER-BOTTOM:
medium none; BORDER-COLLAPSE: collapse
}
TD {
BORDER-LEFT-COLOR: #000000; BORDER-BOTTOM-COLOR: #000000; BORDER-TOP-COLOR: #000000; BORDER-
RIGHT-COLOR: #000000
}
TH {
BORDER-LEFT-COLOR: #000000; BORDER-BOTTOM-COLOR: #000000; COLOR: black; BORDER-TOP-COLOR:
#000000; BACKGROUND-COLOR: #ffffff; BORDER-RIGHT-COLOR: #000000
}
.xdTableHeader {
COLOR: black; BACKGROUND-COLOR: #ffffff
}
P {
MARGIN-TOP: 0px
}
H1 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #000000
}
H2 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #000000
}
H3 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #000000
}
H4 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #000000
}
H5 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #000000
}
H6 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #ffffff
}
.primaryVeryDark {
COLOR: #ffffff; BACKGROUND-COLOR: #000000
}
.primaryDark {
COLOR: white; BACKGROUND-COLOR: #000000
}
.primaryMedium {
COLOR: black; BACKGROUND-COLOR: #ffffff
}
.primaryLight {
COLOR: black; BACKGROUND-COLOR: #ffffff
}
.accentDark {
COLOR: white; BACKGROUND-COLOR: #000000
}
.accentLight {
COLOR: black; BACKGROUND-COLOR: #ffffff
}
|
<style themeStyle="urn:office.microsoft.com:themeGray">BODY {
COLOR: black; BACKGROUND-COLOR: white
}
TABLE {
BORDER-RIGHT: medium none; BORDER-TOP: medium none; BORDER-LEFT: medium none; BORDER-BOTTOM:
medium none; BORDER-COLLAPSE: collapse
}
```

```
TD {
BORDER-LEFT-COLOR: #626262; BORDER-BOTTOM-COLOR: #626262; BORDER-TOP-COLOR: #626262; BORDER-
RIGHT-COLOR: #626262
}
TH {
BORDER-LEFT-COLOR: #626262; BORDER-BOTTOM-COLOR: #626262; COLOR: black; BORDER-TOP-COLOR:
#626262; BACKGROUND-COLOR: #d2d2d2; BORDER-RIGHT-COLOR: #626262
}
.xdTableHeader {
COLOR: black; BACKGROUND-COLOR: #f6f6f6
}
P {
MARGIN-TOP: 0px
}
H1 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #000000
}
H2 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #000000
}
H3 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #000000
}
H4 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #000000
}
H5 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #626262
}
H6 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #f6f6f6
}
.primaryVeryDark {
COLOR: #f6f6f6; BACKGROUND-COLOR: #000000
}
.primaryDark {
COLOR: white; BACKGROUND-COLOR: #626262
}
.primaryMedium {
COLOR: black; BACKGROUND-COLOR: #d2d2d2
}
.primaryLight {
COLOR: black; BACKGROUND-COLOR: #f6f6f6
}
.accentDark {
COLOR: white; BACKGROUND-COLOR: #626262
}
.accentLight {
COLOR: black; BACKGROUND-COLOR: #f6f6f6
}
</style>
|
<style themeStyle="urn:office.microsoft.com:themeSlate">BODY {
COLOR: black; BACKGROUND-COLOR: white
}
TABLE {
BORDER-RIGHT: medium none; BORDER-TOP: medium none; BORDER-LEFT: medium none; BORDER-BOTTOM:
medium none; BORDER-COLLAPSE: collapse
}
TD {
BORDER-LEFT-COLOR: #585867; BORDER-BOTTOM-COLOR: #585867; BORDER-TOP-COLOR: #585867; BORDER-
RIGHT-COLOR: #585867
}
TH {
BORDER-LEFT-COLOR: #585867; BORDER-BOTTOM-COLOR: #585867; COLOR: black; BORDER-TOP-COLOR:
#585867; BACKGROUND-COLOR: #dadae1; BORDER-RIGHT-COLOR: #585867
}
.xdTableHeader {
COLOR: black; BACKGROUND-COLOR: #efeff6
```

```
}
P {
MARGIN-TOP: 0px
}
H1 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #000000
}
H2 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #000000
}
H3 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #000000
}
H4 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #000000
}
H5 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #585867
}
H6 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #efeff6
}
.primaryVeryDark {
COLOR: #efeff6; BACKGROUND-COLOR: #000000
}
.primaryDark {
COLOR: white; BACKGROUND-COLOR: #585867
}
.primaryMedium {
COLOR: black; BACKGROUND-COLOR: #dadae1
}
.primaryLight {
COLOR: black; BACKGROUND-COLOR: #efeff6
}
.accentDark {
COLOR: white; BACKGROUND-COLOR: #585867
}
.accentLight {
COLOR: black; BACKGROUND-COLOR: #efeff6
}
</style>
|
<style themeStyle="urn:office.microsoft.com:themeBurgundy">BODY {
COLOR: black; BACKGROUND-COLOR: white
}
TABLE {
BORDER-RIGHT: medium none; BORDER-TOP: medium none; BORDER-LEFT: medium none; BORDER-BOTTOM:
medium none; BORDER-COLLAPSE: collapse
}
TD {
BORDER-LEFT-COLOR: #ce5764; BORDER-BOTTOM-COLOR: #ce5764; BORDER-TOP-COLOR: #ce5764; BORDER-
RIGHT-COLOR: #ce5764
}
TH {
BORDER-LEFT-COLOR: #ce5764; BORDER-BOTTOM-COLOR: #ce5764; COLOR: black; BORDER-TOP-COLOR:
#ce5764; BACKGROUND-COLOR: #fcc8c7; BORDER-RIGHT-COLOR: #ce5764
}
.xdTableHeader {
COLOR: black; BACKGROUND-COLOR: #fde9ec
}
P {
MARGIN-TOP: 0px
}
H1 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #79194d
}
H2 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #79194d
}
```

```
H3 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #79194d
}
H4 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #79194d
}
H5 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #ce5764
}
H6 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #fde9ec
}
.primaryVeryDark {
COLOR: #fde9ec; BACKGROUND-COLOR: #79194d
}
.primaryDark {
COLOR: white; BACKGROUND-COLOR: #ce5764
}
.primaryMedium {
COLOR: black; BACKGROUND-COLOR: #fcc8c7
}
.primaryLight {
COLOR: black; BACKGROUND-COLOR: #fde9ec
}
.accentDark {
COLOR: white; BACKGROUND-COLOR: #79194d
}
.accentLight {
COLOR: black; BACKGROUND-COLOR: #fde9ec
}
</style>
|
<style themeStyle="urn:office.microsoft.com:themeMahogany">BODY {
COLOR: black; BACKGROUND-COLOR: white
}
TABLE {
BORDER-RIGHT: medium none; BORDER-TOP: medium none; BORDER-LEFT: medium none; BORDER-BOTTOM:
medium none; BORDER-COLLAPSE: collapse
}
TD {
BORDER-LEFT-COLOR: #9b5d2c; BORDER-BOTTOM-COLOR: #9b5d2c; BORDER-TOP-COLOR: #9b5d2c; BORDER-
RIGHT-COLOR: #9b5d2c
}
TH {
BORDER-LEFT-COLOR: #9b5d2c; BORDER-BOTTOM-COLOR: #9b5d2c; COLOR: black; BORDER-TOP-COLOR:
#9b5d2c; BACKGROUND-COLOR: #d3a04f; BORDER-RIGHT-COLOR: #9b5d2c
}
.xdTableHeader {
COLOR: black; BACKGROUND-COLOR: #ffdaab
}
P {
MARGIN-TOP: 0px
}
H1 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #61000a
}
H2 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #61000a
}
H3 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #61000a
}
H4 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #61000a
}
H5 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #9b5d2c
}
H6 {
```

```
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #ffdaab
}
.primaryVeryDark {
COLOR: #ffdaab; BACKGROUND-COLOR: #61000a
}
.primaryDark {
COLOR: white; BACKGROUND-COLOR: #9b5d2c
}
.primaryMedium {
COLOR: black; BACKGROUND-COLOR: #d3a04f
}
.primaryLight {
COLOR: black; BACKGROUND-COLOR: #ffdaab
}
.accentDark {
COLOR: white; BACKGROUND-COLOR: #5c73b6
}
.accentLight {
COLOR: black; BACKGROUND-COLOR: #bfcefa
}
</style>
|
<style themeStyle="urn:office.microsoft.com:themeBrown">BODY {
COLOR: black; BACKGROUND-COLOR: white
}
TABLE {
BORDER-RIGHT: medium none; BORDER-TOP: medium none; BORDER-LEFT: medium none; BORDER-BOTTOM:
medium none; BORDER-COLLAPSE: collapse
}
TD {
BORDER-LEFT-COLOR: #845c42; BORDER-BOTTOM-COLOR: #845c42; BORDER-TOP-COLOR: #845c42; BORDER-
RIGHT-COLOR: #845c42
}
TH {
BORDER-LEFT-COLOR: #845c42; BORDER-BOTTOM-COLOR: #845c42; COLOR: black; BORDER-TOP-COLOR:
#845c42; BACKGROUND-COLOR: #e7d3bd; BORDER-RIGHT-COLOR: #845c42
}
.xdTableHeader {
COLOR: black; BACKGROUND-COLOR: #f8eee0
}
P {
MARGIN-TOP: 0px
}
H1 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #442422
}
H2 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #442422
}
H3 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #442422
}
H4 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #442422
}
H5 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #845c42
}
H6 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #f8eee0
}
.primaryVeryDark {
COLOR: #f8eee0; BACKGROUND-COLOR: #442422
}
.primaryDark {
COLOR: white; BACKGROUND-COLOR: #845c42
}
.primaryMedium {
COLOR: black; BACKGROUND-COLOR: #e7d3bd
```

```
}
.primaryLight {
COLOR: black; BACKGROUND-COLOR: #f8eee0
}
.accentDark {
COLOR: white; BACKGROUND-COLOR: #3757b4
}
.accentLight {
COLOR: black; BACKGROUND-COLOR: #e1eaff
}
</style>
|
<style themeStyle="urn:office.microsoft.com:themeBlue">BODY {
COLOR: black; BACKGROUND-COLOR: white
}
TABLE {
BORDER-RIGHT: medium none; BORDER-TOP: medium none; BORDER-LEFT: medium none; BORDER-BOTTOM:
medium none; BORDER-COLLAPSE: collapse
}
TD {
BORDER-LEFT-COLOR: #517dbf; BORDER-BOTTOM-COLOR: #517dbf; BORDER-TOP-COLOR: #517dbf; BORDER-
RIGHT-COLOR: #517dbf
}
TH {
BORDER-LEFT-COLOR: #517dbf; BORDER-BOTTOM-COLOR: #517dbf; COLOR: black; BORDER-TOP-COLOR:
#517dbf; BACKGROUND-COLOR: #cbd8eb; BORDER-RIGHT-COLOR: #517dbf
}
.xdTableHeader {
COLOR: black; BACKGROUND-COLOR: #ebf0f9
}
P {
MARGIN-TOP: 0px
}
H1 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #1e3c7b
}
H2 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #1e3c7b
}
H3 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #1e3c7b
}
H4 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #1e3c7b
}
H5 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #517dbf
}
H6 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #ebf0f9
}
.primaryVeryDark {
COLOR: #ebf0f9; BACKGROUND-COLOR: #1e3c7b
}
.primaryDark {
COLOR: white; BACKGROUND-COLOR: #517dbf
}
.primaryMedium {
COLOR: black; BACKGROUND-COLOR: #cbd8eb
}
.primaryLight {
COLOR: black; BACKGROUND-COLOR: #ebf0f9
}
.accentDark {
COLOR: white; BACKGROUND-COLOR: #517dbf
}
.accentLight {
COLOR: black; BACKGROUND-COLOR: #ebf0f9
}
```

```
</style>
|
<style themeStyle="urn:office.microsoft.com:themeBlueberry">BODY {
COLOR: black; BACKGROUND-COLOR: white
}
TABLE {
BORDER-RIGHT: medium none; BORDER-TOP: medium none; BORDER-LEFT: medium none; BORDER-BOTTOM:
medium none; BORDER-COLLAPSE: collapse
}
TD {
BORDER-LEFT-COLOR: #637595; BORDER-BOTTOM-COLOR: #637595; BORDER-TOP-COLOR: #637595; BORDER-
RIGHT-COLOR: #637595
}
TH {
BORDER-LEFT-COLOR: #637595; BORDER-BOTTOM-COLOR: #637595; COLOR: black; BORDER-TOP-COLOR:
#637595; BACKGROUND-COLOR: #bbc9dc; BORDER-RIGHT-COLOR: #637595
}
.xdTableHeader {
COLOR: black; BACKGROUND-COLOR: #ede6ef
}
P {
MARGIN-TOP: 0px
}
H1 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #183569
}
H2 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #183569
}
H3 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #183569
}
H4 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #183569
}
H5 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #637595
}
H6 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #ede6ef
}
.primaryVeryDark {
COLOR: #ede6ef; BACKGROUND-COLOR: #183569
}
.primaryDark {
COLOR: white; BACKGROUND-COLOR: #637595
}
.primaryMedium {
COLOR: black; BACKGROUND-COLOR: #bbc9dc
}
.primaryLight {
COLOR: black; BACKGROUND-COLOR: #ede6ef
}
.accentDark {
COLOR: white; BACKGROUND-COLOR: #637595
}
.accentLight {
COLOR: black; BACKGROUND-COLOR: #ede6ef
}
</style>
|
<style themeStyle="urn:office.microsoft.com:themeBrightBlue">BODY {
COLOR: black; BACKGROUND-COLOR: white
}
TABLE {
BORDER-RIGHT: medium none; BORDER-TOP: medium none; BORDER-LEFT: medium none; BORDER-BOTTOM:
medium none; BORDER-COLLAPSE: collapse
}
TD {
```

```
BORDER-LEFT-COLOR: #408ce8; BORDER-BOTTOM-COLOR: #408ce8; BORDER-TOP-COLOR: #408ce8; BORDER-
RIGHT-COLOR: #408ce8
}
TH {
BORDER-LEFT-COLOR: #408ce8; BORDER-BOTTOM-COLOR: #408ce8; COLOR: black; BORDER-TOP-COLOR:
#408ce8; BACKGROUND-COLOR: #d3e5fa; BORDER-RIGHT-COLOR: #408ce8
}
.xdTableHeader {
COLOR: black; BACKGROUND-COLOR: #f5f3eb
}
P {
MARGIN-TOP: 0px
}
H1 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #134fc7
}
H2 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #134fc7
}
H3 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #134fc7
}
H4 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #134fc7
}
H5 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #408ce8
}
H6 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #f5f3eb
}
.primaryVeryDark {
COLOR: #f5f3eb; BACKGROUND-COLOR: #134fc7
}
.primaryDark {
COLOR: white; BACKGROUND-COLOR: #408ce8
}
.primaryMedium {
COLOR: black; BACKGROUND-COLOR: #d3e5fa
}
.primaryLight {
COLOR: black; BACKGROUND-COLOR: #f5f3eb
}
.accentDark {
COLOR: white; BACKGROUND-COLOR: #ff8716
}
.accentLight {
COLOR: black; BACKGROUND-COLOR: #ffd991
}
</style>
|
<style themeStyle="urn:office.microsoft.com:themeTurquoise">BODY {
COLOR: black; BACKGROUND-COLOR: white
}
TABLE {
BORDER-RIGHT: medium none; BORDER-TOP: medium none; BORDER-LEFT: medium none; BORDER-BOTTOM:
medium none; BORDER-COLLAPSE: collapse
}
TD {
BORDER-LEFT-COLOR: #2ea8e7; BORDER-BOTTOM-COLOR: #2ea8e7; BORDER-TOP-COLOR: #2ea8e7; BORDER-
RIGHT-COLOR: #2ea8e7
}
TH {
BORDER-LEFT-COLOR: #2ea8e7; BORDER-BOTTOM-COLOR: #2ea8e7; COLOR: black; BORDER-TOP-COLOR:
#2ea8e7; BACKGROUND-COLOR: #a4e0ff; BORDER-RIGHT-COLOR: #2ea8e7
}
.xdTableHeader {
COLOR: black; BACKGROUND-COLOR: #fff4c7
}
```

```
P {
MARGIN-TOP: 0px
}
H1 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #7673fd
}
H2 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #7673fd
}
H3 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #7673fd
}
H4 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #7673fd
}
H5 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #2ea8e7
}
H6 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #fff4c7
}
.primaryVeryDark {
COLOR: #fff4c7; BACKGROUND-COLOR: #7673fd
}
.primaryDark {
COLOR: white; BACKGROUND-COLOR: #2ea8e7
}
.primaryMedium {
COLOR: black; BACKGROUND-COLOR: #a4e0ff
}
.primaryLight {
COLOR: black; BACKGROUND-COLOR: #fff4c7
}
.accentDark {
COLOR: white; BACKGROUND-COLOR: #2ea8e7
}
.accentLight {
COLOR: black; BACKGROUND-COLOR: #fff4c7
}
</style>
|
<style themeStyle="urn:office.microsoft.com:themeGreen">BODY {
COLOR: black; BACKGROUND-COLOR: white
}
TABLE {
BORDER-RIGHT: medium none; BORDER-TOP: medium none; BORDER-LEFT: medium none; BORDER-BOTTOM:
medium none; BORDER-COLLAPSE: collapse
}
TD {
BORDER-LEFT-COLOR: #489659; BORDER-BOTTOM-COLOR: #489659; BORDER-TOP-COLOR: #489659; BORDER-
RIGHT-COLOR: #489659
}
TH {
BORDER-LEFT-COLOR: #489659; BORDER-BOTTOM-COLOR: #489659; COLOR: black; BORDER-TOP-COLOR:
#489659; BACKGROUND-COLOR: #d6eace; BORDER-RIGHT-COLOR: #489659
}
.xdTableHeader {
COLOR: black; BACKGROUND-COLOR: #f4fbee
}
P {
MARGIN-TOP: 0px
}
H1 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #035647
}
H2 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #035647
}
H3 {
```

```
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #035647
}
H4 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #035647
}
H5 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #489659
}
H6 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #f4fbee
}
.primaryVeryDark {
COLOR: #f4fbee; BACKGROUND-COLOR: #035647
}
.primaryDark {
COLOR: white; BACKGROUND-COLOR: #489659
}
.primaryMedium {
COLOR: black; BACKGROUND-COLOR: #d6eace
}
.primaryLight {
COLOR: black; BACKGROUND-COLOR: #f4fbee
}
.accentDark {
COLOR: white; BACKGROUND-COLOR: #9b58ba
}
.accentLight {
COLOR: black; BACKGROUND-COLOR: #f1e9ff
}
</style>
|
<style themeStyle="urn:office.microsoft.com:themeOlive">BODY {
COLOR: black; BACKGROUND-COLOR: white
}
TABLE {
BORDER-RIGHT: medium none; BORDER-TOP: medium none; BORDER-LEFT: medium none; BORDER-BOTTOM:
medium none; BORDER-COLLAPSE: collapse
}
TD {
BORDER-LEFT-COLOR: #7c925d; BORDER-BOTTOM-COLOR: #7c925d; BORDER-TOP-COLOR: #7c925d; BORDER-
RIGHT-COLOR: #7c925d
}
TH {
BORDER-LEFT-COLOR: #7c925d; BORDER-BOTTOM-COLOR: #7c925d; COLOR: black; BORDER-TOP-COLOR:
#7c925d; BACKGROUND-COLOR: #d6e0c3; BORDER-RIGHT-COLOR: #7c925d
}
.xdTableHeader {
COLOR: black; BACKGROUND-COLOR: #f5f3eb
}
P {
MARGIN-TOP: 0px
}
H1 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #545f38
}
H2 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #545f38
}
H3 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #545f38
}
H4 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #545f38
}
H5 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #7c925d
}
H6 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #f5f3eb
```

```
}
.primaryVeryDark {
COLOR: #f5f3eb; BACKGROUND-COLOR: #545f38
}
.primaryDark {
COLOR: white; BACKGROUND-COLOR: #7c925d
}
.primaryMedium {
COLOR: black; BACKGROUND-COLOR: #d6e0c3
}
.primaryLight {
COLOR: black; BACKGROUND-COLOR: #f5f3eb
}
.accentDark {
COLOR: white; BACKGROUND-COLOR: #7c925d
}
.accentLight {
COLOR: black; BACKGROUND-COLOR: #f5f3eb
}
</style>
|
<style themeStyle="urn:office.microsoft.com:themeAqua">BODY {
COLOR: black; BACKGROUND-COLOR: white
}
TABLE {
BORDER-RIGHT: medium none; BORDER-TOP: medium none; BORDER-LEFT: medium none; BORDER-BOTTOM:
medium none; BORDER-COLLAPSE: collapse
}
TD {
BORDER-LEFT-COLOR: #17889c; BORDER-BOTTOM-COLOR: #17889c; BORDER-TOP-COLOR: #17889c; BORDER-
RIGHT-COLOR: #17889c
}
TH {
BORDER-LEFT-COLOR: #17889c; BORDER-BOTTOM-COLOR: #17889c; COLOR: black; BORDER-TOP-COLOR:
#17889c; BACKGROUND-COLOR: #d2deed; BORDER-RIGHT-COLOR: #17889c
}
.xdTableHeader {
COLOR: black; BACKGROUND-COLOR: #eaeef4
}
P {
MARGIN-TOP: 0px
}
H1 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #046a7c
}
H2 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #046a7c
}
H3 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #046a7c
}
H4 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #046a7c
}
H5 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #17889c
}
H6 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #eaeef4
}
.primaryVeryDark {
COLOR: #eaeef4; BACKGROUND-COLOR: #046a7c
}
.primaryDark {
COLOR: white; BACKGROUND-COLOR: #17889c
}
.primaryMedium {
COLOR: black; BACKGROUND-COLOR: #d2deed
}
```

```
.primaryLight {
COLOR: black; BACKGROUND-COLOR: #eaeef4
}
.accentDark {
COLOR: white; BACKGROUND-COLOR: #da7b00
}
.accentLight {
COLOR: black; BACKGROUND-COLOR: #fedc91
}
</style>
|
<style themeStyle="urn:office.microsoft.com:themeRed">BODY {
COLOR: black; BACKGROUND-COLOR: white
}
TABLE {
BORDER-RIGHT: medium none; BORDER-TOP: medium none; BORDER-LEFT: medium none; BORDER-BOTTOM:
medium none; BORDER-COLLAPSE: collapse
}
TD {
BORDER-LEFT-COLOR: #f61208; BORDER-BOTTOM-COLOR: #f61208; BORDER-TOP-COLOR: #f61208; BORDER-
RIGHT-COLOR: #f61208
}
TH {
BORDER-LEFT-COLOR: #f61208; BORDER-BOTTOM-COLOR: #f61208; COLOR: black; BORDER-TOP-COLOR:
#f61208; BACKGROUND-COLOR: #fed3d1; BORDER-RIGHT-COLOR: #f61208
}
.xdTableHeader {
COLOR: black; BACKGROUND-COLOR: #f8eee0
}
P {
MARGIN-TOP: 0px
}
H1 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #c00b02
}
H2 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #c00b02
}
H3 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #c00b02
}
H4 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #c00b02
}
H5 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #f61208
}
H6 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #f8eee0
}
.primaryVeryDark {
COLOR: #f8eee0; BACKGROUND-COLOR: #c00b02
}
.primaryDark {
COLOR: white; BACKGROUND-COLOR: #f61208
}
.primaryMedium {
COLOR: black; BACKGROUND-COLOR: #fed3d1
}
.primaryLight {
COLOR: black; BACKGROUND-COLOR: #f8eee0
}
.accentDark {
COLOR: white; BACKGROUND-COLOR: #f61208
}
.accentLight {
COLOR: black; BACKGROUND-COLOR: #f8eee0
}
</style>
```

```
|
<style themeStyle="urn:office.microsoft.com:themeOrange">BODY {
COLOR: black; BACKGROUND-COLOR: white
}
TABLE {
BORDER-RIGHT: medium none; BORDER-TOP: medium none; BORDER-LEFT: medium none; BORDER-BOTTOM:
medium none; BORDER-COLLAPSE: collapse
}
TD {
BORDER-LEFT-COLOR: #fda102; BORDER-BOTTOM-COLOR: #fda102; BORDER-TOP-COLOR: #fda102; BORDER-
RIGHT-COLOR: #fda102
}
TH {
BORDER-LEFT-COLOR: #fda102; BORDER-BOTTOM-COLOR: #fda102; COLOR: black; BORDER-TOP-COLOR:
#fda102; BACKGROUND-COLOR: #fedc8e; BORDER-RIGHT-COLOR: #fda102
}
.xdTableHeader {
COLOR: black; BACKGROUND-COLOR: #fff7e7
}
P {
MARGIN-TOP: 0px
}
H1 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #f56116
}
H2 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #f56116
}
H3 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #f56116
}
H4 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #f56116
}
H5 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #fda102
}
H6 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #fff7e7
}
.primaryVeryDark {
COLOR: #fff7e7; BACKGROUND-COLOR: #f56116
}
.primaryDark {
COLOR: white; BACKGROUND-COLOR: #fda102
}
.primaryMedium {
COLOR: black; BACKGROUND-COLOR: #fedc8e
}
.primaryLight {
COLOR: black; BACKGROUND-COLOR: #fff7e7
}
.accentDark {
COLOR: white; BACKGROUND-COLOR: #fda102
}
.accentLight {
COLOR: black; BACKGROUND-COLOR: #fff7e7
}
</style>
|
<style themeStyle="urn:office.microsoft.com:themePurpleSage">BODY {
COLOR: black; BACKGROUND-COLOR: white
}
TABLE {
BORDER-RIGHT: medium none; BORDER-TOP: medium none; BORDER-LEFT: medium none; BORDER-BOTTOM:
medium none; BORDER-COLLAPSE: collapse
}
TD {
```

```
BORDER-LEFT-COLOR: #87978b; BORDER-BOTTOM-COLOR: #87978b; BORDER-TOP-COLOR: #87978b; BORDER-
RIGHT-COLOR: #87978b
}
TH {
BORDER-LEFT-COLOR: #87978b; BORDER-BOTTOM-COLOR: #87978b; COLOR: black; BORDER-TOP-COLOR:
#87978b; BACKGROUND-COLOR: #d7e1d9; BORDER-RIGHT-COLOR: #87978b
}
.xdTableHeader {
COLOR: black; BACKGROUND-COLOR: #f0f1fb
}
P {
MARGIN-TOP: 0px
}
H1 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #665484
}
H2 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #665484
}
H3 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #665484
}
H4 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #665484
}
H5 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #87978b
}
H6 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #f0f1fb
}
.primaryVeryDark {
COLOR: #f0f1fb; BACKGROUND-COLOR: #665484
}
.primaryDark {
COLOR: white; BACKGROUND-COLOR: #87978b
}
.primaryMedium {
COLOR: black; BACKGROUND-COLOR: #d7e1d9
}
.primaryLight {
COLOR: black; BACKGROUND-COLOR: #f0f1fb
}
.accentDark {
COLOR: white; BACKGROUND-COLOR: #87978b
}
.accentLight {
COLOR: black; BACKGROUND-COLOR: #d7e1d9
}
</style>
|
<style themeStyle="urn:office.microsoft.com:themePlum">BODY {
COLOR: black; BACKGROUND-COLOR: white
}
TABLE {
BORDER-RIGHT: medium none; BORDER-TOP: medium none; BORDER-LEFT: medium none; BORDER-BOTTOM:
medium none; BORDER-COLLAPSE: collapse
}
TD {
BORDER-LEFT-COLOR: #845c42; BORDER-BOTTOM-COLOR: #845c42; BORDER-TOP-COLOR: #845c42; BORDER-
RIGHT-COLOR: #845c42
}
TH {
BORDER-LEFT-COLOR: #845c42; BORDER-BOTTOM-COLOR: #845c42; COLOR: black; BORDER-TOP-COLOR:
#845c42; BACKGROUND-COLOR: #dabe9b; BORDER-RIGHT-COLOR: #845c42
}
.xdTableHeader {
COLOR: black; BACKGROUND-COLOR: #fbf3de
}
```

```
P {
MARGIN-TOP: 0px
}
H1 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #3e2244
}
H2 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #3e2244
}
H3 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #3e2244
}
H4 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #3e2244
}
H5 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #845c42
}
H6 {
MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; COLOR: #fbf3de
}
.primaryVeryDark {
COLOR: #fbf3de; BACKGROUND-COLOR: #3e2244
}
.primaryDark {
COLOR: white; BACKGROUND-COLOR: #845c42
}
.primaryMedium {
COLOR: black; BACKGROUND-COLOR: #dabe9b
}
.primaryLight {
COLOR: black; BACKGROUND-COLOR: #fbf3de
}
.accentDark {
COLOR: white; BACKGROUND-COLOR: #845c42
}
.accentLight {
COLOR: black; BACKGROUND-COLOR: #fbf3de
}
</style>
```

2.4.1.4 Control Data Formatting

This section specifies the rules that MUST be used for formatting data in controls.

DATA_FMT_LOCALE_VAL: MUST be an LCID.

DATA_FMT_LOCALE: locale:DATA_FMT_LOCALE_VAL

DATA_FMT_NUM_DIGITS: [0-9] | auto. See **numDigits** in section 2.4.2.11.

DATA_FMT_GROUPING: -1 | [0-9] | 32. See **grouping** in section 2.4.2.11.

DATA_FMT_DECIMAL_SEP: . | , | space_char. See **decimalSep** in section 2.4.2.11.

DATA_FMT_THOUSAND_SEP: . | , | space_char. See **thousandSep** in section 2.4.2.11.

DATA_FMT_NEG_ORDER: See **negativeOrder** in section 2.4.2.11.

DATA_FMT_POS_ORDER: See **positiveOrder** in section 2.4.2.11.

DATA_FMT_CUR_LOCALE: currencyLocale:DATA_FMT_LOCALE_VAL

DATA_FMT_DATE_FORMAT_CUSTOM: See **dateFormat** in section 2.4.2.11.

DATA_FMT_DATE_FORMAT: Short Date | Long Date | Year Month | none | DATA_FMT_DATE_FORMAT_CUSTOM

DATA_FMT_ALT_CAL: 0 | 1. See **useAltCalendar** in section 2.4.2.11.

DATA_FMT_EN_STR: 0 | 1. See **englishStringOnly** in section 2.4.2.11.

DATA_FMT_TIME_FORMAT_CUSTOM: See **timeFormat** in section 2.4.2.11.

DATA_FMT_TIME_FORMAT: Short Time | Long Time | none | DATA_FMT_TIME_FORMAT_CUSTOM

DATA_FMT_NOSECONDS: 0 | 1. See **noSeconds** in section 2.4.2.11.

DATA_FMT_CAT_STRING: "string", "plainMutiline"

DATA_FMT_CAT_PERCENTAGE: "percentage","semicolon delimited list of **(DATA_FMT_LOCALE**?, **DATA_FMT_NUM_DIGITS**, **DATA_FMT_GROUPING**?, **DATA_FMT_DECIMAL_SEP**?, **DATA_FMT_THOUSAND_SEP**?, **DATA_FMT_NEG_ORDER**)"

DATA_FMT_CAT_NUMBER: "number","semicolon delimited list of (**DATA_FMT_LOCALE**?, **DATA_FMT_NUM_DIGITS**, **DATA_FMT_GROUPING**?, **DATA_FMT_DECIMAL_SEP**?, **DATA_FMT_THOUSAND_SEP**?, **DATA_FMT_NEG_ORDER**)"

DATA_FMT_CAT_DATETIME: "datetime","semicolon delimited list of (**DATA_FMT_LOCALE**?, **DATA_FMT_DATE_FORMAT**, **DATA_FMT_ALT_CAL**?, **DATA_FMT_EN_STR**?, **DATA_FMT_TIME_FORMAT**, **DATA_FMT_NOSECONDS**?)"

DATA_FMT_CAT_DATE: "date","semicolon delimited list of (**DATA_FMT_LOCALE**?, **DATA_FMT_DATE_FORMAT**, **DATA_FMT_ALT_CAL**?, **DATA_FMT_EN_STR**?)"

DATA_FMT_CAT_TIME: "time","semicolon delimited list of (**DATA_FMT_LOCALE**?, **DATA_FMT_TIME_FORMAT**, **DATA_FMT_NOSECONDS**?)"

DATA_FMT_CTRL_DATE_PICKER: DATA_FMT_CAT_DATE or **DATA_FMT_CAT_DATETIME**.

DATA_FMT_CTRL_EXPBOX: DATA_FMT_CAT_TIME or **DATA_FMT_CAT_DATE** or **DATA_FMT_CAT_DATETIME** or **DATA_FMT_CAT_NUMBER** or **DATA_FMT_CAT_PERCENTAGE**.

DATA_FMT_CTRL_TEXTBOX: DATA_FMT_CTRL_EXPBOX

2.4.1.5 Button Control

The button control is an unbound control that executes actions, rules, or custom code when clicked. A **BUTTON** MUST have one of the symbols in the following table.

Symbol	Description
BUTTON_RULES_AND_CUSTOM_CODE	The button executes rules (1) and custom code when clicked.
BUTTON_RULES_AND_CUSTOM_CODE_WITH_CONDITIONAL_FORMATTING	The button executes rules (1) and custom code when clicked, and supports conditional formatting.
BUTTON_RULES_AND_CUSTOM_CODE_WITH_DYNAMIC_DISPLAY_NAME_AND_CONDITIONAL_FORMATTING	The button executes rules (1) and custom code when clicked, renders a dynamic display name, and supports conditional formatting.
BUTTON_UPDATE_FORM_WITH_CONDITIONAL_FORMATTING	The button updates the form content when clicked, and supports conditional formatting.
BUTTON_UPDATE_FORM_WITH_DYNAMIC_DISPLAY_NAME_AND_CONDITIONAL_FORMATTING	The button updates the form (1) content when clicked, renders a dynamic display name, and supports conditional formatting.

Symbol	Description
BUTTON_ACTION	The button executes actions (submit, query, new, and refresh) when clicked.
BUTTON_ACTION_WITH_CONDITIONAL_FORMATTING	The button executes actions (submit, query, new, and refresh) when clicked, and supports conditional formatting.
BUTTON_ACTION_WITH_DYNAMIC_DISPLAY_NAME_AND_CONDITIONAL_FORMATTING	The button executes actions (submit, query, new, and refresh) when clicked, renders a dynamic display name, and supports conditional formatting.

BUTTON_ACTION_TYPE: MUST be one of the following:

- submit
- query
- new
- refresh

BUTTON_STYLE: Semicolon-delimited list of (**STYLE_SIZE**?, **STYLE_MARGIN**?, **STYLE_PADDING**?, **STYLE_TEXT_DECORATION**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_BORDER**?, **STYLE_FONT**?, **STYLE_COLOR**?, **STYLE_VERTICAL_ALIGN**?).

BUTTON_ACTION_STYLE: Semicolon-delimited list of (**BEHAVIOR**: url(#default#ActionButton), **BUTTON_STYLE**).

BUTTON_CONDITIONAL_FORMATTING_STYLE: Semicolon-delimited list of (**STYLE_TEXT_DECORATION**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_FONT**?, **STYLE_COLOR**?)

BUTTON_CONDITIONAL_FORMATTING:

```
(<xsl:attribute name="style">BUTTON_STYLE?<xsl:choose>
        (<xsl:when test="BOOLEAN XPATH EXPRESSION">STYLE DISPLAY NONE</xsl:when>|
        <xsl:when test="BOOLEAN_XPATH_EXPRESSION"/>|
        <xsl:when
test="BOOLEAN_XPATH_EXPRESSION">BUTTON_CONDITIONAL_FORMATTING_STYLE</xsl:when>)+
    </xsl:choose>
</xsl:attribute>)?
(<xsl:choose>
    (<xsl:when test="BOOLEAN_XPATH_EXPRESSION"/>|
    <xsl:when test="BOOLEAN XPATH EXPRESSION">
        <xsl:attribute name="disabled">true</xsl:attribute>
    </xsl:when>)+
</xsl:choose>)?

BUTTON ACTION CONDITIONAL FORMATTING ::=
(<xsl:attribute name="style">BUTTON_ACTION_STYLE<xsl:choose>
        (<xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_DISPLAY_NONE</xsl:when>|
        <xsl:when test="BOOLEAN_XPATH_EXPRESSION"/>|
        <xsl:when
test="BOOLEAN_XPATH_EXPRESSION">BUTTON_CONDITIONAL_FORMATTING_STYLE</xsl:when>)+
    </xsl:choose>
</xsl:attribute>)?
(<xsl:choose>
    (<xsl:when test="BOOLEAN_XPATH_EXPRESSION"/>|
    <xsl:when test="BOOLEAN XPATH EXPRESSION">
        <xsl:attribute name="disabled">true</xsl:attribute>
    </xsl:when>)+
</xsl:choose>)?

BUTTON_UPDATE_FORM_CONDITIONAL_FORMATTING ::=
```

```
<xsl:attribute name="style">BUTTON_ACTION_STYLE<xsl:choose>
        <xsl:when test="not(xdEnvironment:IsBrowser())">STYLE_DISPLAY_NONE</xsl:when>
        (<xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_DISPLAY_NONE</xsl:when>|
        <xsl:when test="BOOLEAN_XPATH_EXPRESSION"/>|
        <xsl:when
test="BOOLEAN XPATH EXPRESSION">BUTTON CONDITIONAL FORMATTING STYLE</xsl:when>)*
    </xsl:choose>
</xsl:attribute>
(<xsl:choose>
    <xsl:when test="not(xdEnvironment:IsBrowser())"/>
    (<xsl:when test="BOOLEAN XPATH EXPRESSION"/>|
    <xsl:when test="BOOLEAN_XPATH_EXPRESSION">
        <xsl:attribute name="disabled">true</xsl:attribute>
    </xsl:when>)+
</xsl:choose>)?
```

BUTTON_RULES_AND_CUSTOM_CODE:

```
<input class="langFont" title="ANY_STRING" type="button" (value="NON_EMPTY_STRING")?
xd:xctname="Button" xd:CtrlId="CONTROL ID" (xd:auxDom="AUX DOM SOURCE NAME")?
(tabIndex="TAB_INDEX")? (style="BUTTON_STYLE")? (xd:postbackModel="BUTTON_POSTBACKMODEL")?
(accessKey="SINGLE_CHARACTER")? (size="SIZE")?/>
```

BUTTON_RULES_AND_CUSTOM_CODE_WITH_CONDITIONAL_FORMATTING:

```
<input class="langFont" title="ANY_STRING" type="button" (value="NON_EMPTY_STRING")?
xd:xctname="Button" xd:CtrlId="CONTROL_ID" (xd:auxDom="AUX_DOM_SOURCE_NAME")?
(tabIndex="TAB INDEX")? (style="BUTTON STYLE")? (xd:postbackModel="BUTTON POSTBACKMODEL")?
(accessKey="SINGLE_CHARACTER")? (size="SIZE")?>
    BUTTON_CONDITIONAL_FORMATTING
</input>
```

BUTTON_RULES_AND_CUSTOM_CODE_WITH_DYNAMIC_DISPLAY_NAME_AND_CONDITIONAL_FORMATTING:

```
<input class="langFont" title="ANY STRING" type="button" xd:xctname="Button"
xd:CtrlId="CONTROL_ID" (xd:auxDom="AUX_DOM_SOURCE_NAME")? (tabIndex="TAB_INDEX")?
(style="BUTTON_STYLE")? (xd:postbackModel="BUTTON_POSTBACKMODEL")?
(accessKey="SINGLE CHARACTER")? (size="SIZE")?>
    BUTTON_CONDITIONAL_FORMATTING
    CHECK FOR GETDOM BEGIN1
    <xsl:attribute name="value">
        <xsl:value-of select="STRING_XPATH_EXPRESSION"/>
    </xsl:attribute>
    CHECK FOR GETDOM END1
</input>
```

BUTTON_UPDATE_FORM_WITH_CONDITIONAL_FORMATTING:

```
<input class="langFont" title="ANY STRING" type="button" value="NON EMPTY STRING"
xd:xctname="Button" xd:CtrlId="CONTROL_ID" xd:action="updateForm"
(xd:auxDom="AUX_DOM_SOURCE_NAME")? (tabIndex="TAB_INDEX")? (style="BUTTON_ACTION_STYLE")?
(xd:postbackModel="BUTTON_POSTBACKMODEL")? (accessKey="SINGLE_CHARACTER")? (size="SIZE")?>
    BUTTON UPDATE FORM CONDITIONAL FORMATTING
</input>

BUTTON_UPDATE_FORM_WITH_DYNAMIC_DISPLAY_NAME_AND_CONDITIONAL_FORMATTING ::=
<input class="langFont" title="ANY_STRING" type="button" xd:xctname="Button"
xd:CtrlId="CONTROL_ID" xd:action="updateForm" (xd:auxDom="AUX_DOM_SOURCE_NAME")?
```

```
(tabIndex="TAB_INDEX")? (style="BUTTON_ACTION_STYLE")?
(xd:postbackModel="BUTTON POSTBACKMODEL")? (accessKey="SINGLE CHARACTER")? (size="SIZE")?>
    BUTTON UPDATE FORM CONDITIONAL FORMATTING
    CHECK FOR GETDOM BEGIN1
    <xsl:attribute name="value">
        <xsl:value-of select="STRING_XPATH_EXPRESSION"/>
    </xsl:attribute>
    CHECK FOR GETDOM END1
</input>
```

BUTTON_ACTION:

```
<input class="langFont" title="ANY STRING" style="BUTTON ACTION STYLE" type="button"
(value="NON_EMPTY_STRING")? xd:xctname="Button" xd:CtrlId="CONTROL_ID"
(xd:action="BUTTON_ACTION_TYPE")? (xd:auxDom="AUX_DOM_SOURCE_NAME")? (tabIndex="TAB_INDEX")?
(xd:postbackModel="BUTTON_POSTBACKMODEL")? (accessKey="SINGLE_CHARACTER")? (size="SIZE")?/>
```

BUTTON_ACTION_WITH_CONDITIONAL_FORMATTING:

```
<input class="langFont" title="ANY STRING" type="button" (value="NON EMPTY STRING")?
xd:xctname="Button" xd:CtrlId="CONTROL_ID" (xd:action="BUTTON_ACTION_TYPE")?
(xd:auxDom="AUX_DOM_SOURCE_NAME")? (tabIndex="TAB_INDEX")? (style="BUTTON_ACTION_STYLE")?
(xd:postbackModel="BUTTON_POSTBACKMODEL")? (accessKey="SINGLE_CHARACTER")? (size="SIZE")?>
    BUTTON_ACTION_CONDITIONAL_FORMATTING
</input>
```

BUTTON_ACTION_WITH_DYNAMIC_DISPLAY_NAME_AND_CONDITIONAL_FORMATTING:

```
<input class="langFont" title="ANY_STRING" type="button" xd:xctname="Button"
xd:CtrlId="CONTROL_ID" xd:action="BUTTON_ACTION_TYPE" (xd:auxDom="AUX_DOM_SOURCE_NAME")?
(tabIndex="TAB_INDEX")? (style="BUTTON_ACTION_STYLE")?
(xd:postbackModel="BUTTON POSTBACKMODEL")? (accessKey="SINGLE CHARACTER")? (size="SIZE")?>
    BUTTON ACTION CONDITIONAL FORMATTING
    CHECK FOR GETDOM BEGIN1
    <xsl:attribute name="value">
        <xsl:value-of select="STRING_XPATH_EXPRESSION"/>
    </xsl:attribute>
    CHECK FOR GETDOM END1
</input>
```

Following are the control-specific attributes used by the button control:

- xd:action (section 2.4.2.1)
- xd:auxDom (section 2.4.2.4)
- xd:CtrlId (section 2.4.2.10)
- xd:postbackModel (section 2.4.2.29)
- xd:xctname (section 2.4.2.35)

The **xdEnvironment:IsBrowser** XSL function extension, as specified in section 2.4.3.3.1, is used by the button control.

2.4.1.6 Check Box Control

A check box control is a bi-state leaf control that has a value when it is checked, and a different value when it is not checked. A **CHECK_BOX** MUST have one of the symbols in the following table.

Symbol	Description
SIMPLE_CHECK_BOX	A control that has two states, **checked** and **unchecked**. The **unchecked** state tends to be represented as a blank white square, and the **checked** state has a mark, which is commonly a check mark, contained in the white square.
CHECK_BOX_WITH_CONDITIONAL_FORMATTING	Similar to **SIMPLE_CHECK_BOX**, with the addition that the control can be disabled conditionally. A disabled checkbox does not allow the user to directly toggle the control between its two states.

SIMPLE_CHECK_BOX:

```
<input class="xdBehavior_Boolean" title="ANY_STRING1" type="checkbox"
(accessKey="SINGLE_CHARACTER")? xd:binding="LEAF_XPATH1" xd:boundProp="xd:value"
(CHECK_BOX_SINGLE_VALUE | CHECK_BOX_BOTH_VALUES) (tabIndex="TAB_INDEX")?
xd:xctname="CheckBox" xd:CtrlId="CONTROL_ID" (xd:postbackModel="POSTBACKMODEL")?
(style="CHECK BOX STYLE")?>
    CHECK_FOR_GETDOM_BEGIN1
    <xsl:attribute name="xd:value">
        <xsl:value-of select="LEAF_XPATH1" />
    </xsl:attribute>
    <xsl:if test="BOOLEAN XPATH EXPRESSION">
        <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
    </xsl:if>
    CHECK_FOR_GETDOM_END1
</input>
(ANY_STRING2)?

CHECK_BOX_WITH_CONDITIONAL_FORMATTING ::=
<input class="xdBehavior_Boolean" title="ANY_STRING1" type="checkbox"
(accessKey="SINGLE_CHARACTER")? xd:binding="LEAF_XPATH1" xd:boundProp="xd:value"
(CHECK_BOX_SINGLE_VALUE | CHECK_BOX_BOTH_VALUES) (tabIndex="TAB_INDEX")?
xd:xctname="CheckBox" xd:CtrlId="CONTROL_ID" (xd:postbackModel="POSTBACKMODEL")?
(style="CHECK BOX STYLE")?>
    CHECK_FOR_GETDOM_BEGIN1
    <xsl:choose>
        (<xsl:when test="BOOLEAN_XPATH_EXPRESSIONY">
            <xsl:attribute name="disabled">true</xsl:attribute>
        </xsl:when>)*
    </xsl:choose>
    <xsl:attribute name="xd:value">
        <xsl:value-of select="LEAF XPATH1" />
    </xsl:attribute>
    <xsl:if test="BOOLEAN_XPATH_EXPRESSION">
        <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
    </xsl:if>
    CHECK FOR GETDOM END1
</input>
(ANY_STRING2)?
```

CHECK_BOX_ONVALUE: xd:onValue="(ISO_DIGIT+)|("ANY_STRING")"

CHECK_BOX_OFFVALUE: xd:offValue="(ISO_DIGIT+)|("ANY_STRING")"

CHECK_BOX_SINGLE_VALUE: CHECK_BOX_OFFVALUE or **CHECK_BOX_ONVALUE.**

CHECK_BOX_BOTH_VALUES: CHECK_BOX_OFFVALUE and **CHECK_BOX_ONVALUE.**

CHECK_BOX_STYLE: Semicolon-delimited list of (**STYLE_MARGIN**?, **STYLE_WIDTH**?, **STYLE_HEIGHT**?, **STYLE_VERTICAL_ALIGN**?, **STYLE_COLOR**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_BORDER**?, **STYLE_FONT**?, **STYLE_TEXT_DECORATION**?)

Control-specific attributes used by the check box control are as follows:

- xd:binding (section 2.4.2.6)
- xd:boundProp (section 2.4.2.9)
- xd:CrtlId (section 2.4.2.10)
- xd:offValue (section 2.4.2.27)
- xd:onValue (section 2.4.2.28)
- xd:postbackModel (section 2.4.2.29)
- xd:value (section 2.4.2.34
- xd:xctname (section 2.4.2.35)

2.4.1.7 Contact Selector Control

The contact selector control provides the ability to select one or more entities from a user information list.

CONTACT_SELECTOR:

```
<object class="xdActiveX" hideFocus="1" style="CONTACT_SELECTOR_STYLE" (height="HEIGHT"
width="WIDTH")? classid="clsid:61e40d31-993d-4777-8fa0-19ca59b6d0bb" tabIndex="TAB_INDEX"
tabStop="true" xd:xctname="{{61e40d31-993d-4777-8fa0-19ca59b6d0bb}}" xd:CtrlId="CONTROL_ID"
xd:bindingType="xmlNode" xd:bindingProperty="Value" xd:boundProp="xd:inline"
contentEditable="false" xd:binding="GROUP XPATH" (title="ANY STRING")?
(accessKey="SINGLE_CHARACTER")?>
    <xsl:if test="function-available('xdImage:getImageUrl')">
        <xsl:attribute name="src">
            <xsl:value-of select="xdImage:getImageUrl(GROUP_XPATH)"/>
        </xsl:attribute>
    </xsl:if>
    (<xsl:choose>
        (<xsl:when test="BOOLEAN XPATH EXPRESSION">
            <xsl:attribute name="xd:disableEditing">yes</xsl:attribute>
        </xsl:when>)+
    </xsl:choose>)?
    <param NAME="ButtonFont" VALUE="CONTACT SELECTOR BUTTON FONT"/>
    <param NAME="ButtonText" VALUE="ANY_STRING"/>
    <param NAME="DisplayNameXPath" VALUE="CONTACT SELECTOR DISPLAY NAME XPATH"/>
    <param NAME="ObjectIdXPath" VALUE="CONTACT_SELECTOR_ACCOUNT_ID_XPATH"/>
    <param NAME="ObjectTypeXPath" VALUE="CONTACT_SELECTOR_ACCOUNT_TYPE_XPATH"/>
    <param NAME="SiteUrlXPath" VALUE="/Context/@siteUrl"/>
    <param NAME="SiteUrlDataSource" VALUE="Context"/>
    <param NAME="NewNodeTemplate" VALUE="CONTACT SELECTOR NEW NODE TEMPLATE"/>
    <param NAME="BackgroundColor" VALUE="CONTACT_SELECTOR_BACKGROUND_COLOR"/>
    <param NAME="MaxLines" VALUE="CONTACT_SELECTOR_MAX_LINES"/>
    <param NAME="Direction" VALUE="CONTACT_SELECTOR_DIRECTION"/>
</object>
```

Parameters used by the contact selector control are listed in the following table.

Parameter	Specification
ButtonFont	This parameter specifies the font that is used to render the button text and display names.
ButtonText	This parameter specifies the text that is displayed on the button that opens the address book.
DisplayNameXPath	This parameter specifies the **LEAF_XPATH** containing the display names.
ObjectIdXPath	This parameter specifies the **LEAF_XPATH** containing the object identifiers.

Parameter	Specification
ObjectTypeXPath	This parameter specifies the **LEAF_XPATH** containing the object types.
SiteUrlXPath	This parameter specifies the **LEAF_XPATH** in the secondary data source containing the server URL, whose user information list this control is querying. This parameter MUST be ignored by the form server.
SiteUrlDataSource	This parameter specifies the name of the secondary data source that contains the server URL, whose user information list (1) this control is querying. This parameter MUST be ignored by the form server.
NewNodeTemplate	This parameter specifies the XML template that is inserted in the form when a new contact is selected.
BackgroundColor	This parameter specifies the **background color** of the contact selector input box.
MaxLines	This parameter specifies the maximum number of lines used by the contact selector input box to render display names.
Direction	This parameter specifies whether this control is displaying **left-to-right** or **right-to-left**.

FONT: ANY_STRING without a comma.

FONT_ITALIC: The symbol that specifies if the text is shown italic or not. The following table lists the possible values and explanations.

Value	Description
"0"	Not italic
"1"	Italic

FONT_SIZE: All **integers** and all real numbers, ending in .5, between 1 and 2000, inclusive.

FONT_STRIKETHROUGH: The symbol that specifies if the text is shown with a strike though line. The following table lists the possible values and explanations.

Value	Description
"0"	No strike through
"1"	Strike through

FONT_UNDERLINE: The symbol that specifies if the text shown is underlined. The following table lists the possible values and explanations.

Value	Description
"0"	Not underlined
"1"	Underlined

FONT_WEIGHT: The symbol that specifies if the text shown is bold. The following table lists the possible values and explanations.

Value	Description
"400"	Not bold
"700"	Bold

CHARACTER_SET: ANY_STRING without a comma.

CONTACT_SELECTOR_BUTTON_FONT: FONT, **FONT_SIZE**, **CHARACTER_SET**, **FONT_WEIGHT**, **FONT_ITALIC**, **FONT_UNDERLINE**, **FONT_STRIKETHROUGH**

CONTACT_SELECTOR_STYLE: Semicolon-delimited list of (**STYLE_SIZE**?, **STYLE_MARGIN**?, **STYLE_TEXT_DECORATION**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_BORDER**?, **STYLE_FONT**?, **STYLE_COLOR**?, **STYLE_VERTICAL_ALIGN**?)

CONTACT_SELECTOR_BACKGROUND_COLOR: 2147483653 | MUST be an **integer** value that represents a **red-green-blue (RGB)** color. The value MUST be calculated using three variables (blue part, red part, green part), each of which MUST be an **integer** between 0 and 255, in the following formula:

blue part * 65536 + green part * 256 + red part

CONTACT_SELECTOR_MAX_LINES: MUST be an **integer** between zero and 999, inclusive.

CONTACT_SELECTOR_DIRECTION: The symbol that specifies if the control is rendered left-to-right or right-to-left. The following table lists the possible values and explanations.

Value	Description
"0"	Use the orientation of the form.
"1"	Left to right.
"2"	Right to-left.

CONTACT_SELECTOR_PERSON_XPATH: RELATIVE_REPREATING_GROUP_XPATH

CONTACT_SELECTOR_DISPLAY_NAME_XPATH: RELATIVE_LEAF_XPATH

CONTACT_SELECTOR_ACCOUNT_ID_XPATH: RELATIVE_LEAF_XPATH

CONTACT_SELECTOR_ACCOUNT_TYPE_XPATH: RELATIVE_LEAF_XPATH

CONTACT_SELECTOR_NEW_NODE_TEMPLATE:

```
&lt;RELATIVE REPREATING GROUP XPATH&gt;&#xA;

&lt;CONTACT SELECTOR DISPLAY NAME XPATH&gt;&lt;/CONTACT SELECTOR DISPLAY NAME XPATH&gt;&#xA;
    &lt;CONTACT_SELECTOR_ACCOUNT_ID_XPATH&gt;&lt;/CONTACT_SELECTOR_ACCOUNT_ID_XPATH&gt;&#xA;

&lt;CONTACT_SELECTOR_ACCOUNT_TYPE_XPATH&gt;&lt;/CONTACT_SELECTOR_ACCOUNT_TYPE_XPATH&gt;&#xA;
&lt;/RELATIVE_REPREATING_GROUP_XPATH&gt;
```

GROUP_XPATH: MUST point to an XML node in the main data source.

Control-specific attributes used by the contact selector control are as follows:

- xd:binding (section 2.4.2.6)
- xd:bindingProperty (section 2.4.2.7)
- xd:boundProp (section 2.4.2.9)
- xd:CtrlId (section 2.4.2.10)
- xd:disableEditing (section 2.4.2.12)
- xd:xctname (section 2.4.2.35)

The **xdImage:getImageUrl** XSL function extension, as specified in section 2.4.3.5, is used by the contact selector control.

2.4.1.8 Date Picker Control

A date picker control is used to select and display a date. A **DATE_PICKER** MUST have one of the symbols in the following table.

Symbol	Description
SIMPLE_DATE_PICKER	A date picker is a control with the capabilities of displaying, as well as selecting, a date. This is usually accomplished by having a button that displays a view of a calendar. Clicking on the calendar allows the user to select a specific date. The date can also be manually entered in the date picker's display text box.
DATE_PICKER_WITH_CONDITIONAL_FORMATTING	Similar to **SIMPLE_DATE_PICKER**, but allows for conditional formatting. Conditional text formatting attributes such as bold, italics, and color can be applied to the displayed date. Conditional disabling disables both the text box and the date picker's calendar button. Conditionally hiding the control hides both the display box and the calendar button.
DATE_PICKER_WITH_DATA_FORMATTING	Similar to **SIMPLE_DATE_PICKER**, but the data is formatted from its natural XML value.
DATE_PICKER_WITH_DATA_FORMATTING_AND_CONDITIONAL_FORMATTING	Similar to **DATE_PICKER_WITH_CONDITIONAL_FORMATTING**, but the data is formatted from its natural XML value.
DATE_PICKER_WITH_PLACEHOLDER_TEXT	Similar to **SIMPLE_DATE_PICKER**, with the addition of text that is displayed in the control until actual data is entered. This displayed value is not persisted in the field as a value.
DATE_PICKER_WITH_CONDITIONAL_FORMATTING_AND_PLACEHOLDER_TEXT	Similar to **DATE_PICKER_WITH_PLACEHOLDER_TEXT** and **DATE_PICKER_WITH_CONDITIONAL_FORMATTING**.
DATE_PICKER_WITH_DATA_FORMATTING_AND_PLACEHOLDER_TEXT	Similar to **DATE_PICKER_WITH_PLACEHOLDER_TEXT** and **DATE_PICKER_WITH_DATA_FORMATTING**.
DATE_PICKER_WITH_DATA_FORMATTING_AND_CONDITIONAL_FORMATTING_AND_PLACEHOLDER_TEXT	Similar to **DATE_PICKER_WITH_PLACEHOLDER_TEXT** and **DATE_PICKER_WITH_DATA_FORMATTING_AND_CONDITIONAL_FORMATTING**.

SIMPLE_DATE_PICKER:

```
<div class="xdDTPicker" title="ANY_STRING1" style="DATE_PICKER_STYLE" noWrap="1"
xd:CtrlId="CONTROL_ID" xd:xctname="DTPicker" (xd:postbackModel="POSTBACKMODEL")?>
    <span class="DATE_PICKER_TEXT_BOX_CLASS_NAME xdBehavior_NoBUI" hideFocus="1"
(title="ANY STRING1")? (accessKey="SINGLE CHARACTER")? xd:xctname="DTPicker DTText"
xd:binding="LEAF_XPATH1" (tabIndex="TAB_INDEX")? xd:innerCtrl="_DTText" (INPUT_SCOPE)?>
    CHECK FOR GETDOM BEGIN1
        <xsl:value-of select="LEAF_XPATH1" />
    CHECK_FOR_GETDOM_END1
    </span>
    <button class="DATE PICKER BUTTON CLASS NAME" (title="ANY STRING1")?
xd:xctname="DTPicker DTButton" xd:innerCtrl=" DTButton" (tabIndex="-1")?>
        <img (title="ANY_STRING1")? src="res://infopath.exe/calendar.gif" (Linked="true")?/>
    </button>
</div>
```

DATE_PICKER_WITH_CONDITIONAL_FORMATTING:

```
<div class="xdDTPicker" title="ANY_STRING1" (style="DATE_PICKER_STYLE")? noWrap="1"
xd:CtrlId="CONTROL_ID" xd:xctname="DTPicker" (xd:postbackModel="POSTBACKMODEL")?>
    <span class="DATE PICKER TEXT BOX CLASS NAME xdBehavior NoBUI" hideFocus="1"
(title="ANY_STRING1")? (accessKey="SINGLE_CHARACTER")? xd:xctname="DTPicker_DTText"
xd:binding="LEAF_XPATH1" (tabIndex="TAB_INDEX")? xd:innerCtrl="_DTText" (INPUT_SCOPE)?>
    CHECK_FOR_GETDOM_BEGIN1
        (<xsl:attribute name="style">
            <xsl:choose>
                (<xsl:when
test="BOOLEAN_XPATH_EXPRESSONx">(DATE_PICKER_STYLE_CONDITIONAL_FORMATTING)?</xsl:when>)*
            </xsl:choose>
        </xsl:attribute>)?
```

```
        (<xsl:choose>
            (<xsl:when test="BOOLEAN_XPATH_EXPRESSIONy">
                (<xsl:attribute name="contentEditable">false</xsl:attribute>)?
            </xsl:when>)*
        </xsl:choose>)?
        <xsl:value-of select="LEAF XPATH1" />
    CHECK_FOR_GETDOM_END1
    </span>
    <button class="DATE_PICKER_BUTTON_CLASS_NAME" (title="ANY_STRING1")?
xd:xctname="DTPicker_DTButton" xd:innerCtrl="_DTButton" (tabIndex="-1")?>
        <img (title="ANY STRING1")? src="res://infopath.exe/calendar.gif" (Linked="true")?/>
    </button>
</div>
```

DATE_PICKER_WITH_DATA_FORMATTING:

```
<div class="xdDTPicker" title="ANY_STRING1" style="DATE_PICKER_STYLE" noWrap="1"
xd:CtrlId="CONTROL ID" xd:xctname="DTPicker" (xd:postbackModel="POSTBACKMODEL")?>
    <span class="DATE_PICKER_TEXT_BOX_CLASS_NAME xdBehavior_FormattingNoBUI" hideFocus="1"
contentEditable="true" xd:xctname="DTPicker DTText" xd:datafmt="DATA FMT CTRL DATE PICKER"
xd:boundProp="xd:num" xd:binding="LEAF_XPATH1" (accessKey="SINGLE_CHARACTER")?
(title="ANY_STRING1")? (tabIndex="TAB_INDEX")? xd:innerCtrl="_DTText" (INPUT_SCOPE)?>
    CHECK_FOR_GETDOM_BEGIN1
        (<xsl:attribute name="xd:num">
            <xsl:value-of select="LEAF_XPATH1" />
        </xsl:attribute>)?
        <xsl:choose>
            <xsl:when test="function-available('xdFormatting:formatString')">
                <xsl:value-of select="xdFormatting:formatString(LEAFXPATH,
DATA_FMT_CTRL_DATE_PICKER)" />
            </xsl:when>
            <xsl:otherwise>
                <xsl:value-of select="LEAF_XPATH1" />
            </xsl:otherwise>
        </xsl:choose>
    CHECK_FOR_GETDOM_END1
    </span>
    <button class="DATE_PICKER_BUTTON_CLASS_NAME" (title="ANY_STRING1")?
xd:xctname="DTPicker_DTButton" xd:innerCtrl="_DTButton" (tabIndex="-1")?>
        <img (title="ANY_STRING1")? src="res://infopath.exe/calendar.gif" (Linked="true")?/>
    </button>
</div>
```

DATE_PICKER_WITH_DATA_FORMATTING_AND_CONDITIONAL_FORMATTING:

```
<div class="xdDTPicker" title="ANY STRING1" (style="DATE PICKER STYLE")? noWrap="1"
xd:CtrlId="CONTROL_ID" xd:xctname="DTPicker" (xd:postbackModel="POSTBACKMODEL")?>
    <span class="DATE_PICKER_TEXT_BOX_CLASS_NAME xdBehavior_FormattingNoBUI" hideFocus="1"
contentEditable="true" xd:xctname="DTPicker DTText" xd:datafmt="DATA FMT CTRL DATE PICKER"
xd:boundProp="xd:num" xd:binding="LEAF_XPATH1" (accessKey="SINGLE_CHARACTER")?
(title="ANY STRING1")? (tabIndex="TAB INDEX")? xd:innerCtrl=" DTText" (INPUT SCOPE)?>
    CHECK_FOR_GETDOM_BEGIN1
        (<xsl:attribute name="style">
            <xsl:choose>
                (<xsl:when
test="BOOLEAN_XPATH_EXPRESSIONx">(DATE_PICKER_STYLE_CONDITIONAL_FORMATTING)?</xsl:when>)*
            </xsl:choose>
        </xsl:attribute>)?
        (<xsl:choose>
            (<xsl:when test="BOOLEAN_XPATH_EXPRESSIONy">
                (<xsl:attribute name="contentEditable">false</xsl:attribute>)?
            </xsl:when>)*
        </xsl:choose>)?
        (<xsl:attribute name="xd:num">
            <xsl:value-of select="LEAF_XPATH1" />
```

```
        </xsl:attribute>)?
        <xsl:choose>
            <xsl:when test="function-available('xdFormatting:formatString')">
                <xsl:value-of select="xdFormatting:formatString(LEAFXPATH,
DATA_FMT_CTRL_DATE_PICKER)" />
            </xsl:when>
            <xsl:otherwise>
                <xsl:value-of select="LEAF_XPATH1" />
            </xsl:otherwise>
        </xsl:choose>
    CHECK FOR GETDOM END1
    </span>
    <button class="DATE_PICKER_BUTTON_CLASS_NAME" (title="ANY_STRING1")?
xd:xctname="DTPicker_DTButton" xd:innerCtrl="_DTButton" (tabIndex="-1")?>
        <img (title="ANY_STRING1")? src="res://infopath.exe/calendar.gif" (Linked="true")?/>
    </button>
</div>
```

DATE_PICKER_WITH_PLACEHOLDER_TEXT:

```
<div class="xdDTPicker" title="ANY_STRING1" style="DATE_PICKER_STYLE" noWrap="1"
xd:CtrlId="CONTROL_ID" xd:xctname="DTPicker" (xd:postbackModel="POSTBACKMODEL")?>
    <span class="DATE_PICKER_TEXT_BOX_CLASS_NAME xdBehavior_GhostedTextNoBUI" hideFocus="1"
contentEditable="true" (title="ANY STRING1")? (accessKey="SINGLE CHARACTER")?
xd:xctname="DTPicker_DTText" xd:binding="LEAF_XPATH1" (tabIndex="TAB_INDEX")?
xd:innerCtrl=" DTText" (INPUT SCOPE)?>
    CHECK_FOR_GETDOM_BEGIN1
        (<xsl:choose>
            <xsl:when test="not(string(LEAF XPATH1))">
                <xsl:attribute name="xd:ghosted">true</xsl:attribute>
                    ANY STRING
                </xsl:when>
            <xsl:otherwise>
                <xsl:value-of select="LEAF_XPATH1" />
            </xsl:otherwise>
        </xsl:choose>) | (<xsl:value-of select="LEAF_XPATH1" />)
    CHECK FOR GETDOM END1
    </span>
    <button class="DATE_PICKER_BUTTON_CLASS_NAME" (title="ANY_STRING1")?
xd:xctname="DTPicker_DTButton" xd:innerCtrl="_DTButton" (tabIndex="-1")?>
        <img (title="ANY STRING1")? src="res://infopath.exe/calendar.gif" (Linked="true")?/>
    </button>
</div>
DATE_PICKER_WITH_CONDITIONAL_FORMATTING_AND_PLACEHOLDER_TEXT ::=
<div class="xdDTPicker" title="ANY_STRING1" (style="DATE_PICKER_STYLE")? noWrap="1"
xd:CtrlId="CONTROL_ID" xd:xctname="DTPicker" (xd:postbackModel="POSTBACKMODEL")?>
    <span class="DATE_PICKER_TEXT_BOX_CLASS_NAME xdBehavior_GhostedTextNoBUI" hideFocus="1"
contentEditable="true" (title="ANY_STRING1")? (accessKey="SINGLE_CHARACTER")?
xd:xctname="DTPicker DTText" xd:binding="LEAF XPATH1" (tabIndex="TAB INDEX")?
xd:innerCtrl="_DTText" (INPUT_SCOPE)?>
    CHECK_FOR_GETDOM_BEGIN1
        (<xsl:attribute name="style">
            <xsl:choose>
                (<xsl:when
test="BOOLEAN XPATH EXPRESSIONx">(DATE PICKER STYLE CONDITIONAL FORMATTING)?</xsl:when>)*
            </xsl:choose>
        </xsl:attribute>)?
        (<xsl:choose>
            (<xsl:when test="BOOLEAN_XPATH_EXPRESSIONy">
                (<xsl:attribute name="contentEditable">false</xsl:attribute>)?
            </xsl:when>)*
        </xsl:choose>)?
        (<xsl:choose>
            <xsl:when test="not(string(LEAF_XPATH1))">
                <xsl:attribute name="xd:ghosted">true</xsl:attribute>
                    ANY STRING
                </xsl:when>
```

```
            <xsl:otherwise>
                <xsl:value-of select="LEAF_XPATH1" />
            </xsl:otherwise>
        </xsl:choose>) | (<xsl:value-of select="LEAF_XPATH1" />)
    CHECK_FOR_GETDOM_END1
    </span>
    <button class="DATE_PICKER_BUTTON_CLASS_NAME" (title="ANY_STRING1")?
xd:xctname="DTPicker_DTButton" xd:innerCtrl="_DTButton" (tabIndex="-1")?>
        <img (title="ANY_STRING1")? src="res://infopath.exe/calendar.gif" (Linked="true")?/>
    </button>
</div>
DATE_PICKER_WITH_DATA_FORMATTING_AND_PLACEHOLDER_TEXT ::=
<div class="xdDTPicker" title="ANY_STRING1" style="DATE_PICKER_STYLE" noWrap="1"
xd:CtrlId="CONTROL_ID" xd:xctname="DTPicker" (xd:postbackModel="POSTBACKMODEL")?>
    <span class="DATE_PICKER_TEXT_BOX_CLASS_NAME xdBehavior_GTFormattingNoBUI" hideFocus="1"
contentEditable="true" contentEditable="true" xd:xctname="DTPicker_DTText"
xd:datafmt="DATA FMT CTRL DATE PICKER" xd:boundProp="xd:num" xd:binding="LEAF XPATH1"
(accessKey="SINGLE_CHARACTER")? (title="ANY_STRING1")? (tabIndex="TAB_INDEX")?
xd:innerCtrl=" DTText" (INPUT SCOPE)?>
    CHECK_FOR_GETDOM_BEGIN1
        (<xsl:attribute name="xd:num">
            <xsl:value-of select="LEAF_XPATH1" />
        </xsl:attribute>)?
        <xsl:choose>
            (<xsl:when test="not(string(LEAF_XPATH1))">
                <xsl:attribute name="xd:ghosted">true</xsl:attribute>
                ANY_STRING
            </xsl:when>)?
            <xsl:when test="function-available('xdFormatting:formatString')">
                <xsl:value-of select="xdFormatting:formatString(LEAFXPATH,
DATA FMT CTRL DATE PICKER)" />
            </xsl:when>
            <xsl:otherwise>
                <xsl:value-of select="LEAF XPATH1" />
            </xsl:otherwise>
        </xsl:choose>
    CHECK_FOR_GETDOM_END1
    </span>
    <button class="DATE_PICKER_BUTTON_CLASS_NAME" (title="ANY_STRING1")?
xd:xctname="DTPicker DTButton" xd:innerCtrl=" DTButton" (tabIndex="-1")?>
        <img (title="ANY_STRING1")? src="res://infopath.exe/calendar.gif" (Linked="true")?/>
    </button>
</div>
```

DATE_PICKER_WITH_DATA_FORMATTING_AND_CONDITIONAL_FORMATTING_AND_PLACEHOLDER_TEXT:

```
<div class="xdDTPicker" title="ANY_STRING1" (style="DATE_PICKER_STYLE")? noWrap="1"
xd:CtrlId="CONTROL ID" xd:xctname="DTPicker" (xd:postbackModel="POSTBACKMODEL")?>
    <span class="DATE_PICKER_TEXT_BOX_CLASS_NAME xdBehavior_GTFormattingNoBUI" hideFocus="1"
contentEditable="true" contentEditable="true" xd:xctname="DTPicker_DTText"
xd:datafmt="DATA_FMT_CTRL_DATE_PICKER" xd:boundProp="xd:num" xd:binding="LEAF_XPATH1"
(accessKey="SINGLE CHARACTER")? (title="ANY STRING1")? (tabIndex="TAB INDEX")?
xd:innerCtrl="_DTText" (INPUT_SCOPE)?>
    CHECK FOR GETDOM BEGIN1
        (<xsl:attribute name="style">
            <xsl:choose>
                (<xsl:when
test="BOOLEAN XPATH EXPRESSIONx">(DATE PICKER STYLE CONDITIONAL FORMATTING)?</xsl:when>)*
            </xsl:choose>
        </xsl:attribute>)?
        <xsl:choose>
            (<xsl:when test="BOOLEAN_XPATH_EXPRESSIONy">
                (<xsl:attribute name="contentEditable">false</xsl:attribute>)?
            </xsl:when>)*
        </xsl:choose>
        (<xsl:attribute name="xd:num">
```

```
            <xsl:value-of select="LEAF_XPATH1" />
        </xsl:attribute>)?
        <xsl:choose>
            (<xsl:when test="not(string(LEAF_XPATH1))">
                <xsl:attribute name="xd:ghosted">true</xsl:attribute>
                ANY STRING
            </xsl:when>)?
            <xsl:when test="function-available('xdFormatting:formatString')">
                <xsl:value-of select="xdFormatting:formatString(LEAFXPATH,
DATA_FMT_CTRL_DATE_PICKER)" />
            </xsl:when>
            <xsl:otherwise>
                <xsl:value-of select="LEAF_XPATH1" />
            </xsl:otherwise>
        </xsl:choose>
    CHECK_FOR_GETDOM_END1
    </span>
    <button class="DATE_PICKER_BUTTON_CLASS_NAME" (title="ANY_STRING1")?
xd:xctname="DTPicker_DTButton" xd:innerCtrl="_DTButton" (tabIndex="-1")?>
        <img (title="ANY_STRING1")? src="res://infopath.exe/calendar.gif" (Linked="true")?/>
    </button>
</div>
```

DATE_PICKER_TEXT_BOX_CLASS_NAME: xdDTText or **xdDTTextRTL.**

DATE_PICKER_BUTTON_CLASS_NAME: xdDTButton or **xdDTButtonRTL.**

DATE_PICKER_STYLE: Semicolon-delimited list of (**STYLE_WIDTH**?, **STYLE_PADDING**?, **STYLE_FONT**?, **STYLE_HEIGHT**?, **STYLE_TEXT_ALIGN**?, **STYLE_MARGIN**?, **WHITE-SPACE: nowrap**?, **STYLE_TEXT_DECORATION**?, **STYLE_BORDER**?, **STYLE_VERTICAL_ALIGN**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_COLOR**?, **STYLE_DIRECTION**?).

DATE_PICKER_STYLE_CONDITIONAL_FORMATTING: Semicolon-delimited list of (**STYLE_FONT_WEIGHT**?, **STYLE_COLOR**?, **STYLE_TEXT_DECORATION**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_FONT_STYLE**?).

Control-specific attributes used by the date picker control are as follows:

- xd:AllowNonMatching (section 2.4.2.2)
- xd:binding (section 2.4.2.6)
- xd:boundProp (section 2.4.2.9)
- xd:CtrlId (section 2.4.2.10
- xd:datafmt (section 2.4.2.11)
- xd:innerCtrl (section 2.4.2.19)
- xd:inputscope (section 2.4.2.20
- xd:num (section 2.4.2.26)
- xd:postbackModel (section 2.4.2.29)
- xd:xctname (section 2.4.2.35)

2.4.1.9 Drop-Down List Control

The dropdown list control enables the user to select a single value from a list of options that can be specified manually by the form template designer, or is populated from a data source. A

DROPDOWN_LIST_BOX MUST have one of the symbols in the following table.

Symbol	Description
SIMPLE_DROPDOWN_LIST_BOX	A drop down list box is a control that allows the user to select an entry from a collection of values. The collection of values tends to be hidden until the user has them displayed. The collection of values is statically available in the XSL.
DROPDOWN_LIST_BOX_WITH_CONDITIONAL_FORMATTING	Similar to **SIMPLE_DROPDOWN_LIST_BOX**, but allows conditional formatting (text formatting and disabling).
DROPDOWN_LIST_BOX_WTIH_VALUES_FROM_DATA_SOURCE	Similar to **SIMPLE_DROPDOWN_LIST_BOX** with the exception that the values for the collection are drawn from another location within the data source of the form.
DROPDOWN_LIST_BOX_WITH_VALUES_FROM_DATA_SOURCE_AND_CONDITIONAL_FORMATTING	Similar to **DROPDOWN_LIST_BOX_WITH_VALUES_FROM_EXTERNAL_DATA_SOURCE** and **DROPDOWN_LIST_BOX_WTIH_VALUES_FROM_DATA_SOURCE_AND_CONDITIONAL_FORMATTING.**
DROPDOWN_LIST_BOX_WITH_VALUES_FROM_DATA_SOURCE_AND_UNIQUE_DISPLAY_NAMES	Similar to **DROPDOWN_LIST_BOX_WITH_VALUES_FROM_DATA_SOURCE** with the difference that each value from the collection of values is unique.
DROPDOWN_LIST_BOX_WITH_VALUES_FROM_DATA_SOURCE_AND_UNIQUE_DISPLAY_NAMES_AND_CONDITIONAL_FORMATTING	Similar to **DROPDOWN_LIST_BOX_WITH_VALUES_FROM_DATA_SOURCE_AND_UNIQUE_DISPLAY_NAMES** and **DROPDOWN_LIST_BOX_WITH_CONDITIONAL_FORMATTING.**

SIMPLE_DROPDOWN_LIST_BOX:

```
<select class="xdComboBox xdBehavior_Select" title="ANY_STRING1"
(accessKey="SINGLE_CHARACTER")? size="FONT_SIZE" xd:binding="LEAF_XPATH1"
xd:boundProp="value" xd:xctname="dropdown" (xd:postbackModel="POSTBACKMODEL")?
(tabIndex="TAB_INDEX")? xd:CtrlId="CONTROL_ID" (style="DROPDOWN_LIST_BOX_STYLES")?>
    CHECK_FOR_GETDOM_BEGIN1
    <xsl:attribute name="value">
        <xsl:value-of select="LEAF XPATH1" />
    </xsl:attribute>
    ((<option (value="")?>
        <xsl:if test="LEAF_XPATH1=&amp;quot;&amp;quot;">
            <xsl:attribute name="selected">selected</xsl:attribute>
        </xsl:if>
        ANY STRING2
    </option>) |
    (<option value="LEAF VALUE1">
        <xsl:if test="LEAF_XPATH1=LEAF_VALUE1">
            <xsl:attribute name="selected">selected</xsl:attribute>
        </xsl:if>
        ANY_STRINGX
    </option>))+
    CHECK_FOR_GETDOM_END1
</select>
```

DROPDOWN_LIST_BOX_WITH_CONDITIONAL_FORMATTING:

```
<select class="xdComboBox xdBehavior_Select" title="ANY_STRING1"
(accessKey="SINGLE CHARACTER")? (style="DROPDOWN LIST BOX STYLES")? size="FONT SIZE"
xd:binding="LEAF_XPATH1" xd:boundProp="value" xd:xctname="dropdown"
(xd:postbackModel="POSTBACKMODEL")? (tabIndex="TAB INDEX")? xd:CtrlId="CONTROL ID">
    CHECK_FOR_GETDOM_BEGIN1
    (<xsl:attribute name="style">
        DROPDOWN_LIST_BOX_STYLES
        <xsl:choose>
```

```
            (DROPDOWN_LIST_BOX_CONDITIONAL_FORMATTING)*
        </xsl:choose>
    </xsl:attribute>)?
    (<xsl:choose>
        (DROPDOWN_LIST_BOX_CONDITIONAL_FORMATTING | DROPDOWN_LIST_BOX_CONDITIONAL_DISABLING)+
    </xsl:choose>)?
    <xsl:attribute name="value">
        <xsl:value-of select="LEAF_XPATH1" />
    </xsl:attribute>
    ((<option (value="")?>
        <xsl:if test="LEAF XPATH1=&amp;quot;&amp;quot;">
            <xsl:attribute name="selected">selected</xsl:attribute>
        </xsl:if>
        ANY_STRING2
    </option>) |
    (<option value="LEAF_VALUE1">
        <xsl:if test="LEAF XPATH1=LEAF VALUE1">
            <xsl:attribute name="selected">selected</xsl:attribute>
        </xsl:if>
        ANY_STRINGX
    </option>))+
    CHECK_FOR_GETDOM_END1
</select>
```

DROPDOWN_LIST_BOX_WITH_VALUES_FROM_DATA_SOURCE:

```
<select class="xdComboBox xdBehavior_Select" title="ANY_STRING1"
(accessKey="SINGLE_CHARACTER")? size="FONT_SIZE" xd:binding="LEAF_XPATH1"
xd:boundProp="value" value="ANY STRING2" xd:xctname="dropdown"
(xd:postbackModel="POSTBACKMODEL")? (tabIndex="TAB_INDEX")? xd:CtrlId="CONTROL_ID"
(style="DROPDOWN LIST BOX STYLES")?>
    CHECK_FOR_GETDOM_BEGIN1
    <xsl:attribute name="value">
        <xsl:value-of select="LEAF_XPATH1" />
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="function-available('xdXDocument:GetDOM')">
            <option />
            <xsl:variable name="val" select="LEAF_XPATH1" />
            <xsl:if test="not(REPEATING_LEAF_XPATH1[LEAF_XPATH=$val] or $val='')">
                <option selected="selected">
                    <xsl:attribute name="value">
                        <xsl:value-of select="$val" />
                    </xsl:attribute>
                    <xsl:value-of select="$val" />
                </option>
            </xsl:if>
            <xsl:for-each select="REPEATING_LEAF_XPATH1">
                <option>
                    <xsl:attribute name="value">
                        <xsl:value-of select="RELATIVE_LEAF_XPATH1" />
                    </xsl:attribute>
                    <xsl:if test="$val=RELATIVE_LEAF_XPATH1">
                        <xsl:attribute name="selected">selected</xsl:attribute>
                    </xsl:if>
                    <xsl:value-of select="RELATIVE_LEAF_XPATH2"/>
                </option>
            </xsl:for-each>
        </xsl:when>
        <xsl:otherwise>
            <option>
                <xsl:value-of select="LEAF_XPATH1" />
            </option>
        </xsl:otherwise>
    </xsl:choose>
    CHECK_FOR_GETDOM_END1
```

```
</select>
```

DROPDOWN_LIST_BOX_WITH_VALUES_FROM_DATA_SOURCE_AND_CONDITIONAL_FORMATTING:

```
<select class="xdComboBox xdBehavior_Select" title="ANY_STRING1"
(accessKey="SINGLE CHARACTER")? (style="DROPDOWN LIST BOX STYLES")? size="FONT SIZE"
xd:binding="LEAF_XPATH1" xd:boundProp="value" value="ANY_STRING2" xd:xctname="dropdown"
(xd:postbackModel="POSTBACKMODEL")? (tabIndex="TAB INDEX")? xd:CtrlId="CONTROL ID">
    CHECK_FOR_GETDOM_BEGIN1
    (<xsl:attribute name="style">
        DROPDOWN_LIST_BOX_STYLES
        <xsl:choose>
            (DROPDOWN_LIST_BOX_CONDITIONAL_FORMATTING)*
        </xsl:choose>
    </xsl:attribute>)?
    (<xsl:choose>
                (DROPDOWN_LIST_BOX_CONDITIONAL_FORMATTING |
DROPDOWN LIST BOX CONDITIONAL DISABLING)+
    </xsl:choose>)?
    <xsl:attribute name="value">
        <xsl:value-of select="LEAF_XPATH1" />
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="function-available('xdXDocument:GetDOM')">
            <option />
            <xsl:variable name="val" select="LEAF_XPATH1" />
            <xsl:if test="not(REPEATING_LEAF_XPATH1[LEAF_XPATH=$val] or $val='')">
                <option selected="selected">
                    <xsl:attribute name="value">
                        <xsl:value-of select="$val" />
                    </xsl:attribute>
                    <xsl:value-of select="$val" />
                </option>
            </xsl:if>
            <xsl:for-each select="REPEATING LEAF XPATH1">
                <option>
                    <xsl:attribute name="value">
                        <xsl:value-of select="RELATIVE_LEAF_XPATH1" />
                    </xsl:attribute>
                    <xsl:if test="$val=RELATIVE_LEAF_XPATH1">
                        <xsl:attribute name="selected">selected</xsl:attribute>
                    </xsl:if>
                    <xsl:value-of select="RELATIVE LEAF XPATH2"/>
                </option>
            </xsl:for-each>
        </xsl:when>
        <xsl:otherwise>
            <option>
                <xsl:value-of select="LEAF_XPATH1" />
            </option>
        </xsl:otherwise>
    </xsl:choose>
    CHECK FOR GETDOM END1
</select>
```

DROPDOWN_LIST_BOX_WITH_VALUES_FROM_DATA_SOURCE_AND_UNIQUE_DISPLAY_NAMES:

```
<select class="xdComboBox xdBehavior_Select" title="ANY_STRING1"
(accessKey="SINGLE_CHARACTER")? size="FONT_SIZE" xd:binding="LEAF_XPATH1"
xd:boundProp="value" value="ANY STRING2" xd:xctname="dropdown"
(xd:postbackModel="POSTBACKMODEL")? (tabIndex="TAB_INDEX")? xd:CtrlId="CONTROL_ID"
(style="DROPDOWN_LIST_BOX_STYLES")?>
```

```
    CHECK_FOR_GETDOM_BEGIN1
    <xsl:attribute name="value">
        <xsl:value-of select="LEAF_XPATH1" />
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="function-available('xdXDocument:GetDOM')">
            <option />
            <xsl:variable name="val" select="LEAF_XPATH1" />
            <xsl:if test="not(REPEATING_LEAF_XPATH1[LEAF_XPATH=$val] or $val='')">
                <option selected="selected">
                    <xsl:attribute name="value">
                        <xsl:value-of select="$val" />
                    </xsl:attribute>
                    <xsl:value-of select="$val" />
                </option>
            </xsl:if>
            <xsl:variable name="items">
                <xsl:copy-of select="REPEATING_LEAF_XPATH1" />
            </xsl:variable>
            <xsl:variable name="uniqueItems" select="msxsl:node-set($items)/*[not(LEAF_XPATH
= preceding::LEAF_XPATH2)]" />
            <xsl:for-each select="$uniqueItems">
                <option>
                    <xsl:attribute name="value">
                        <xsl:value-of select="RELATIVE_LEAF_XPATH1" />
                    </xsl:attribute>
                    <xsl:if test="$val=RELATIVE_LEAF_XPATH1">
                        <xsl:attribute name="selected">selected</xsl:attribute>
                    </xsl:if>
                    <xsl:value-of select="RELATIVE_LEAF_XPATH2"/>
                </option>
            </xsl:for-each>
        </xsl:when>
        <xsl:otherwise>
            <option>
                <xsl:value-of select="LEAF XPATH1" />
            </option>
        </xsl:otherwise>
    </xsl:choose>
    CHECK FOR GETDOM END1
</select>
```

DROPDOWN_LIST_BOX_WITH_VALUES_FROM_DATA_SOURCE_AND_UNIQUE_DISPLAY_NAMES_AND_CONDITIONAL_FORMATTING:

```
<select class="xdComboBox xdBehavior_Select" title="ANY_STRING1"
(accessKey="SINGLE CHARACTER")? (style="DROPDOWN LIST BOX STYLES")? size="FONT SIZE"
xd:binding="LEAF_XPATH1" xd:boundProp="value" value="ANY_STRING2" xd:xctname="dropdown"
(xd:postbackModel="POSTBACKMODEL ")? (tabIndex="TAB INDEX")? xd:CtrlId="CONTROL ID">
    CHECK_FOR_GETDOM_BEGIN1
    (<xsl:attribute name="style">
        DROPDOWN_LIST_BOX_STYLES
        <xsl:choose>
            (DROPDOWN_LIST_BOX_CONDITIONAL_FORMATTING)*
        </xsl:choose>
    </xsl:attribute>)?
    (<xsl:choose>
        (DROPDOWN_LIST_BOX_CONDITIONAL_FORMATTING | DROPDOWN_LIST_BOX_CONDITIONAL_DISABLING)+
    </xsl:choose>)?
    <xsl:attribute name="value">
        <xsl:value-of select="LEAF XPATH1" />
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="function-available('xdXDocument:GetDOM')">
            <option />
            <xsl:variable name="val" select="LEAF XPATH1" />
            <xsl:if test="not(REPEATING_LEAF_XPATH1[LEAF_XPATH=$val] or $val='')">
```

```
                <option selected="selected">
                    <xsl:attribute name="value">
                        <xsl:value-of select="$val" />
                    </xsl:attribute>
                    <xsl:value-of select="$val" />
                </option>
            </xsl:if>
            <xsl:variable name="items">
                <xsl:copy-of select="REPEATING_LEAF_XPATH1" />
            </xsl:variable>
            <xsl:variable name="uniqueItems" select="msxsl:node-set($items)/*[not(LEAF XPATH
= preceding::LEAF_XPATH2)]" />
            <xsl:for-each select="$uniqueItems">
                <option>
                    <xsl:attribute name="value">
                        <xsl:value-of select="RELATIVE_LEAF_XPATH1" />
                    </xsl:attribute>
                    <xsl:if test="$val=RELATIVE_LEAF_XPATH1">
                        <xsl:attribute name="selected">selected</xsl:attribute>
                    </xsl:if>
                    <xsl:value-of select="RELATIVE_LEAF_XPATH2"/>
                </option>
            </xsl:for-each>
        </xsl:when>
        <xsl:otherwise>
            <option>
                <xsl:value-of select="LEAF_XPATH1" />
            </option>
        </xsl:otherwise>
    </xsl:choose>
    CHECK_FOR_GETDOM_END1
</select>
```

DROPDOWN_LIST_BOX_CONDITIONAL_DISABLING:

```
<xsl:when test="BOOLEAN_XPATH_EXPRESSIONY">
    <xsl:attribute name="disabled">true</xsl:attribute>
</xsl:when>
```

DROPDOWN_LIST_BOX_CONDITIONAL_FORMATTING:

```
<xsl:when
test="BOOLEAN_XPATH_EXPRESSIONX">(LEAF_CONTROL_CONDITIONAL_FORMATTING_CAPTION)?</xsl:when>
```

DROPDOWN_LIST_BOX_STYLES: Semicolon-delimited list of (**STYLE_WIDTH**?, **STYLE_FONT**?, **STYLE_MARGIN**?, **STYLE_VERTICAL_ALIGN**?, **STYLE_TEXT_DECORATION**?, **STYLE_COLOR**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_DIRECTION**?).

Control-specific attributes used by the dropdown list control are as follows:

- xd:binding (section 2.4.2.6)
- xd:boundProp (section 2.4.2.9)
- xd:CrtlId (section 2.4.2.10)
- xd:postbackModel (section 2.4.2.29)
- xd:xctname (section 2.4.2.35)

The **xdXDocument:GetDOM** XSL function extension, as specified in section 2.4.3.9.2, is used by the

dropdown list control.

2.4.1.10 Expression Box Control

The expression box control is a read-only control that displays the result of an XPath expression evaluation. An **EXPRESSION_BOX** MUST have one of the symbols in the following table.

Symbol	Description
SIMPLE_EXPRESSION_BOX	An expression box is a control that displays the value of an XPath expression. It is constantly disabled.
EXPRESSIONBOX_WITH_CONDITIONAL_FORMATTING	Similar to **SIMPLE_EXPRESSION_BOX**, with text formatting and conditional formatting.
EXPRESSIONBOX_WITH_DATA_FORMATTING	Similar to **SIMPLE_EXPRESSION_BOX**, with the result formatted as a type of data.
EXPRESSIONBOX_WITH_DATA_FORMATTING_AND_CONDITIONAL_FORMATTING	Similar to **EXPRESSIONBOX_WITH_CONDITIONAL_FORMATTING** and **EXPRESSIONBOX_WITH_DATA_FORMATTING**.

SIMPLE_EXPRESSION_BOX:

```
<span class="xdExpressionBox xdDataBindingUI (xdBehavior Formatting)?" title="ANY STRING"
(tabIndex="-1")? xd:xctname="ExpressionBox" xd:CtrlId="CONTROL_ID" xd:disableEditing="yes"
(xd:binding="EXPRESSION_BOX_XPATH1")? style="EXPRESSION_BOX_STYLE">
    CHECK_FOR_GETDOM_BEGIN1
    <xsl:value-of select="EXPRESSION_BOX_XPATH1" />
    CHECK_FOR_GETDOM_END1
</span>
```

EXPRESSIONBOX_WITH_CONDITIONAL_FORMATTING:

```
<span class="xdExpressionBox xdDataBindingUI" title="ANY_STRING" (tabIndex="-1")?
xd:xctname="ExpressionBox" xd:CtrlId="CONTROL_ID" xd:disableEditing="yes"
(xd:binding="EXPRESSION_BOX_XPATH1")?>
    CHECK FOR GETDOM BEGIN1
    <xsl:attribute name="style">EXPRESSION BOX STYLE
       <xsl:choose>
            (<xsl:when
test="BOOLEAN_XPATH_EXPRESSIONX">LEAF_CONTROL_CONDITIONAL_FORMATTING_CAPTION</xsl:when>)+
        </xsl:choose>
    </xsl:attribute>
    <xsl:value-of select="EXPRESSION BOX XPATH1" />
    CHECK_FOR_GETDOM_END1
</span>
```

EXPRESSION_BOX_WITH_DATA_FORMATTING:

```
<span class="xdExpressionBox xdDataBindingUI xdBehavior_Formatting" title="ANY_STRING"
(tabIndex="-1")? xd:xctname="ExpressionBox" xd:CtrlId="CONTROL ID" xd:disableEditing="yes"
xd:binding="EXPRESSION BOX XPATH1" xd:datafmt="DATA FMT CTRL EXPBOX" (xd:num="")?
style="EXPRESSION BOX STYLE">
    CHECK_FOR_GETDOM_BEGIN1
    (<xsl:attribute name="xd:num">
        <xsl:value-of select="EXPRESSION_BOX_XPATH1" />
    </xsl:attribute>)?
    <xsl:choose>
        <xsl:when test="function-available('xdFormatting:formatString')">
            <xsl:value-of select="xdFormatting:formatString(EXPRESSION_BOX_XPATH1,
DATA_FMT_CTRL_EXPBOX)"
        </xsl:when>
        <xsl:otherwise>
```

```
            <xsl:value-of select="EXPRESSION_BOX_XPATH1" />
        </xsl:otherwise>
    </xsl:choose>
    CHECK_FOR_GETDOM_END1
</span>
```

EXPRESSION_BOX_WITH_DATA_FORMATTING_AND_CONDITIONAL_FORMATTING:

```
<span class="xdExpressionBox xdDataBindingUI xdBehavior_Formatting" title="ANY_STRING"
(tabIndex="-1")? xd:xctname="ExpressionBox" xd:CtrlId="CONTROL_ID" xd:disableEditing="yes"
xd:binding="EXPRESSION_BOX_XPATH1" xd:datafmt="DATA_FMT_CTRL_EXPBOX" (xd:num="")?>
    CHECK FOR GETDOM BEGIN1
    <xsl:attribute name="style">EXPRESSION_BOX_STYLE
       <xsl:choose>
            (<xsl:when
test="BOOLEAN_XPATH_EXPRESSIONx">LEAF_CONTROL_CONDITIONAL_FORMATTING_CAPTION</xsl:when>)+
        </xsl:choose>
    </xsl:attribute>
    (<xsl:attribute name="xd:num">
        <xsl:value-of select="EXPRESSION BOX XPATH1" />
    </xsl:attribute>)?
    <xsl:choose>
        <xsl:when test="function-available('xdFormatting:formatString')">
            <xsl:value-of select="xdFormatting:formatString(EXPRESSION BOX XPATH1,
DATA_FMT_CTRL_EXPBOX)"
        </xsl:when>
        <xsl:otherwise>
            <xsl:value-of select="EXPRESSION_BOX_XPATH1" />
        </xsl:otherwise>
    </xsl:choose>
    CHECK FOR GETDOM END1
</span>
```

EXPRESSION_BOX_XPATH: LEAF_XPATH or **STRING_XPATH_EXPRESSION.**

EXPRESSION_BOX_OVERFLOW_Y: OVERFLOW-Y: auto

EXPRESSION_BOX_OVERFLOW_X: OVERFLOW-X: auto | OVERFLOW-X: visible

EXPRESSION_BOX_STYLE: Semicolon-delimited list of (**STYLE_WIDTH**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_BORDER**?, **STYLE_PADDING**?, **STYLE_VERTICAL_ALIGN**?, **EXPRESSION_BOX_OVERFLOW_Y**?, **EXPRESSION_BOX_OVERFLOW_X**?, **STYLE_FONT**?, **STYLE_MARGIN**?, **STYLE_HEIGHT**?, **STYLE_TEXT_DECORATION**?, **STYLE_WRAP**?, **STYLE_COLOR**?, **STYLE_DIRECTION**?)

Control-specific attributes used by the expression box control are as follows:

- xd:binding (section 2.4.2.6)
- xd:CrtlId (section 2.4.2.10)
- xd:datafmt (section 2.4.2.11)
- xd:disableEditing (section 2.4.2.12)
- xd:num (section 2.4.2.26)
- xd:xctname (section 2.4.2.35)

2.4.1.11 File Attachment Control

The file attachment control enables users to attach a file to a form.

FILE_ATTACHMENT:

```
<span class="xdFileAttachment" hideFocus="1" style="FILE ATTACHMENT STYLE" tabStop="true"
xd:binding="LEAF_XPATH" xd:boundProp="xd:inline" tabIndex="TAB_INDEX"
xd:xctname="FileAttachment" xd:CtrlId="CONTROL ID" (title="ANY STRING")?
(accessKey="SINGLE_CHARACTER")? (xd:disableEditing="yes")?/>
    <xsl:if test="function-available('xdImage:getImageUrl')">
        <xsl:attribute name="src">
            <xsl:value-of select="xdImage:getImageUrl(LEAF_XPATH)"/>
        </xsl:attribute>
    </xsl:if>
</span>
```

FILE_ATTACHMENT_STYLE: Semicolon-delimited list of (**STYLE_SIZE**, **STYLE_TEXT_DECORATION**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_BORDER**?, **STYLE_FONT**?, **STYLE_COLOR**?, **STYLE_VERTICAL_ALIGN**?)

LEAF_XPATH: MUST point to an XML node in the main data source.

Control-specific attributes used by the file attachment control are as follows:

- xd:binding (section 2.4.2.6)
- xd:boundProp (section 2.4.2.9)
- xd:CrtlId (section 2.4.2.10)
- xd:disableEditing (section 2.4.2.12)
- xd:xctname (section 2.4.2.35)

The **xdImage:getImageUrl** XSL function extension, as specified in section 2.4.3.5, is used by the file attachment control.

2.4.1.12 Hyperlink Control

The hyperlink control allows the user to create a hyperlink that navigates the default browser to a specified URL. A **HYPERLINK** MUST have one of the symbols in the following table.

Symbol	Description
SIMPLE_HYPERLINK	Hyperlinks are read-only controls that open a new browser window to another web location. They are composed of link target and display text.
HYPERLINK_WITH_DYNAMIC_LINK	Similar to **SIMPLE_HYPERLINK**, with the exception that the hyperlink's target link is dynamically populated from a node in the form.
HYPERLINK_WITH_DYNAMIC_TEXT	Similar to **SIMPLE_HYPERLINK**, with the exception that the hyperlink's display text is dynamically populated from a node in the form (1).
HYPERLINK_WITH_DYNAMIC_LINK_AND_ DYNAMIC_TEXT	Similar to **HYPERLINK_WITH_DYNAMIC_LINK** and **HYPERLINK_WITH_DYNAMIC_TEXT**.

SIMPLE_HYPERLINK:

```
<a href="ANY_STRING" (title="ANY_STRING")? (accessKey="SINGLE_CHARACTER")?
(tabIndex="TAB_INDEX")? xd:disableEditing="yes">ANCHOR_TEXT</a>

HYPERLINK_WITH_DYNAMIC_LINK ::=
<a class="xdDataBindingUI" (title="ANY_STRING")? (accessKey="SINGLE_CHARACTER")?
(tabIndex="TAB INDEX")? xd:disableEditing="yes">
    CHECK_FOR_GETDOM_BEGIN1
    <xsl:attribute name="href">
        <xsl:value-of select="LEAF_XPATH"/>
    </xsl:attribute>
ANCHOR TEXT
    CHECK_FOR_GETDOM_END1
</a>
```

HYPERLINK_WITH_DYNAMIC_TEXT:

```
<span class="xdHyperlink" hideFocus="1" (title="ANY_STRING")?
style="DYNAMIC HYPERLINK TEXT STYLE" xd:xctname="hyperlink">
    <a class="xdDataBindingUI" (title="ANY_STRING")? (accessKey="SINGLE_CHARACTER")?
(tabIndex="TAB INDEX")? href="ANY STRING" xd:disableEditing="yes" xd:CtrlId="CONTROL ID">
    CHECK_FOR_GETDOM_BEGIN1
        <xsl:value-of select="LEAF_XPATH"/>
    CHECK_FOR_GETDOM_END1
    </a>
</span>
```

HYPERLINK_WITH_DYNAMIC_LINK_AND_DYNAMIC_TEXT:

```
<span class="xdHyperlink" hideFocus="1" (title="ANY_STRING")?
style="DYNAMIC_HYPERLINK_TEXT_STYLE" xd:xctname="hyperlink">
    <a class="xdDataBindingUI" (title="ANY_STRING")? (accessKey="SINGLE_CHARACTER")?
(tabIndex="TAB INDEX")? xd:disableEditing="yes" xd:CtrlId="CONTROL ID">
    CHECK_FOR_GETDOM_BEGIN1
        <xsl:attribute name="href">
            <xsl:value-of select="LEAF_XPATH1"/>
        </xsl:attribute>
        <xsl:value-of select="LEAF_XPATH2"/>
    CHECK FOR GETDOM END1
    </a>
</span>
```

DYNAMIC_HYPERLINK_TEXT_STYLE: Semicolon-delimited list of (**OVERFLOW**: visible, **STYLE_WIDTH**?, **STYLE_TEXT_ALIGN**?, **STYLE_BORDER**?, **STYLE_FONT**?, **STYLE_VERTICAL_ALIGN**?, **STYLE_TEXT_DECORATION**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_COLOR**?)

Control-specific attributes used by the hyperlink control are as follows:

```
xd:CrtlId (section 2.4.2.10)
xd:disableEditing (section 2.4.2.12)
xd:xctname (section 2.4.2.35)
```

2.4.1.13 List Box Control

The list box control enables the user to select a single value from a list of options that can be specified manually by the form template designer, or is populated from a data source. A **LIST_BOX** MUST have one of the symbols in the following table.

Symbol	Description
LIST_BOX_WITH_MANUAL_ENTRIES	The list box displays selection options that have been manually specified by the form template designer.
LIST_BOX_WITH_LOOKUP_ENTRIES	The list box displays selection options that are populated from a data source (2).
LIST_BOX_WITH_UNIQUE_LOOKUP_ENTRIES	The list box displays only unique selection options that are populated from a data source (2).

LIST_BOX_CONDITIONAL_FORMATTING:

```
(<xsl:attribute name="style">LIST_BOX_STYLE<xsl:choose>
        (<xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_DISPLAY_NONE</xsl:when>|
        <xsl:when test="BOOLEAN_XPATH_EXPRESSION"/>|
        <xsl:when
test="BOOLEAN_XPATH_EXPRESSION">LIST_BOX_CONDITIONAL_FORMATTING_STYLE</xsl:when>)+
    </xsl:choose>
</xsl:attribute>)?
(<xsl:choose>
    (<xsl:when test="BOOLEAN_XPATH_EXPRESSION"/>|
    <xsl:when test="BOOLEAN_XPATH_EXPRESSION">
        <xsl:attribute name="disabled">true</xsl:attribute>
    </xsl:when>)+
</xsl:choose>)?
```

LIST_BOX_WITH_MANUAL_ENTRIES:

```
<select class="xdListBox xdBehavior_Select" title="ANY_STRING" size="3"
(tabindex="TAB INDEX")? xd:CtrlId="CONTROL ID" xd:xctname="ListBox" xd:binding="LEAF XPATH1"
xd:boundProp="value" (style="LIST_BOX_STYLE")? (xd:postbackModel="POSTBACKMODEL")?
(accessKey="SINGLE_CHARACTER")?>
    CHECK_FOR_GETDOM_BEGIN1
    LIST BOX CONDITIONAL FORMATTING?
    <xsl:attribute name="value">
        <xsl:value-of select="LEAF XPATH1"/>
    </xsl:attribute>
    ((<option>
        <xsl:if test="LEAF_XPATH1=&amp;quot;&amp;quot;">
            <xsl:attribute name="selected">selected</xsl:attribute>
        </xsl:if>OPTION_DISPLAY_VALUE?</option>)|
    (<option value="OPTION VALUE1">
        <xsl:if test="LEAF_XPATH1=&amp;quot;OPTION_VALUE1&amp;quot;">
            <xsl:attribute name="selected">selected</xsl:attribute>
        </xsl:if>OPTION_DISPLAY_VALUE?</option>))+
    CHECK FOR GETDOM END1
</select>
```

LIST_BOX_WITH_LOOKUP_ENTRIES:

```
<select class="xdListBox xdBehavior_Select" title="ANY_STRING" size="3"
(tabindex="TAB_INDEX")? xd:CtrlId="CONTROL_ID" xd:xctname="ListBox" xd:binding="LEAF_XPATH1"
xd:boundProp="value" (value="ANY_STRING")? (style="LIST_BOX_STYLE")?
(xd:postbackModel="POSTBACKMODEL")? (accessKey="SINGLE CHARACTER")?>
    LIST_BOX_CONDITIONAL_FORMATTING?
    <xsl:attribute name="value">
        <xsl:value-of select="LEAF_XPATH1"/>
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="function-available('xdXDocument:GetDOM')">
            <option/>
            <xsl:variable name="val" select="LEAF_XPATH1"/>
```

```
            <xsl:if
test="not(GROUP_XPATH1/RELATIVE_REPEATING_GROUP_XPATH1[RELATIVE_LEAF_XPATH1 =$val] or
$val='')">
                <option selected="selected">
                    <xsl:attribute name="value">
                        <xsl:value-of select="$val"/>
                    </xsl:attribute>
                    <xsl:value-of select="$val"/>
                </option>
            </xsl:if>
            <xsl:for-each select="GROUP XPATH1/RELATIVE REPEATING GROUP XPATH1">
                <option>
                    <xsl:attribute name="value">
                        <xsl:value-of select="RELATIVE_LEAF_XPATH1"/>
                    </xsl:attribute>
                    <xsl:if test="$val=RELATIVE_LEAF_XPATH1">
                        <xsl:attribute name="selected">selected</xsl:attribute>
                    </xsl:if>
                    <xsl:value-of select="RELATIVE LEAF XPATH2"/>
                </option>
            </xsl:for-each>
        </xsl:when>
        <xsl:otherwise>
            <option>
                <xsl:value-of select="LEAF_XPATH1"/>
            </option>
        </xsl:otherwise>
    </xsl:choose>
</select>
```

LIST_BOX_WITH_UNIQUE_LOOKUP_ENTRIES:

```
<select class="xdListBox xdBehavior_Select" title="ANY_STRING" size="3"
(tabindex="TAB_INDEX")? xd:CtrlId="CONTROL_ID" xd:xctname="ListBox" xd:binding="LEAF_XPATH1"
xd:boundProp="value" (value="ANY_STRING")? (style="LIST_BOX_STYLE")?
(xd:postbackModel="POSTBACKMODEL")? (accessKey="SINGLE_CHARACTER")?>
    LIST BOX CONDITIONAL FORMATTING?
    <xsl:attribute name="value">
        <xsl:value-of select="LEAF_XPATH1"/>
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="function-available('xdXDocument:GetDOM')">
            <option/>
            <xsl:variable name="val" select="LEAF_XPATH1"/>
            <xsl:if
test="not(GROUP_XPATH1/RELATIVE_REPEATING_GROUP_XPATH1[RELATIVE_LEAF_XPATH1=$val] or
$val='')">
                <option selected="selected">
                    <xsl:attribute name="value">
                        <xsl:value-of select="$val"/>
                    </xsl:attribute>
                    <xsl:value-of select="$val"/>
                </option>
            </xsl:if>
            <xsl:variable name="items">
                <xsl:copy-of select="GROUP_XPATH1/RELATIVE_REPEATING_GROUP_XPATH1"/>
            </xsl:variable>
            <xsl:variable name="uniqueItems" select="msxsl:node-
set($items)/*[not((RELATIVE_LEAF_XPATH2=
preceding::RELATIVE REPEATING GROUP XPATH1/RELATIVE LEAF XPATH2)|(.=
preceding::RELATIVE_REPEATING_GROUP_XPATH1))]"/>
            <xsl:for-each select="$uniqueItems">
                <option>
                    <xsl:attribute name="value">
                        <xsl:value-of select="RELATIVE_LEAF_XPATH1"/>
                    </xsl:attribute>
                    <xsl:if test="$val=RELATIVE_LEAF_XPATH1">
```

```
                        <xsl:attribute name="selected">selected</xsl:attribute>
                    </xsl:if>
                    <xsl:value-of select="RELATIVE_LEAF_XPATH2"/>
                </option>
            </xsl:for-each>
        </xsl:when>
        <xsl:otherwise>
            <option>
                <xsl:value-of select="LEAF_XPATH1"/>
            </option>
        </xsl:otherwise>
    </xsl:choose>
</select>
```

LIST_BOX_STYLE: Semicolon-delimited list of (STYLE_SIZE?, STYLE_MARGIN?, STYLE_DIRECTION?, STYLE_TEXT_DECORATION?, STYLE_BACKGROUND_COLOR?, STYLE_FONT?, STYLE_COLOR?, STYLE_VERTICAL_ALIGN?)

LIST_BOX_CONDITIONAL_FORMATTING_STYLE: Semicolon-delimited list of (STYLE_TEXT_DECORATION?, STYLE_BACKGROUND_COLOR?, STYLE_FONT?, STYLE_COLOR?)

Control-specific attributes used by the list box control are as follows:

- xd:binding (section 2.4.2.6)
- xd:boundProp (section 2.4.2.9)
- xd:CrtlId (section 2.4.2.10)
- xd:postbackModel (section 2.4.2.29)
- xd:xctname (section 2.4.2.35)

The **xdXDocument:GetDom** XSL function, as specified in section 2.4.3.9.2, extension is used by the list box control.

2.4.1.14 Option Button Control

An option button is a bi-state control that has a value when selected and no value when not selected. Option button controls are meant to be used in groups, with selection among the option buttons in the group being mutually exclusive. An **OPTION_BUTTON** MUST have one of the symbols in the following table.

Symbol	Description
SIMPLE_OPTION_BUTTON	An option button is a binary state control. It is found in a group (1) of option buttons where only a single member of the group (1) can be selected at any one time.
OPTION_BUTTON_WITH_CONDITIONAL_FORMATTING	The same as a **SIMPLE_OPTION_BUTTON**, but also allows the button to be conditionally disabled.

SIMPLE_OPTION_BUTTON:

```
<input class="xdBehavior_Boolean" title="ANY_STRING1" type="radio" name="{generate-
id(LEAF_XPATH1)}" (accessKey="SINGLE_CHARACTER")? xd:binding="LEAF_XPATH1"
xd:boundProp="xd:value" (xd:onValue="(ISO DIGIT+)|(&quot;ANY STRING2&quot;)")?
(tabIndex="LEAF_CONTROL_TAB_INDEX")? xd:xctname="OptionButton" xd:CtrlId="CONTROL_ID"
(xd:postbackModel="POSTBACKMODEL")? (style="OPTION_BUTTON_STYLE")?>
    CHECK_FOR_GETDOM_BEGIN1
    <xsl:attribute name="xd:value">
        <xsl:value-of select="LEAF XPATH1" />
    </xsl:attribute>
```

```
    <xsl:if test="BOOLEAN_XPATH_EXPRESSION">
        <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
    </xsl:if>
    CHECK_FOR_GETDOM_END1
</input>
ANY_STRING3
```

OPTION_BUTTON_WITH_CONDITIONAL_FORMATTING:

```
<input class="xdBehavior_Boolean" title="ANY_STRING1" type="radio" name="{generate-
id(LEAF_XPATH1)}" (accessKey="SINGLE_CHARACTER")? xd:binding="LEAF_XPATH1"
xd:boundProp="xd:value" (xd:onValue="(ISO DIGIT+)|(&quot;ANY STRING2&quot;)")?
(tabIndex="LEAF_CONTROL_TAB_INDEX")? xd:xctname="OptionButton" xd:CtrlId="CONTROL_ID"
(xd:postbackModel="POSTBACKMODEL")? (style="OPTION BUTTON STYLE")?>
    CHECK_FOR_GETDOM_BEGIN1
    <xsl:choose>
        (<xsl:when test="BOOLEAN_XPATH_EXPRESSIONY">
            <xsl:attribute name="disabled">true</xsl:attribute>
        </xsl:when>)*
    </xsl:choose>
    <xsl:attribute name="xd:value">
        <xsl:value-of select="LEAF_XPATH1" />
    </xsl:attribute>
    <xsl:if test="BOOLEAN XPATH EXPRESSION">
        <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
    </xsl:if>
    CHECK_FOR_GETDOM_END1
</input>
ANY STRING3
```

OPTION_BUTTON_STYLE: Semicolon-delimited list of (STYLE_MARGIN?, STYLE_FONT?, STYLE_VERTICAL_ALIGN?, STYLE_BORDER?, STYLE_BACKGROUND_COLOR?, STYLE_COLOR?, STYLE_TEXT_DECORATION?, STYLE_WIDTH?, STYLE_HEIGHT?)

Control-specific attributes used by the option button control are as follows:

- xd:binding (section 2.4.2.6)
- xd:boundProp (section 2.4.2.9)
- xd:CrtlId (section 2.4.2.10)
- xd:onValue (section 2.4.2.28)
- xd:postbackModel (section 2.4.2.29)
- xd:value (section 2.4.2.34)
- xd:xctname (section 2.4.2.35)

2.4.1.15 Repeating Section Control

A repeating section control acts as a container for other controls that can appear multiple times in the same form. A **REPEATING_SECTION** MUST have one of the symbols in the following table.

Symbol	Description
REPEATING_SECTION_CALL	The first part of the repeating section control that indicates where the repeating section is rendered.
REPEATING_SECTION_BODY_	The second part of the repeating section that defines the

Symbol	Description
WITHOUT_CONDITIONAL_HIDING	properties of the repeating section and its content.
REPEATING_SECTION_BODY_ WITHOUT_CONDITIONAL_HIDING	The second part of the repeating section that defines the properties of the repeating section and its content. This part is used when the section is conditionally hidden.

A **REPEATING_SECTION** MUST consist of a **REPEATING_SECTION_CALL** at the point in the XSL where the control appears, which is either the body of the main template or another XSL template, and a **REPEATING_SECTION_BODY** that defines the section and appears as a separate XSL template. **TEMPLATE_MODE_ID** values in **REPEATING_SECTION_CALL** and **REPEATING_SECTION_BODY** MUST match.

REPEATING_SECTION_CALL: SIMPLE_SECTION_CALL or

```
(CHECK_FOR_GETDOM_BEGIN1
<xsl:apply-templates select="(GROUP_XPATH/)?RELATIVE_REPEATING_GROUP_XPATH"
mode="TEMPLATE MODE ID1"/>
    (<div class="optionalPlaceholder" xd:xmlToEdit="XML_TO_EDIT_NAME" tabIndex="TABINDEX"
xd:action="xCollection::insert" align="ALIGN" style="STYLE WIDTH">ANY STRING</div>)?
CHECK_FOR_GETDOM_END1)
```

REPEATING_SECTION_BODY:
REPEATING_SECTION_BODY_WITHOUT_CONDITIONAL_HIDING or
REPEATING_SECTION_BODY_WITH_CONDITIONAL_HIDING.

REPEATING_SECTION_BODY_WITHOUT_CONDITIONAL_HIDING:

```
<xsl:template match="RELATIVE_REPEATING_GROUP_XPATH" mode="TEMPLATE_MODE_ID1">
    <div class="xdRepeatingSection xdRepeating" title="ANY_STRING" (style="SECTION_STYLE")?
align="ALIGN" xd:xctname="RepeatingSection" xd:CtrlId="CONTROL_ID" (tabIndex="-1")?
(xd:postbackModel="POSTBACKMODEL")?>
        XML_HTML_4_1_WITH_CONTROLS
        (<xsl:attribute name="style">SECTION STYLE(<xsl:choose>
                (<xsl:when test="BOOLEAN_XPATH_EXPRESSION"/>|
                <xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_DISPLAY_NONE</xsl:when>|
                <xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_BACKGROUND_COLOR</xsl:when>)+
            </xsl:choose>)?
            (<xsl:if
test="BOOLEAN XPATH EXPRESSION">STYLE DISABLE CHILD XML TO EDIT</xsl:if>)*
        </xsl:attribute>)?
    </div>
</xsl:template>
```

REPEATING_SECTION_BODY_WITH_CONDITIONAL_HIDING:

```
<xsl:template match="RELATIVE_REPEATING_GROUP_XPATH" mode="TEMPLATE_MODE_ID1">
    <xsl:if test="BOOLEAN XPATH EXPRESSION">
        <div class="xdRepeatingSection xdRepeating" title="ANY_STRING"
(style="SECTION_STYLE")? align="ALIGN" xd:xctname="RepeatingSection" xd:CtrlId="CONTROL_ID"
(tabIndex="-1")? (xd:postbackModel="POSTBACKMODEL")?>
            XML HTML 4 1 WITH CONTROLS
            (<xsl:attribute name="style">SECTION_STYLE(<xsl:choose>
                    (<xsl:when test="BOOLEAN XPATH EXPRESSION"/>|
                    <xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_DISPLAY_NONE</xsl:when>|
                    <xsl:when
test="BOOLEAN_XPATH_EXPRESSION">STYLE_BACKGROUND_COLOR</xsl:when>)+
                </xsl:choose>)?
                (<xsl:if
test="BOOLEAN XPATH EXPRESSION">STYLE DISABLE CHILD XML TO EDIT</xsl:if>)*
            </xsl:attribute>)?
        </div>
    </xsl:if>
```

```
</xsl:template>
```

Control-specific attributes used by the repeating section control are as follows:

- xd:action (section 2.4.2.1)
- xd:CrtlId (section 2.4.2.10)
- xd:postbackModel (section 2.4.2.29)
- xd:xctname (section 2.4.2.35)
- xd:xmlToEdit (section 2.4.2.36)

2.4.1.16 Repeating Table Control

A repeating table control acts as a container for other controls, and can appear multiple times in an instance of a form. It has a tabular format. A **REPEATING_TABLE** MUST have one of the symbols in the following table.

Symbol	Description
SIMPLE_REPEATING_TABLE	A repeating table is a structural control, which overloads the HTML **table** element. It is possible to dynamically add and remove rows.
REPEATING_TABLE_WITH_CONDITIONAL_FORMATTING	The same as **SIMPLE_REPEATING_TABLE**, with the ability to conditionally change the background color and disable adding or removing rows.
REPEATING_TABLE_ROWS_WITH_CONDITIONALVISIBILITY	The same as **TABLE_ROW**, but allows for changing the background color.

SIMPLE_REPEATING_TABLE:

```
<table class="xdRepeatingTable msoUcTable" title="ANY STRING" style="TABLE-LAYOUT: fixed;
STYLE_WIDTH; BORDER-TOP-STYLE: none; BORDER-RIGHT-STYLE: none; BORDER-LEFT-STYLE: none;
BORDER-COLLAPSE: collapse; WORD-WRAP: break-word; BORDER-BOTTOM-STYLE: none" border="1"
xd:CtrlId="CONTROL_ID" (xd:postBackModel="POST_BACK_MODEL_VALUE")? WIDTH?>
    <colgroup>
        TABLE_COLUMN+
    </colgroup>
    (<tbody (valign="VALIGN")? (style="STYLE_DIRECTION")? class="xdTableHeader">
        TABLE_ROW*
    </tbody>)?
    <tbody (style="STYLE DIRECTION")? xd:xctname="RepeatingTable">
    CHECK_FOR_GETDOM_BEGIN1
        <xsl:for-each select="GROUP XPATH">
            TABLE_ROW*
        </xsl:for-each>
    CHECK_FOR_GETDOM_END1
    </tbody>
    (<tbody (valign="VALIGN")? (style="STYLE DIRECTION")? class="xdTableFooter">
        TABLE_ROW*
    </tbody>)?
</table>
(<div class="optionalPlaceholder" xd:xmlToEdit="XML TO EDIT NAME" tabIndex="TAB INDEX"
xd:action="xCollection::insert" style="STYLE_WIDTH">ANY_STRING</div>)?
```

REPEATING_TABLE_WITH_CONDITIONAL_FORMATTING:

```
<table class="xdRepeatingTable msoUcTable" title="ANY_STRING" style="TABLE-LAYOUT: fixed;
STYLE_WIDTH; BORDER-TOP-STYLE: none; BORDER-RIGHT-STYLE: none; BORDER-LEFT-STYLE: none;
```

```
BORDER-COLLAPSE: collapse; WORD-WRAP: break-word; BORDER-BOTTOM-STYLE: none" border="1"
xd:CtrlId="CONTROL ID" (xd:postBackModel="POST BACK MODEL VALUE")?>
    <colgroup>
        TABLE COLUMN+
    </colgroup>
    (<tbody (valign="VALIGN")? (style="STYLE_DIRECTION")? class="xdTableHeader">
        TABLE_ROW*
    </tbody>)?
    <tbody (style="STYLE_DIRECTION")? xd:xctname="RepeatingTable">
    CHECK FOR GETDOM BEGIN1
        <xsl:for-each select="GROUP_XPATH">
            REPEATING_TABLE_ROWS_WITH_CONDITIONALVISIBILITY |
TABLE_ROW_WITH_CONDITIONAL_FORMATTING*
        </xsl:for-each>
    CHECK_FOR_GETDOM_END1
    </tbody>
    (<tbody (valign="VALIGN")? (style="STYLE_DIRECTION")? class="xdTableFooter">
        TABLE_ROW*
    </tbody>)?
</table>
(<div class="optionalPlaceholder" xd:xmlToEdit="XML_TO_EDIT_NAME" tabIndex="TAB_INDEX"
xd:action="xCollection::insert" style="STYLE_WIDTH">ANY_STRING</div>)?
```

REPEATING_TABLE_ROWS_WITH_CONDITIONALVISIBILITY:

```
<xsl:if test="not(BOOLEAN_XPATH_EXPRESSION)">
TABLE_ROW_WITH_CONDITIONAL_FORMATTING*
</xsl:if>


TABLE_ROW_WITH_CONDITIONAL_FORMATTING ::=
<tr>
    (<xsl:attribute name="style">
        MIN HEIGHT?
        <xsl:choose>
           (<xsl:when
test="BOOLEAN_XPATH_EXPRESSIONx">CONTAINER_CONDITIONAL_FORMATTING</xsl:when>)+
        </xsl:choose>
        (<xsl:if test="BOOLEAN_XPATH_EXPRESSION">STYLE_DISABLE_CHILD_XML_TO_EDIT</xsl:if>)*
    </xsl:attribute>)?
    (TABLE CELL)+
</tr>
```

CONTAINER_CONDITIONAL_FORMATTING: Semicolon-delimited list of (STYLE_DISPLAY_NONE?, STYLE_BACKGROUND_COLOR?)

Control-specific attributes used by the repeating table control are as follows:

- xd:action (section 2.4.2.1)
- xd:CrtlId (section 2.4.2.10)
- xd:postbackModel (section 2.4.2.29)
- xd:xctname (section 2.4.2.35)
- xd:xmlToEdit (section 2.4.2.36)

2.4.1.17 Rich Text Box Control

The rich text box control allows the user to enter **rich text**, such as formatted text, tables, hyperlinks, and images, in the form. A **RICH_TEXT_BOX** MUST have one of the symbols in the following table.

Symbol	Description
RICH_TEXT_BOX_PLAIN	The rich text box
RICH_TEXT_BOX_WITH_PLACEHOLDER_TEXT	The rich text box shows place holder text as long as the field it is bound to contains no data. (Important: Place holder text is not supported on the form server).

RICH_TEXT_BOX_PLAIN:

```
<span class="xdRichTextBox(RTL)?" hideFocus="1" title="ANY_STRING" xd:binding="LEAF_XPATH"
(tabIndex="TAB_INDEX")? xd:xctname="RichText" xd:CtrlId="CONTROL_ID"
(style="RICH TEXT BOX STYLE")? (accessKey="SINGLE CHARACTER")?
(xd:postbackModel="POSTBACKMODEL")? (INPUT_SCOPE)? (contentEditable="true" |
xd:disableEditing="yes")>
    CHECK_FOR_GETDOM_BEGIN1
    RICH_TEXT_BOX_CONDITIONAL_FORMATTING?
    <xsl:copy-of select="LEAF_XPATH/node()"/>
    CHECK FOR GETDOM END1
</span>

RICH_TEXT_BOX_WITH_PLACEHOLDER_TEXT ::=
<span class="xdRichTextBox(RTL)? xdBehavior_GhostedText" hideFocus="1" title="ANY_STRING"
xd:binding="LEAF XPATH" (tabIndex="TAB INDEX")? xd:xctname="RichText" xd:CtrlId="CONTROL ID"
(style="RICH_TEXT_BOX_STYLE")? (accessKey="SINGLE_CHARACTER")?
(xd:postbackModel="POSTBACKMODEL")? (INPUT SCOPE)? (contentEditable="true" |
xd:disableEditing="yes")>
    CHECK_FOR_GETDOM_BEGIN1
    RICH_TEXT_BOX_CONDITIONAL_FORMATTING?
    (<xsl:choose>
        <xsl:when test="not(string(LEAF_XPATH) or LEAF_XPATH/node())">
            <xsl:attribute name="xd:ghosted">true</xsl:attribute>ANY STRING</xsl:when>
        <xsl:otherwise>
            <xsl:copy-of select="LEAF_XPATH/node()"/>
        </xsl:otherwise>
    </xsl:choose>) | (<xsl:copy-of select="LEAF XPATH/node()"/>)
    CHECK FOR GETDOM END1
</span>
```

RICH_TEXT_BOX_SCROLLING_STYLE: OVERFLOW-X: visible | OVERFLOW-Y: scroll; OVERFLOW-X: scroll | OVERFLOW-Y: auto; OVERFLOW-X: auto | OVERFLOW-Y: auto | OVERFLOW-Y: hidden.

RICH_TEXT_BOX_STYLE: Semicolon-delimited list of (**RICH_TEXT_BOX_SCROLLING_STYLE?**, **STYLE_WRAP?**, **STYLE_SIZE?**, **STYLE_MARGIN?**, **STYLE_PADDING?**, **STYLE_TEXT_DECORATION?**, **STYLE_BACKGROUND_COLOR?**, **STYLE_BORDER?**, **STYLE_FONT?**, **STYLE_COLOR?**, **STYLE_TEXT_ALIGN?**, **STYLE_DIRECTION?**)

RICH_TEXT_BOX_CONDITIONAL_FORMATTING_STYLE: Semicolon-delimited list of (**STYLE_BACKGROUND_COLOR?, STYLE_COLOR?**)

RICH_TEXT_BOX_CONDITIONAL_FORMATTING:

```
(<xsl:attribute name="style">RICH_TEXT_BOX_STYLE<xsl:choose>
        (<xsl:when test="BOOLEAN XPATH EXPRESSION">STYLE DISPLAY NONE</xsl:when>|
        <xsl:when test="BOOLEAN_XPATH_EXPRESSION"/>|
        <xsl:when test="BOOLEAN_XPATH_EXPRESSION">RICH_TEXT_BOX_CONDITIONAL_FORMATTING_STYLE
        </xsl:when>)+
    </xsl:choose>
</xsl:attribute>)?
(<xsl:choose>
    (<xsl:when test="BOOLEAN XPATH EXPRESSION"/>|
    <xsl:when test="BOOLEAN_XPATH_EXPRESSION">
        <xsl:attribute name="contentEditable">false</xsl:attribute>
    </xsl:when>)+
</xsl:choose>)?
```

Control-specific attributes used by the rich text box control are as follows:

- xd:allownonmatching (section 2.4.2.2)
- xd:binding (section 2.4.2.6)
- xd:CrtlId (section 2.4.2.10)
- xd:disableEditing (section 2.4.2.12)
- xd:ghosted (section 2.4.2.15)
- xd:inputScopeId (section 2.4.2.21)
- xd:postbackModel (section 2.4.2.29)
- xd:xctname (section 2.4.2.35)

2.4.1.18 Section Control and Optional Section Control

A section control acts as a container for other controls. An optional section control has the same functionality as a regular section, but it can also be deleted or inserted by the user. A **SECTION** MUST have one of the symbols in the following table.

Symbol	Description
SIMPLE_SECTION_CALL	The first part of the section that indicates where the section control is rendered.
OPTIONAL_SECTION_CALL	The first part of the optional section that indicates where the optional section control is rendered. This part can be configured to not allow the user to delete or insert the optional section.
SIMPLE_SECTION_BODY	The second part of the section or optional section that defines the properties of the section and its content.
SIMPLE_SECTION_BODY_WITH_DIGITAL_SIGNATURE	The second part of the section that defines the properties of the section and its content. It also contains a digital signature component that enables the user to sign it.
SIMPLE_SECTION_BODY_WITH_CONDITIONAL_HIDING	The second part of the section or optional section that defines the properties of the section and its content. This part is used when the section is conditionally hidden.
SIMPLE_SECTION_BODY_WITH_CONDITIONAL_HIDING_AND_DIGITAL_SIGNATURE	The second part of the section or optional section that defines the properties of the section and its content. This part is used when the section is conditionally hidden. It also contains a digital signature component that enables the user to sign it.

A **SECTION** MUST consist of a **SIMPLE_SECTION_CALL** at the point in the XSL where the control appears, which is the body of the main template or another XSL template, and a **SECTION_BODY** that defines the section and appears as a separate XSL template. **TEMPLATE_MODE_ID** values in **SIMPLE_SECTION_CALL** and **SECTION_BODY** MUST match.

An **OPTIONAL_SECTION** MUST consist of an **OPTIONAL_SECTION_CALL** at the point in the XSL where the control appears, which is the body of the main template or another XSL template, and a **SECTION_BODY** that defines the section and appears as a separate XSL template. **TEMPLATE_MODE_ID** values in **OPTIONAL_SECTION_CALL** and **SECTION_BODY** MUST match.

SIMPLE_SECTION_CALL:

```
CHECK_FOR_GETDOM_BEGIN1
<xsl:apply-templates select="(GROUP_XPATH/)?RELATIVE_GROUP_XPATH" mode="TEMPLATE_MODE_ID1"/>
CHECK_FOR_GETDOM_END1
```

```
OPTIONAL_SECTION_CALL ::= SIMPLE_SECTION_CALL |
(CHECK_FOR_GETDOM_BEGIN1
<xsl:choose>
    <xsl:when test="OPTIONAL_SECTION_XPATH1">
        <xsl:apply-templates select="OPTIONAL SECTION XPATH1" mode="TEMPLATE MODE ID1"/>
    </xsl:when>
    <xsl:otherwise>
        <div class="optionalPlaceholder" xd:xmlToEdit="XML_TO_EDIT_NAME" tabIndex="TABINDEX"
align="ALIGN" style="STYLE_WIDTH">ANY_STRING</div>
    </xsl:otherwise>
</xsl:choose>
CHECK_FOR_GETDOM_END1)
```

OPTIONAL_SECTION_XPATH: (GROUP_XPATH/)?RELATIVE_GROUP_XPATH

SECTION_STYLE: Semicolon-delimited list of (STYLE_SIZE?, STYLE_DIRECTION?, STYLE_BACKGROUND_COLOR?, STYLE_BORDER?, STYLE_MARGIN?, STYLE_PADDING?)

DIGITAL_SIGNATURE_STYLE: Semicolon-delimited list of (STYLE_MARGIN?, BEHAVIOR: url (#default#SignaturesInDocUI), STYLE_WIDTH?)

SECTION_BODY: SIMPLE_SECTION_BODY or SIMPLE_SECTION_BODY_WITH_DIGITAL_SIGNATURE or SIMPLE_SECTION_BODY_WITH_CONDITIONAL_HIDING or SIMPLE_SECTION_BODY_WITH_CONDITIONAL_HIDING_AND_DIGITAL_SIGNATURE.

SIMPLE_SECTION_BODY:

```
<xsl:template match="RELATIVE_GROUP_XPATH" mode="TEMPLATE_MODE_ID1">
    <div class="xdSection xdRepeating" title="ANY STRING" (style="SECTION STYLE")?
align="ALIGN" xd:xctname="Section" xd:CtrlId="CONTROL_ID" (tabIndex="-1")?
(xd:postbackModel="POSTBACKMODEL")?>
        XML HTML 4 1 WITH CONTROLS
        (<xsl:attribute name="style">SECTION_STYLE(<xsl:choose>
                (<xsl:when test="BOOLEAN XPATH EXPRESSION"/>|
                <xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_DISPLAY_NONE</xsl:when>|
                <xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_BACKGROUND_COLOR</xsl:when>)+
            </xsl:choose>)?
            (<xsl:if
test="BOOLEAN_XPATH_EXPRESSION">STYLE_DISABLE_CHILD_XML_TO_EDIT</xsl:if>)*
        </xsl:attribute>)?
    </div>
</xsl:template>
```

SIMPLE_SECTION_BODY_WITH_DIGITAL_SIGNATURE:

```
<xsl:template match="RELATIVE_GROUP_XPATH" mode="TEMPLATE_MODE_ID1">
    <div class="xdSection xdRepeating" title="ANY STRING" (style="SECTION STYLE")?
align="ALIGN" xd:xctname="Section" xd:CtrlId="CONTROL_ID" xd:SignedSectionName="ANY_STRING1"
(tabIndex="-1")? (xd:postbackModel="POSTBACKMODEL")?>
        XML_HTML_4_1_WITH_CONTROLS
        (<xsl:attribute name="style">SECTION_STYLE(<xsl:choose>
                (<xsl:when test="BOOLEAN XPATH EXPRESSION"/>|
                <xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_DISPLAY_NONE</xsl:when>|
                <xsl:when test="BOOLEAN XPATH EXPRESSION">STYLE BACKGROUND COLOR</xsl:when>)+
            </xsl:choose>)?
            (<xsl:if
test="BOOLEAN_XPATH_EXPRESSION">STYLE_DISABLE_CHILD_XML_TO_EDIT</xsl:if>)*
        </xsl:attribute>)?
    </div>
    (SECTION DIGITAL SIGNATURE BLOCK |
SECTION_DIGITAL_SIGNATURE_BLOCK_NOT_SHOWING_SIGNATURES)
```

```
</xsl:template>
```

SIMPLE_SECTION_BODY_WITH_CONDITIONAL_HIDING:

```
<xsl:template match="RELATIVE_GROUP_XPATH" mode="TEMPLATE_MODE_ID1">
    <xsl:if test="BOOLEAN XPATH EXPRESSION">
        <div class="xdSection xdRepeating" title="ANY_STRING" (style="SECTION_STYLE")?
align="ALIGN" xd:xctname="Section" xd:CtrlId="CONTROL ID" (tabIndex="-1")?
(xd:postbackModel="POSTBACKMODEL")?>
            XML_HTML_4_1_WITH_CONTROLS
            (<xsl:attribute name="style">SECTION_STYLE(<xsl:choose>
                    (<xsl:when test="BOOLEAN XPATH EXPRESSION"/>|
                    <xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_DISPLAY_NONE</xsl:when>|
                    <xsl:when
test="BOOLEAN_XPATH_EXPRESSION">STYLE_BACKGROUND_COLOR</xsl:when>)+
                </xsl:choose>)?
                (<xsl:if
test="BOOLEAN XPATH EXPRESSION">STYLE DISABLE CHILD XML TO EDIT</xsl:if>)*
            </xsl:attribute>)?
        </div>
    </xsl:if>
</xsl:template>
```

SIMPLE_SECTION_BODY_WITH_CONDITIONAL_HIDING_AND_DIGITAL_SIGNATURE:

```
<xsl:template match="RELATIVE_GROUP_XPATH" mode="TEMPLATE_MODE_ID1">
    <xsl:if test="BOOLEAN_XPATH_EXPRESSION">
        <div class="xdSection xdRepeating" title="ANY STRING" (style="SECTION STYLE")?
align="ALIGN" xd:xctname="Section" xd:CtrlId="CONTROL_ID" xd:SignedSectionName="ANY_STRING1"
(tabIndex="-1")? (xd:postbackModel="POSTBACKMODEL")?>
            XML_HTML_4_1_WITH_CONTROLS
            (<xsl:attribute name="style">SECTION STYLE(<xsl:choose>
                    (<xsl:when test="BOOLEAN_XPATH_EXPRESSION"/>|
                    <xsl:when test="BOOLEAN XPATH EXPRESSION">STYLE DISPLAY NONE</xsl:when>|
                    <xsl:when
test="BOOLEAN_XPATH_EXPRESSION">STYLE_BACKGROUND_COLOR</xsl:when>)+
                </xsl:choose>)?
                (<xsl:if
test="BOOLEAN_XPATH_EXPRESSION">STYLE_DISABLE_CHILD_XML_TO_EDIT</xsl:if>)*
            </xsl:attribute>)?
        </div>
        (SECTION_DIGITAL_SIGNATURE_BLOCK |
SECTION_DIGITAL_SIGNATURE_BLOCK_NOT_SHOWING_SIGNATURES)
    </xsl:if>
</xsl:template>
```

SECTION_DIGITAL_SIGNATURE_BLOCK_VALID_SIGNATURE_BUTTON:

```
<button title="" style="width: 100%; height: 100%; text-align: left; border: 0px solid;
padding: 2px; background-color: window; cursor: hand;">
    <table style="color: windowtext;" class="defaultInDocUI">
        <tbody>
            <tr>
                <xsl:choose>
                    <xsl:when test="function-available('xdImage:getImageUrl') and
sig:Object/sig:SignatureProperties/sig:SignatureProperty/xdSignatureProperties:NonRepudiation
/xdSignatureProperties:ValidSignedImage">
                        <td style="display: none;"><img
src="res://infopath.exe/signaturevalid.gif"/></td>
                        <td><img
src="{xdImage:getImageUrl(sig:Object/sig:SignatureProperties/sig:SignatureProperty/xdSignatur
eProperties:NonRepudiation/xdSignatureProperties:ValidSignedImage)}"/> </td>
                    </xsl:when>
```

```
                    <xsl:otherwise>
                        <td><img src="res://infopath.exe/signaturevalid.gif" width="12"
height="12"/></td>
                        <td style="color: gray;">
                            <div><b><xsl:value-of select="xdXDocument:GetNamedNodeProperty(.,
'SignedBy', '???')"/></b><span style="margin: 0pt 20pt">ANY STRING</span></div>
                            <div><xsl:value-of select="xdXDocument:GetNamedNodeProperty(.,
'SignedOn', '???')"/></div>
                        </td>
                    </xsl:otherwise>
                </xsl:choose>
            </tr>
        </tbody>
    </table>
</button>
```

SECTION_DIGITAL_SIGNATURE_BLOCK_INVALID_SIGNATURE_BUTTON:

```
<button title="" style="width: 100%; height: 100%; text-align: left; border: 0px solid;
padding: 2px; background-color: window; cursor: hand;">
    <table style="font: message-box; color: windowtext;">
        <tbody>
            <tr>
                <xsl:choose>
                    <xsl:when test="function-available('xdImage:getImageUrl') and
sig:Object/sig:SignatureProperties/sig:SignatureProperty/xdSignatureProperties:NonRepudiation
/xdSignatureProperties:InvalidSignedImage">
                        <td style="display: none;"><img
src="res://infopath.exe/signaturevalid.gif"/></td>
                        <td><img
src="{xdImage:getImageUrl(sig:Object/sig:SignatureProperties/sig:SignatureProperty/xdSignatur
eProperties:NonRepudiation/xdSignatureProperties:InvalidSignedImage)}"/> </td>
                    </xsl:when>
                    <xsl:otherwise>
                        <td><img src="res://infopath.exe/signatureerror.gif" width="12"
height="12"/></td>
                        <td style="color: red;"><b>ANY STRING</b><span style="margin: 0pt
20pt">ANY_STRING</span></td>
                    </xsl:otherwise>
                </xsl:choose>
            </tr>
        </tbody>
    </table>
</button>
```

SECTION_DIGITAL_SIGNATURE_BLOCK:

```
<div xd:disableEditing="yes" xd:SignatureBlock="ANY_STRING1"
xd:SignedSectionDisplaySignatures="true" style="DIGITAL SIGNATURE STYLE">
    <xsl:if test="function-available('xdXDocument:GetNamedNodeProperty')">
        <xsl:if test="xdXDocument:GetNamedNodeProperty(DIGITAL SIGNATURE XPATH,
'CanAddSignature', 'false') = 'true'">
            <button title="" style="width: 100%; height: 100%; text-align: left; border: 0px
solid; padding: 2px; background-color: window; cursor: hand;">
                <table style="color: windowtext;" class="defaultInDocUI">
                    <tbody>
                        <tr>
                            <td><img src="res://infopath.exe/AddSignatureButton.gif"
width="12" height="12"/></td>
                            <td>ANY_STRING</td>
                        </tr>
                    </tbody>
                </table>
            </button>
        </xsl:if>
```

```
            <xsl:for-each select="DIGITAL_SIGNATURE_XPATH">
                <xsl:for-each select="sig:Signature">
                    <xsl:choose>
                        <xsl:when test="xdXDocument:GetNamedNodeProperty(., 'IsValidSignature',
'false') = 'true'">
```

SECTION_DIGITAL_SIGNATURE_BLOCK_VALID_SIGNATURE_BUTTON:

```
                        </xsl:when>
                        <xsl:otherwise>
```

SECTION_DIGITAL_SIGNATURE_BLOCK_INVALID_SIGNATURE_BUTTON:

```
                        </xsl:otherwise>
                    </xsl:choose>
                </xsl:for-each>
            </xsl:for-each>
        </xsl:if>
    </div>
```

SECTION_DIGITAL_SIGNATURE_BLOCK_NOT_SHOWING_SIGNATURES:

```
    <div xd:disableEditing="yes" xd:SignatureBlock="ANY STRING1" style="DIGITAL SIGNATURE STYLE">
        <xsl:if test="function-available('xdXDocument:GetNamedNodeProperty')">
            <xsl:if test="xdXDocument:GetNamedNodeProperty(DIGITAL_SIGNATURE_XPATH,
'CanAddSignature', 'false') = 'true'">
                <button title="" style="width: 100%; height: 100%; text-align: left; border: 0px
solid; padding: 2px; background-color: window; cursor: hand;">
                    <table style="color: windowtext;" class="defaultInDocUI">
                        <tbody>
                            <tr>
                                <td><img src="res://infopath.exe/AddSignatureButton.gif"
width="12" height="12"/></td>
                                <td>ANY_STRING</td>
                            </tr>
                        </tbody>
                    </table>
                </button>
            </xsl:if>
        </xsl:if>
    </div>
```

Control-specific attributes used by the section control are as follows:

- xd:disableEditing (section 2.4.2.12)
- xd:postbackModel (section 2.4.2.29)
- xd:SignatureBlock (section 2.4.2.31)
- xd:SignedSectionDisplaySignatures (section 2.4.2.32)
- xd:SignedSectionName (section 2.4.2.33)
- xd:xctname (section 2.4.2.35)
- xd:xmlToEdit (section 2.4.2.36)

XSL function extensions used by the section control are as follows:

- xdImage:getImageUrl (section 2.4.3.5)
- xdXDocument:GetNamedNodeProperty (section 2.4.3.9.3)

2.4.1.19 Table Control

A table control is used to ensure that elements of the form are positioned according to the requirements of the form designer. A **TABLE** MUST have one of the symbols in the following table.

Symbol	Description
TABLE	Table maps to a standard html layout table as specified by [HTML] section 11.2.1. Each cell can contain arbitrary html or more controls.
TABLE_COLUMN	Maps to the html col element as specified by [HTML] section 11.2.4.
TABLE_ROW	Maps to the html tr element as specified by [HTML] section 11.2.5.
TABLE_CELL	Maps to the html td element as specified by [HTML] section 11.2.6.

TABLE:

```
<table class="(xdFormLayout)? xdLayout" style="STYLE BORDER STYLE?; STYLE BORDER COLLAPSE?
TABLE-LAYOUT: fixed; STYLE_WIDTH?; WORD-WRAP: break-word" (borderColor="buttontext")? WIDTH?
border="1">
    <colgroup>
        TABLE COLUMN+
    </colgroup>
    <tbody (valign="VALIGN")?>
        TABLE_ROW*
    </tbody>
</table>
```

TABLE_COLUMN:

```
<col style="STYLE_WIDTH" />

TABLE_ROW ::=
<tr (class="primaryVeryDark" | class="primarylight")? (style="MIN_HEIGHT")?>
    (<xsl:attribute name="style">
        <xsl:if test="BOOLEAN XPATH EXPRESSION">STYLE DISABLE CHILD XML TO EDIT</xsl:if>
    </xsl:attribute>)?
    (TABLE CELL)+
</tr>

TABLE_CELL ::=
<td (rowSpan="ROWSPAN")? (colSpan="COLSPAN")? (xd:layoutText="ANY STRING")?
(valign="VALIGN")? (style="TABLE_CELL_STYLE")?>
    XML_HTML_4_1_WITH_CONTROLS
</td>
```

TABLE_CELL_STYLE: Semicolon-delimited list of (**STYLE_BORDER**?, **STYLE_MARGIN**?, **STYLE_PADDING**?, **STYLE_VERTICAL_ALIGN**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_TEXT_ALIGN**?)

Xd:layoutText, as specified in section 2.4.2.22, is the control-specific attribute used by the table control.

2.4.1.20 Text Box Control

The text box control allows the user to enter simple text in the form. A **TEXT_BOX_CONTROL** MUST have one of the symbols in the following table.

Symbol	Description
SIMPLE_TEXT_BOX	The text box with no conditional formatting, multiple lines, placeholder text, or data formatting.
TEXT_BOX_WITH_CONDITIONAL_FORMATTING	The text box with conditional formatting.
SIMPLE_TEXT_BOX_MULTI_LINE	The text box that allows multiple lines input.
TEXT_BOX_MULTI_LINE_WITH_CONDITIONAL_FORMATTING	The text box with conditional formatting and multiple lines input.
SIMPLE_TEXT_BOX_WITH_DATA_FORMATTING	The text box with data formatting.
TEXT_BOX_WITH_CONDITIONAL_FORMATTING_AND_DATA_FORMATTING	The text box with data formatting and conditional formatting.
SIMPLE_TEXT_BOX_WITH_PLACEHOLDER_TEXT	The text box with placeholder text. The placeholder text is ignored by the form server.
SIMPLE_TEXT_BOX_WITH_PLACEHOLER_TEXT_AND_DATA_FORMATTING	The text box with placeholder text and data formatting. The placeholder text is ignored by the form server.
TEXT_BOX_MULTI_LINE_PLACEHOLDER_TEXT	The text box that allows multiple lines input and has placeholder text. The placeholder text is ignored by the form server.
TEXT_BOX_MULTI_LINE_CONDITIONAL_FORMATTING_AND_PLACEHOLDER_TEXT	The text box that allows multiple lines input and has placeholder text with conditional formatting. The placeholder text is ignored by the form server.
TEXT_BOX_WITH_CONDITIONAL_FORMATTING_AND_PLACEHOLDER_TEXT	The text box that has placeholder text with conditional formatting. The placeholder text is ignored by the form server.
TEXT_BOX_WITH_CONDITIONAL_FORMATTING_AND_PLACEHOLDER_TEXT_AND_DATA_FORMATTING	The text box that has placeholder text with conditional formatting and data formatting. The placeholder text is ignored by the form server.

TEXT_BOX_EDITING: contentEditable="true" or xd:disableEditing="yes" or contentEditable="true" xd:disableEditing="yes".

TEXT_BOX_AUTOADVANCE: xd:autoAdvance="yes".

WHITESPACE_NO_WRAP: WHITE-SPACE: nowrap.

TEXT_BOX_OUTPUT_ESC: disable-output-escaping="yes".

TEXT_BOX_STYLE: Semicolon-delimited list of (**STYLE_SIZE**?, **STYLE_MARGIN**?, **STYLE_PADDING**?, **STYLE_TEXT_DECORATION**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_BORDER**?, **STYLE_FONT**?, **STYLE_COLOR**?, **WHITESPACE_NO_WRAP**?, **STYLE_WIDTH**?, **STYLE_WRAP**?, **STYLE_TEXT_ALIGN**?, (**OVERFLOW-Y**: auto; **OVERFLOW-X**: auto;)?, **STYLE_VERTICAL_ALIGN**?, **STYLE_DIRECTION**?)

TEXT_BOX_STYLE_CONDITIONAL_FORMATTING: Semicolon-delimited list of (**STYLE_TEXT_DECORATION**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_FONT**?, **STYLE_COLOR**?, **STYLE_TEXT_ALIGN**?)

TEXT_BOX_BASE_CLASS_NAME: xdTextBox or **xdTextBoxRTL.**

SIMPLE_TEXT_BOX:

```
<span class="TEXT_BOX_BASE_CLASS_NAME" hideFocus="1" title="ANY_STRING"
xd:binding="LEAF_XPATH1"
    tabIndex="TAB_INDEX"  xd:xctname="PlainText" xd:CtrlId="CONTROL_ID"
    (TEXT_BOX_EDITING)? style="TEXT_BOX_STYLE" (TEXT_BOX_AUTOADVANCE)?
    (xd:postbackModel="POSTBACKMODEL")? (accessKey="SINGLE_CHARACTER")? (INPUT_SCOPE)?>
    CHECK FOR GETDOM BEGIN1
    <xsl:value-of select="LEAF_XPATH1" TEXT_BOX_OUTPUT_ESC? />
    CHECK_FOR_GETDOM_END1
</span>
```

TEXT_BOX_WITH_CONDITIONAL_FORMATTING:

```
<span class="TEXT_BOX_BASE_CLASS_NAME" hideFocus="1" title="ANY_STRING"
xd:binding="LEAF XPATH1" (style="TEXT BOX STYLE")?
    tabIndex="TAB_INDEX" xd:xctname="PlainText" xd:CtrlId="CONTROL_ID"
    (TEXT_BOX_EDITING)? (TEXT_BOX_AUTOADVANCE)?
    (xd:postbackModel="POSTBACKMODEL")? (accessKey="SINGLE_CHARACTER")? (INPUT_SCOPE)?>
        CHECK FOR GETDOM BEGIN1
        TEXT_BOX_CONDITIONAL_FORMATTING
        <xsl:value-of select="LEAF XPATH1" TEXT BOX OUTPUT ESC? />
        CHECK_FOR_GETDOM_END1
</span>
```

TEXT_BOX_CONDITIONAL_FORMATTING: TEXT_BOX_CONDITIONAL_FORMATTING_ATT (**TEXT_BOX_CONDITIONAL_FORMATTING_CHOOSE**)? or **TEXT_BOX_CONDITIONAL_FORMATTING_CHOOSE**

TEXT_BOX_CONDITIONAL_FORMATTING_ATT:

```
<xsl:attribute name="style">TEXT BOX STYLE<xsl:choose>
        (<xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_DISPLAY_NONE</xsl:when>|
        <xsl:when test="BOOLEAN_XPATH_EXPRESSION"/>|
        <xsl:when test="BOOLEAN XPATH EXPRESSION">TEXT BOX STYLE CONDITIONAL FORMATTING
        </xsl:when>)+
    </xsl:choose>
</xsl:attribute>
```

TEXT_BOX_CONDITIONAL_FORMATTING_CHOOSE:

```
<xsl:choose>
    (<xsl:when test="BOOLEAN_XPATH_EXPRESSION"/>|
    <xsl:when test="BOOLEAN XPATH EXPRESSION">
        <xsl:attribute name="contentEditable">false</xsl:attribute>
    </xsl:when>)+
</xsl:choose>
```

SIMPLE_TEXT_BOX_MULTI_LINE:

```
<span class="TEXT_BOX_BASE_CLASS_NAME" hideFocus="1" title="ANY_STRING"
    xd:binding="LEAF XPATH1" tabIndex="TAB INDEX" (TEXT BOX AUTOADVANCE)?
    xd:datafmt="DATA_FMT_CAT_STRING" xd:xctname="PlainText" xd:CtrlId="CONTROL_ID"
    (TEXT_BOX_EDITING)? TEXT_BOX_STYLE
    (xd:postbackModel="POSTBACKMODEL")? (accessKey="SINGLE_CHARACTER")? (INPUT_SCOPE)?>
    CHECK FOR GETDOM BEGIN1
    <xsl:choose>
        <xsl:when test="function-available('xdFormatting:formatString')">
            <xsl:value-of select="xdFormatting:formatString(LEAF_XPATH1,
DATA_FMT_CAT_STRING)" TEXT_BOX_OUTPUT_ESC />
        </xsl:when>
        <xsl:otherwise>
```

```
            <xsl:value-of select="LEAF_XPATH1" TEXT_BOX_OUTPUT_ESC />
        </xsl:otherwise>
    </xsl:choose>
    CHECK_FOR_GETDOM_END1
</span>
```

TEXT_BOX_MULTI_LINE_WITH_CONDITIONAL_FORMATTING:

```
<span class="TEXT_BOX_BASE_CLASS_NAME" hideFocus="1" title="ANY_STRING"
    xd:binding="LEAF_XPATH1"  tabIndex="TAB_INDEX" (style="TEXT_BOX_STYLE")?
    xd:datafmt="DATA_FMT_CAT_STRING" xd:xctname="PlainText"
    xd:CtrlId="CONTROL ID" (TEXT BOX EDITING)? (TEXT BOX AUTOADVANCE)?
    (xd:postbackModel="POSTBACKMODEL")? (accessKey="SINGLE_CHARACTER")? (INPUT_SCOPE)?>
    CHECK FOR GETDOM BEGIN1
    TEXT_BOX_CONDITIONAL_FORMATTING
    <xsl:choose>
        <xsl:when test="function-available('xdFormatting:formatString')">
            <xsl:value-of select="xdFormatting:formatString(LEAF XPATH1,
DATA_FMT_CAT_STRING)" TEXT_BOX_OUTPUT_ESC />
        </xsl:when>
        <xsl:otherwise>
            <xsl:value-of select="LEAF_XPATH1" TEXT_BOX_OUTPUT_ESC />
        </xsl:otherwise>
    </xsl:choose>
    CHECK FOR GETDOM END1
</span>
```

DATA_FMT_TEXT_BOX_VAL: xd:datafmt="DATA_FMT_CTRL_TEXTBOX"

DATA_FMT_XSL_BASE:

```
<xsl:when test="function-available('xdFormatting:formatString')">
    <xsl:value-of select="xdFormatting:formatString(LEAF_XPATH1, DATA_FMT_CTRL_TEXTBOX)"/>
</xsl:when>
```

DATA_FMT_XSL_NUM:

```
<xsl:attribute name="xd:num">
    <xsl:value-of select="LEAF_XPATH1"/>
</xsl:attribute>
```

DATA_FMT_XSL:

```
    DATA_FMT_XSL_NUM
    <xsl:choose>
        DATA_FMT_XSL_BASE
        <xsl:otherwise>
            <xsl:value-of select="LEAF_XPATH1"/>
        </xsl:otherwise>
    </xsl:choose>
```

SIMPLE_TEXT_BOX_WITH_DATA_FORMATTING:

```
<span class="TEXT_BOX_BASE_CLASS_NAME xdBehavior_Formatting" hideFocus="1" title="ANY_STRING"
xd:binding="LEAF XPATH1" (TEXT BOX AUTOADVANCE)?
    tabIndex="TAB_INDEX" DATA_FMT_TEXT_BOX_VAL xd:xctname="PlainText" xd:CtrlId="CONTROL_ID"
    (TEXT BOX EDITING)? style="TEXT BOX STYLE" xd:boundProp="xd:num"
    (xd:postbackModel="POSTBACKMODEL")? (accessKey="SINGLE_CHARACTER")? (INPUT_SCOPE)?>
```

```
    CHECK_FOR_GETDOM_BEGIN1
    DATA_FMT_XSL
    CHECK_FOR_GETDOM_END1
</span>
```

TEXT_BOX_WITH_CONDITIONAL_FORMATTING_AND_DATA_FORMATTING:

```
    <span class="TEXT BOX BASE CLASS NAME xdBehavior Formatting" hideFocus="1"
    title="ANY_STRING" xd:binding="LEAF_XPATH1" xd:boundProp="xd:num"
    tabIndex="TAB_INDEX" DATA_FMT_TEXT_BOX_VAL xd:xctname="PlainText" xd:CtrlId="CONTROL_ID"
    (TEXT_BOX_EDITING)? (TEXT_BOX_AUTOADVANCE)? (style="TEXT_BOX_STYLE")?
    (xd:postbackModel="POSTBACKMODEL")? (accessKey="SINGLE CHARACTER")? (INPUT SCOPE)?>
    CHECK_FOR_GETDOM_BEGIN1
    TEXT BOX CONDITIONAL FORMATTING
    DATA_FMT_XSL
    CHECK_FOR_GETDOM_END1
</span>
```

PLACEHOLDER_TEXT_XSL_BASE:

```
<xsl:when test="not(string(LEAF XPATH1))">
    <xsl:attribute name="xd:ghosted">true</xsl:attribute>ANY_STRING
</xsl:when>
```

PLACEHOLDER_TEXT_XSL:

```
<xsl:choose>
    PLACEHOLDER_TEXT_XSL_BASE
    <xsl:otherwise>
        <xsl:value-of select="LEAF_XPATH1" TEXT_BOX_OUTPUT_ESC? />
    </xsl:otherwise>
</xsl:choose>
```

SIMPLE_TEXT_BOX_WITH_PLACEHOLDER_TEXT:

```
<span class="TEXT BOX BASE CLASS NAME xdBehavior GhostedText" hideFocus="1"
title="ANY_STRING" xd:binding="LEAF_XPATH1" (TEXT_BOX_AUTOADVANCE)?
    tabIndex="TAB INDEX"  xd:xctname="PlainText" xd:CtrlId="CONTROL ID"
    (TEXT_BOX_EDITING)? style="TEXT_BOX_STYLE"
    (xd:postbackModel="POSTBACKMODEL")? (accessKey="SINGLE_CHARACTER")? (INPUT_SCOPE)?>
    CHECK FOR GETDOM BEGIN1
    PLACEHOLDER_TEXT_XSL
    CHECK FOR GETDOM END1
</span>
```

SIMPLE_TEXT_BOX_WITH_PLACEHOLER_TEXT_AND_DATA_FORMATTING:

```
<span class="TEXT_BOX_BASE_CLASS_NAME xdBehavior_GTFormatting" hideFocus="1"
    title="ANY_STRING" xd:binding="LEAF_XPATH1" xd:boundProp="xd:num"
    tabIndex="TAB INDEX" DATA FMT TEXT BOX VAL xd:xctname="PlainText" xd:CtrlId="CONTROL ID"
    (TEXT_BOX_EDITING)? style="TEXT_BOX_STYLE" (TEXT_BOX_AUTOADVANCE)?
    (xd:postbackModel="POSTBACKMODEL")? (accessKey="SINGLE CHARACTER")? (INPUT SCOPE)?>
    CHECK_FOR_GETDOM_BEGIN1
    DATA_FMT_XSL_NUM
    <xsl:choose>
        PLACEHOLDER TEXT XSL BASE
        DATA_FMT_XSL_BASE
        <xsl:otherwise>
```

```
            <xsl:value-of select="LEAF_XPATH1"/>
        </xsl:otherwise>
    </xsl:choose>
    CHECK_FOR_GETDOM_END1
</span>
```

TEXT_BOX_MULTI_LINE_PLACEHOLDER_TEXT:

```
<span class="TEXT_BOX_BASE_CLASS_NAME xdBehavior_GhostedText" hideFocus="1"
title="ANY_STRING"
    xd:binding="LEAF_XPATH1" tabIndex="TAB_INDEX"
    xd:datafmt="DATA FMT CAT STRING" xd:xctname="PlainText" xd:CtrlId="CONTROL ID"
    (TEXT_BOX_EDITING)? TEXT_BOX_STYLE (TEXT_BOX_AUTOADVANCE)?
    (xd:postbackModel="POSTBACKMODEL")? (accessKey="SINGLE CHARACTER")? (INPUT SCOPE)?>
    CHECK_FOR_GETDOM_BEGIN1
    <xsl:choose>
        PLACEHOLDER_TEXT_XSL_BASE
        <xsl:when test="function-available('xdFormatting:formatString')">
            <xsl:value-of select="xdFormatting:formatString(LEAF_XPATH1,
DATA FMT CAT STRING)" TEXT BOX OUTPUT ESC />
        </xsl:when>
        <xsl:otherwise>
            <xsl:value-of select="LEAF_XPATH1" TEXT_BOX_OUTPUT_ESC/>
        </xsl:otherwise>
    </xsl:choose>
    CHECK_FOR_GETDOM_END1
</span>
```

TEXT_BOX_MULTI_LINE_CONDITIONAL_FORMATTING_AND_PLACEHOLDER_TEXT:

```
<span class="TEXT_BOX_BASE_CLASS_NAME xdBehavior_GhostedText" hideFocus="1"
title="ANY STRING"
    xd:binding="LEAF_XPATH1" tabIndex="TAB_INDEX"
       xd:datafmt="DATA FMT CAT STRING" xd:xctname="PlainText" xd:CtrlId="CONTROL ID"
    (TEXT_BOX_EDITING)? TEXT_BOX_STYLE (TEXT_BOX_AUTOADVANCE)?
    (style="TEXT_BOX_STYLE")?
    (xd:postbackModel="POSTBACKMODEL")? (accessKey="SINGLE_CHARACTER")? (INPUT_SCOPE)?>
    CHECK FOR GETDOM BEGIN1
    TEXT_BOX_CONDITIONAL_FORMATTING
    <xsl:choose>
        PLACEHOLDER_TEXT_XSL_BASE
        <xsl:when test="function-available('xdFormatting:formatString')">
            <xsl:value-of select="xdFormatting:formatString(LEAF_XPATH1,
DATA FMT CAT STRING)" TEXT BOX OUTPUT ESC />
        </xsl:when>
        <xsl:otherwise>
            <xsl:value-of select="LEAF_XPATH1" TEXT_BOX_OUTPUT_ESC />
        </xsl:otherwise>
    </xsl:choose>
    CHECK FOR GETDOM END1
</span>
```

TEXT_BOX_WITH_CONDITIONAL_FORMATTING_AND_PLACEHOLDER_TEXT:

```
<span class="TEXT_BOX_BASE_CLASS_NAME xdBehavior_GhostedText" hideFocus="1"
title="ANY_STRING" xd:binding="LEAF_XPATH1" (TEXT_BOX_AUTOADVANCE)? (style="TEXT_BOX_STYLE")?
    tabIndex="TAB_INDEX"  xd:xctname="PlainText" xd:CtrlId="CONTROL_ID"
    (TEXT BOX EDITING)? (xd:postbackModel="POSTBACKMODEL")?
    (accessKey="SINGLE_CHARACTER")? (INPUT_SCOPE)?>
    CHECK FOR GETDOM BEGIN1
    TEXT_BOX_CONDITIONAL_FORMATTING
    PLACEHOLDER_TEXT_XSL
    CHECK_FOR_GETDOM_END1
```

```
</span>
```

TEXT_BOX_WITH_CONDITIONAL_FORMATTING_AND_PLACEHOLDER_TEXT_AND_DATA_FORMATTING:

```
<span class="TEXT_BOX_BASE_CLASS_NAME xdBehavior_GTFormatting" hideFocus="1"
    title="ANY STRING" xd:binding="LEAF XPATH1" DATA FMT TEXT BOX VAL xd:boundProp="xd:num"
tabIndex="TAB_INDEX"  xd:xctname="PlainText" xd:CtrlId="CONTROL_ID"
    (style="TEXT BOX STYLE")?
    (TEXT_BOX_EDITING)? (xd:postbackModel="POSTBACKMODEL")? (TEXT_BOX_AUTOADVANCE)?
    (accessKey="SINGLE_CHARACTER")? (INPUT_SCOPE)?>
    CHECK_FOR_GETDOM_BEGIN1
    TEXT BOX CONDITIONAL FORMATTING
    DATA_FMT_XSL_NUM
    <xsl:choose>
        PLACEHOLDER_TEXT_XSL_BASE
        DATA_FMT_XSL_BASE
        <xsl:otherwise>
            <xsl:value-of select="LEAF XPATH1"/>
        </xsl:otherwise>
    </xsl:choose>
    CHECK_FOR_GETDOM_END1
</span>
```

Control-specific attributes used by the textbox control are as follows:

- xd:allownonmatching (section 2.4.2.2)
- xd:autoAdvance (section 2.4.2.3)
- xd:binding (section 2.4.2.6)
- xd:boundProp (section 2.4.2.9)
- xd:CtrlId (section 2.4.2.10)
- xd:datafmt (section 2.4.2.11)
- xd:disableEditing (section 2.4.2.12)
- xd:ghosted (section 2.4.2.15)
- xd:inputScopeId (section 2.4.2.21)
- xd:num (section 2.4.2.26)
- xd:postbackModel (section 2.4.2.29)
- xd:xctname (section 2.4.2.35)

2.4.1.21 Ignored Controls

XSL files SHOULD contain valid controls but MAY contain controls that are not recognized by the form server, and are therefore ignored. The XSL fragment that maps to an ignored control is passed directly by the form server to the **user agent**. Ignored controls have no mechanism for persisting information to the form server, nor are they able to manipulate form data.

The following table is a list of ignored controls and how to identify them.

Controls	Identifying Characteristic
Choice Group	xctName attribute = "ChoiceGroup"

2.4.1.22 Invalid Controls

XHTML elements MUST NOT have an **xd:xctName** attribute (section 2.4.2.35) matching any of the **strings**, case insensitive, in the following list.

- inkpicture
- scrollableregion
- combobox
- multiselectlistbox
- layoutregion
- choiceterm
- choicegrouprepeating
- choicetermrepeating
- bulletedlist
- numberedlist
- plainlist

Every XHTML element that contains an **xctName** attribute (section 2.4.2.35) MUST be specified in the controls section of this document.

An **Expression Box** (section 2.4.1.10) MUST NOT contain the writing-mode style in its style block.

A **Repeating Section** (section 2.4.1.15) MUST NOT be encased in a **SPAN** HTML element instead of a **DIV** or **TABLE** element.

A control MUST NOT have an **XmlToEdit** element (section 2.2.124) with a recursive item XPath expression.

A **Repeating Section** (section 2.4.1.15) MUST NOT contain the attribute **linkedToMaster**.

A form definition (.xsf) file that contains an **editWith** element (section 2.2.108) with the **component** attribute set to the value "xImage" or "xReplace" MUST NOT be present.

2.4.1.23 Invalid Constructs

An XSLT file that contains an **xsl:template** element with the **mode** attribute set to the value "xd:preserve" MUST NOT be present.

2.4.2 Control-Specific Attributes

This section specifies the use of attributes in the XSLT file, as specified in section 2.4.1.

Examples of the use of the attributes specified in this section can be found in section 3.4.2.

These attributes MUST be associated with the "xd" namespace prefix, as specified in [XMLSCHEMA1], and the "http://schemas.microsoft.com/office/infopath/2003" namespace in the XSLT file.

Attribute	Description
action (section 2.4.2.1)	The action executed when a control is clicked.
allownonmatching (section 2.4.2.2)	Client-only.
autoAdvance (section 2.4.2.3)	A **Boolean** value that specifies whether to move focus to the next control in the form when the text limit is reached on a text box control.
auxDom (section 2.4.2.4)	The **name** (section 2.2.36) of the secondary data connection associated with the "refresh" **action** (section 2.4.2.1) of a button control.
backgroundPicture (section 2.4.2.5)	Client-only.
binding (section 2.4.2.6)	The XML field from which the control reads and writes data.
bindingProperty (section 2.4.2.7)	The name of the custom control property used by the custom control to read and write data from and to the XML field associated with the control.
bindingType (section 2.4.2.8)	The format of the data that the custom control reads and writes.
boundProp (section 2.4.2.9)	The name of the XSLT attribute that contains the XML field from which the control reads and writes data.
CtrlId (section 2.4.2.10)	The unique identifier of a control in the form (1).
datafmt (section 2.4.2.11)	The data formatting the control uses to display the data in the associated XML field in the form (1).
disableEditing (section 2.4.2.12)	A Boolean value that specifies whether a control is read-only.
enabledProperty (section 2.4.2.13)	The name of the custom control property called to enable or disable the control.
enabledValue (section 2.4.2.14)	A Boolean value used to enable or disable the custom control.
ghosted (section 2.4.2.15)	Client-only.
ictID (section 2.4.2.16)	Client-only.
ictVersion (section 2.4.2.17)	Client-only.
inline (section 2.4.2.18)	Client-only.
innerCtrl (section 2.4.2.19)	A type of component inside the date picker control.
inputscope (section 2.4.2.20)	Client-only.
inputScopeId (section 2.4.2.21)	Client-only.
layoutText (section 2.4.2.22)	Client-only.
linkedToMaster (section 2.4.2.23)	Client-only.
masterID (section 2.4.2.24)	Client-only.
masterName (section 2.4.2.25)	Client-only.
num (section 2.4.2.26)	Indicates that the **xd:num** XSLT attribute contains the XML field from which the control reads and writes data.
offValue (section 2.4.2.27)	The XML field value for a check box control when that check box control is **unchecked**.
onValue (section 2.4.2.28)	The XML field value for a control when that control is selected.
postbackModel (section 2.4.2.29)	Specifies whether to trigger a postback when the XML field value displayed in the control is changed.

Attribute	Description
ref (section 2.4.2.30)	Client-only.
SignatureBlock (section 2.4.2.31)	The name for the signed data block (section 2.2.126) for a control that allows XML digital signatures.
SignedSectionDisplaySignatures (section 2.4.2.32)	A Boolean value that specifies whether to display XML digital signatures (1) inline in the form (1) for a control that allows XML digital signatures (1).
SignedSectionName (section 2.4.2.33)	The name for the signed data block (section 2.2.126) for a control that allows XML digital signatures (1).
value (section 2.4.2.34)	Indicates that the **xd:value** XSLT attribute contains the XML field from which the control reads and writes data.
xctname (section 2.4.2.35)	The type of the control.
xmlToEdit (section 2.4.2.36)	The additional editing properties of the control.

2.4.2.1 action

The following controls MAY contain the **action** attribute in their respective XSLT representations:

- Button (section 2.4.1.5)
- Repeating section (section 2.4.1.15)
- Repeating table (section 2.4.1.16)

All other controls MUST NOT contain this attribute in their XSLT representations.

For button controls, the value of the **action** attribute MUST be one of the following:

- **delete:** This action is associated with a client-only feature and MUST NOT be present.
- **new:** Creates a new record associated with an ADO (section 2.2.38) data connection. This action MUST be set only if the main data connection is an ADO (section 2.2.38) data connection (1).
- **query:** Calls the **query** of the main data source's data connection. This value MUST be used only within forms in which the main data source is a data connection.
- **refresh:** Calls the query of the secondary data connection specified by the **auxDom** attribute on the same control. This action MUST be set only if the form has at least one secondary data connection that can query.
- **submit:** Calls the action to submit the form.
- **updateForm:** Calls a postback to the form server to update the form.

For repeating section and repeating table controls, the value of this attribute MUST be "xCollection::insert". This value inserts the XML editing component associated with the repeating section and repeating table controls.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.2 allownonmatching

The **allownonmatching** attribute is associated with a client-only feature and MUST be ignored by the form server.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.3 autoAdvance

Text box controls (section 2.4.1.20) MAY contain the **autoAdvance** attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **autoAdvance** attribute MUST be "yes". This value moves the focus to the next control when the text limit is reached.

If this attribute is unspecified, the behavior MUST be to not move focus to the next control when the text limit is reached.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.4 auxDom

Button controls (section 2.4.1.5) MAY contain the **auxDom** attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **auxDom** attribute MUST range from 1 through 255 characters and MUST be equal to the **name** of a secondary data connection (section 2.2.36) that exists in the form.

If this attribute is unspecified, all of the secondary data sources that can query MUST be refreshed as part of the "refresh" **action** (section 2.4.2.1).

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.5 backgroundPicture

The **backgroundPicture** attribute is associated with a client-only feature and MUST be ignored by the form server.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.6 binding

The following controls MUST contain the **binding** attribute in their XSLT representation:

- Check box (section 2.4.1.6)
- Contact selector (section 2.4.1.7)
- Date picker (section 2.3.1.4)
- Drop-down list box (section 2.4.1.9)
- File attachment (section 2.4.1.11)
- List box (section 2.4.1.13)
- Option button (section 2.4.1.14)
- Rich text box (section 2.4.1.17)
- Text box (section 2.4.1.20)

The expression box control (section 2.4.1.10) MAY contain the **binding** attribute in its XSLT representation.

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **binding** attribute for the following controls MUST be set to a **LEAF_XPATH**, as specified in section 2.4.1.1:

- Check box
- Date picker
- Drop-down list
- Expression box
- File attachment
- List box
- Option button
- Rich text box
- Text box

For the **contact selector** control (section 2.4.1.7), the value of the **binding** attribute MUST be set to a **GROUP_XPATH,** as specified in section 2.4.1.1.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.7 bindingProperty

Custom controls MAY contain the **bindingProperty** attribute in their XSLT representation.

Contact selector controls (section 2.4.1.7) MUST contain this attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **bindi**ngProperty attribute MUST be the name of a property exposed by the custom control.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.8 bindingType

Custom controls MAY contain the **bindingType** attribute in their XSLT representation.

Contact selector controls (section 2.4.1.7) MUST contain this attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their XSLT representations.

Custom controls use the **bindingType** attribute to determine how to read and write data in and out of the control.

The value of this attribute MUST be one of the following:

- **text:** A text string data format.
- **xmlnode:** An XML node data format.
- **xmltext:** An XML string data format.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.9 boundProp

The following controls MUST contain the **boundProp** attribute in their XSLT representation:

- Check box (section 2.4.1.6)
- Custom controls
- Drop-down list (section 2.4.1.9)
- File attachment (section 2.4.1.11)
- List box (section 2.4.1.13)
- Option button (section 2.4.1.14)

The following controls MAY contain this attribute in their XSLT representation:

- Date picker (section 2.4.1.8)
- Text box (section 2.4.1.20)

All other controls MUST NOT contain the **boundProp** attribute in their XSLT representations.

The value of the **boundProp** attribute MUST be one of the values specified in the following table.

Value	Description	Controls
"src"	This value is associated with a client-only feature and MUST be ignored by the form server.	
"value"	Indicates that the **value** XSLT attribute contains the XML field from which the control reads and writes data.	Drop-down list List box
"xdInkData"	This value is associated with a client-only feature and MUST be ignored by the form server.	
"xd:inline"	Indicates that the XML field, from which the control reads and writes data, is inline in the control's XSLT.	Custom controls File attachment
"xd:num"	Indicates that the **xd:num** XSLT attribute contains the XML field from which the control reads and writes data.	Date picker Text box
"xd:value"	Indicates that the **xd:value** XSLT attribute contains the XML field from which the control reads and writes data.	Check box Option button

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.10 CtrlId

The following controls MUST contain the **CtrlId** attribute in their XSLT representation:

- Button (section 2.4.1.5)
- Check box (section 2.4.1.6)
- Contact selector (section 2.4.1.7)
- Custom controls
- Date picker (section 2.3.1.4)
- Drop-down list (section 2.4.1.9)
- Expression box (section 2.4.1.10)
- File attachment (section 2.4.1.11)
- List box (section 2.4.1.13)
- Option button (section 2.4.1.14)
- Optional section (section 2.4.1.18)
- Repeating section (section 2.4.1.15)
- Repeating table (section 2.4.1.16)

- Rich text box (section 2.4.1.17)
- Section (section 2.4.1.18)
- Text box (section 2.4.1.20)

Hyperlink controls (section 2.4.1.12) MAY contain this attribute in their XSLT representation.

The value of the **CtrlId** attribute MUST range from 1 through 255 characters, MUST begin with an alphabetic character, and MUST contain only alphanumeric and underscore characters.

The value of this attribute SHOULD be unique for each button control in the form.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.11 datafmt

The following controls MAY contain the **datafmt** attribute in their XSLT representation:

- Date picker (section 2.4.1.8)
- Expression box (section 2.4.1.10)
- Text box (section 2.4.1.20)

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **datafmt** attribute MUST follow this structure:

```
"&quot;<FormatCategory>&quot;,&quot;<FormatSpecification>&quot;"
```

FormatCategory: The type of formatting applied when the control displays the XML field's data. The **FormatCategory** string MUST be one of the following:

- **currency:** Specifies that the data from the XML field be formatted as type "currency" when displayed in the corresponding control.
- **date:** Specifies that the data from the XML field be formatted as type "date" when displayed in the corresponding control.
- **datetime:** Specifies that the data from the XML field be formatted as type "datetime" when displayed in the corresponding control.
- **number:** Specifies that the data from the XML field be formatted as type "number" when displayed in the corresponding control.
- **percentage:** Specifies that the data from the XML field be formatted as type "percentage" when displayed in the corresponding control.
- **string:** Specifies that the data from the XML field be formatted as type "string" when displayed in the corresponding control.
- **time:** Specifies that the data from the XML field be formatted as type "time" when displayed in the corresponding control.

FormatSpecification: The data formatting properties applied when the control displays an XML field's data. The following table specifies which format specifications MUST be specified for each format category.

FormatCategory	Supported FormatSpecification
currency	locale, numDigits, grouping, decimalSep, thousandSep, negativeOrder, positiveOrder, currencyLocale
date	locale, dateFormat, useAltCalendar, englishStringsAlways
datetime	locale, dateFormat, timeFormat, useAltCalendar, englishStringsAlways, noSeconds
number	locale, numDigits, grouping, decimalSep, thousandSep, negativeOrder
percentage	locale, numDigits, grouping, decimalSep, thousandSep, negativeOrder
string	plainMultiline
time	locale, timeFormat, noSeconds

The **FormatSpecification** MUST be a semicolon-delimited list of one or more specification items. Each item MUST obey the proper structure according to the following table.

Format Specification Item Name	Specification Item Structure
plainMultiline	itemName
All other specification items.	itemName:itemValue

The item names MUST be set to the following and adhere to the specified value requirements:

- **currencyLocale:** The LCID, as specified in [MS-LCID]. This is used to map the currency symbol string determined by the currency locale.
- **dateFormat:** The date format pattern to use when displaying date and time values. The value MUST be one of the following:
  - **Short Date**: A short date format, as specified in [MC-NLSIP].
  - **Long Date**: A long date format, as specified in [MC-NLSIP].
  - **Year Month**: A year month format, as specified in [MC-NLSIP].
  - **none**: No date format pattern applied.
  - A date format pattern specified in [ISO-8601].
- **decimalSep:** The string to display as the decimal separator in numeric values.
  - The value MUST be one of the following:
    - "." (Period)
    - "," (Comma)
    - " " (Space)
    - " " (Non-breaking space)
    - "'" (Single Quote)

        - "٬" (Arabic Comma)
        - Empty (No separator)
    - If this item is unspecified, the decimal separator is determined based on the form server settings.
- **englishStringsAlways:** A Boolean value that specifies the ability to always use English as the LCID, as specified in [MS-LCID], when displaying date and date time values.
    - The value MUST be one of the following:
        - "0": Use locale specified by the LCID, as specified in [MS-LCID].
        - "1": Use English as the LCID, as specified in [MS-LCID].
    - If this item is unspecified, the behavior MUST be the same as a value of zero ("0").
- **grouping:** The digit grouping pattern for digits to the left of the decimal.
    - The value MUST be one of the following:
        - Range from zero ("0"), which means no grouping, through "9"
        - "32"
    - If this item is unspecified, the grouping is determined based on the form server settings.
- **locale:** The LCID, as specified in [MS-LCID]. If this item is unspecified, the locale is determined based on the form server settings.
- **negativeOrder:** The format pattern for negative numeric values. If this value is "-1", the negative order is determined by the default pattern associated with the form server's LCID, as specified in [MS-LCID]. If this item is unspecified, the negative order is determined based on the form server settings.
- **noSeconds:** A Boolean value that specifies whether to display seconds in time formatting.
    - The value MUST be one of the following:
        - "0": Display seconds for time formatting.
        - "1": Do not display seconds for time formatting.
    - If this item is unspecified, the behavior MUST be the same as a value of zero ("0").
- **numDigits:** The number of fractional digits to display after the decimal separator.
    - The value MUST be one of the following:
        - Range from zero ("0") through "9".
        - "Auto": The general numeric format string as implemented by the form server.
    - If this item is unspecified, the number of digits is determined based on the form server settings.
- **plainMultiline:** The ability for a text box control to display data across multiple lines. The value MUST be an empty string ("1").
- **positiveOrder:** The format pattern for positive currency values.

- If this is "-1", the positive order is determined by the default pattern associated with the form server's LCID, as specified in [MS-LCID].
- If this item is unspecified, the positive order is determined based on the form server settings.

- **thousandSep:** The string that separates groups of digits to the left of the decimal in numeric values.
  - The value MUST be one of the following:
    - "." (Period)
    - "," (Comma)
    - " " (Space)
    - " " (Non-breaking space)
    - "'" (Single Quote)
    - "٬" (Arabic Comma)
    - Empty (No separator)
  - If this item is unspecified, the thousand separator is determined based on the form server settings.
- **timeFormat:** The time format pattern to use when displaying time values.
  - The value MUST be one of the following:
    - **Short Time**: A short time format, as specified in [MC-NLSIP].
    - **Long Time**: A long time format, as specified in [MC-NLSIP].
    - **none**: No time format pattern applied.
    - A time format pattern specified in [ISO-8601].
  - If this item is unspecified, the time format is determined based on the form server settings.
- **useAltCalendar:** A Boolean value that specifies the ability to display an alternate calendar for calendar formatting, as specified in [MS-WSSFO] section 2.2.3.3.
  - The value MUST be one of the following:
    - "0": Use calendar type associated with the LCID, as specified in [MS-LCID].
    - "1": Use alternate calendar type for the LCID, as specified in [MS-LCID].
  - If this item is unspecified, the behavior MUST be the same as a value of zero ("0").

If the format category is **datetime**, at least one of the two specification items **timeFormat** and **dateFormat** MUST be "none".

The combinations of locale and date and time format specifications contained in the following table MUST NOT be present.

Locale	Date and time format
1028	"M'\u6708'd'\u65E5'"

Locale	Date and time format
1036	"HH' h 'mm"
1037	"dd \u05D1MMMM yyyy"
1037	"dddd dd \u05D1MMMM yyyy"
1037	"ddd dd \u05D1MMMM yyyy"
1038	"MMMM d."
1041	"M'\u6708'd'\u65E5'"
1042	"M'\uC6D4' d'\uC77C'"
1045	"d MMMM"
1052	"MMMM dd"
1052	"h:mm.tt"
1052	"h:mm:ss.tt"
1053	"'den 'd MMMM"
1054	"MMMM dd"
1062	"d. MMMM"
1063	"MMMM d 'd.'"
1066	"dd MMMM yyyy"
1066	"dd MMMM"
1066	"MMMM yyyy"
1069	"MMMM dd"
1078	"dd MMMM"
1079	"yyyy '\u10EC\u10DA\u10D8\u10E1' dd MM, dddd"
1104	"d MMMM"
1125	"MMMM dd"
2052	"M'\u6708'd'\u65E5'"
2060	"dd-MMM-yy"
2060	"H' h 'mm"
2060	"H' h 'm' min '"
2060	"H' h 'm' min 's' s '"
2067	"H.mm' u.'"
2070	"HH'H'mm'm'"
3079	"d.MMMyyyy"
3082	"HH'H'mm'\'"

Locale	Date and time format
3084	"d MMMM"
3084	"H' h 'mm"
5132	"HH' h 'mm"
6156	"HH' h 'mm"
7177	"dd MMMM"

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.12 disableEditing

The following controls MAY contain the **disableEditing** attribute in their XSLT representation:

- Custom controls
- File attachment (section 2.4.1.11)
- Rich text box (section 2.4.1.17)
- Text box (section 2.4.1.20)

The following controls MUST contain this attribute in their XSLT representation:

- Expression box (section 2.4.1.10)
- Hyperlink (section 2.3.1.8)
- Section Control and Optional Section Control (section 2.4.1.18) with XML digital signatures.

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **disableEditing** attribute MUST be "yes", which disables editing for the control.

If this attribute is unspecified, the behavior MUST be to enable editing for the control.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.13 enabledProperty

Custom controls MAY contain the **enabledProperty** attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **enabledProperty** attribute MUST map to a property exposed by the custom control.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.14 enabledValue

Custom controls MAY contain the **enabledValue** attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **enabledValue** attribute MUST be one of the following:

- **true**: Enables the custom control using the enabledProperty (section 2.4.2.13).
- **false**: Enable the custom control using the enabledProperty.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.15 ghosted

The **ghosted** attribute is associated with a client-only feature and MUST be ignored by the form server.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.16 ictID

The **ictID** attribute is associated with a client-only feature and MUST be ignored by the form server.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.17 ictVersion

The **ictVersion** attribute is associated with a client-only feature and MUST be ignored by the form server.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.18 inline

The **inline** attribute is associated with a client-only feature and MUST be ignored by the form server.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.19 innerCtrl

Date picker controls, as specified in section 2.4.1.8, MUST contain the **innerCtrl** attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their XSLT representations.

This attribute MUST have one of the following values, which specify the respective component types:

- **_DTButton:** A date picker button.
- **_DTText:** An editable text field.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.20 inputscope

The **inputscope** attribute is associated with a client-only feature and MUST be ignored by the form server.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.21 inputScopeId

The **inputScopeId** attribute is associated with a client-only feature and MUST be ignored by the form server.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.22 layoutText

The **layoutText** attribute is associated with a client-only feature and MUST be ignored by the form server.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.23 linkedToMaster

The **linkedToMaster** attribute is associated with a client-only feature and MUST NOT be present.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.24 masterID

The **masterID** attribute is associated with a client-only feature and MUST be ignored by the form server.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.25 masterName

The **masterName** attribute is associated with a client-only feature and MUST NOT be present.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.26 num

The following controls MAY contain this attribute in their XSLT representation:

- Date picker (section 2.4.1.8)
- Expression box (section 2.4.1.10)
- Text box (section 2.4.1.20)

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of this attribute MUST be set as specified in the **boundProp** attribute (section 2.4.2.9).

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.27 offValue

The **offValue** attribute is applicable to a check box control, as specified in section 2.3.1.2. Inclusion of this attribute is based on the **onValue** attribute, as specified in section 2.4.2.28, as follows:

- Check box controls MUST contain this attribute in their XSLT representation if the **onValue** attribute is unspecified.
- Check box controls MAY contain this attribute in their XSLT representation if the **onValue** attribute is specified.

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of this attribute MUST be a value that is valid for the XML field's data type.

The value of this attribute MUST be a different string value than the string value specified for the **onValue** attribute.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.28 onValue

The **onValue** attribute is applicable to a check box control, as specified in section 2.3.1.2. Inclusion of this attribute is based on the **offValue** attribute, as specified in section 2.4.2.27, as follows:

- Check box controls MUST contain this attribute in their XSLT representation if the **offValue** attribute is unspecified.
- Check box controls MAY contain this attribute in their XSLT representation if the **offValue** attribute is specified.

Option button controls, as specified in section 2.4.1.14, MAY contain this attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **onValue** attribute MUST be a value that is valid for the XML field's data type.

For the check box control, the value of this attribute MUST be a different string value than the string value specified for the **offValue** attribute.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.29 postbackModel

The following controls MAY contain the **postbackModel** attribute in their XSLT representation:

- Button (section 2.4.1.5)
- Check box (section 2.4.1.6)
- Date picker (section 2.4.1.8)

- Drop-down list (section 2.4.1.9)
- List box (section 2.4.1.13)
- Option button (section 2.4.1.14)
- Optional section (section 2.4.1.18)
- Repeating section (section 2.4.1.15)
- Repeating table (section 2.4.1.16)
- Rich text box (section 2.4.1.17)
- Section (section 2.4.1.18)
- Text box (section 2.4.1.20)

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **postbackModel** attribute MUST be one of the following:

- **always:** Always send data to the form server when the XML field value in the control is changed.
- **auto:** Dependent on protocol server implementation. Send data to the form server when the XML field value in the control is changed only if the protocol server implementation requires it.
- **never:** Never send data to the form server when the XML field value in the control is changed.

If this attribute is unspecified, the behavior MUST be the same as an attribute value of "auto".

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.30 ref

The **ref** attribute is associated with a client-only feature and MUST be ignored by the form server.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.31 SignatureBlock

Section and optional section controls (section 2.4.1.18) MAY contain the **SignatureBlock** attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **SignatureBlock** attribute MUST be equal to the **name** attribute of the **signedDataBlock** (section 2.2.126) that exists in the form.

The value of this attribute MUST be equal to the value specified for the **SignedSectionName** attribute (section 2.4.2.33).

The value of this attribute MUST begin with an alphabetic or underscore character, and MUST contain only alphanumeric, underscore, hyphen, and period characters.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.32 SignedSectionDisplaySignatures

Section and optional section controls (section 2.4.1.18) MAY contain the **SignedSectionDisplaySignatures** attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of **SignedSectionDisplaySignatures** attribute MUST be "true", which specifies to show signatures for the control.

If this attribute is unspecified, the behavior MUST be to not show signatures for the control.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.33 SignedSectionName

Section and optional section controls (section 2.4.1.18) MAY contain the **SignedSectionName** attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **SignedSectionName** attribute MUST be equal to the **name** attribute of the **signedDataBlock** (section 2.2.126) that exists in the form.

The value of this attribute MUST be equal to the value specified for the **SignatureBlock** attribute in section 2.4.2.31.

The value of this attribute MUST begin with an alphabetic or underscore character, and MUST contain only alphanumeric, underscore, hyphen, and period characters.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.34 value

The following controls MUST contain the **value** attribute in their XSLT representation:

- Check box (section 2.4.1.6)
- Option button (section 2.4.1.14)

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **value** attribute MUST be set as specified for the **boundProp** attribute in section 2.4.2.9.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.35 xctname

All controls MUST contain the **xctname** attribute in their XSLT representation.

The value of this attribute MUST be one of the values in the following table for **built-in control**.

Control	"xctname" value
Button (section 2.4.1.5)	Button
Check box (section 2.4.1.6)	CheckBox
Date picker (section 2.4.1.8)	DTPicker (specifies the date picker control) DTPicker_DTButton (specifies the date picker button in the date picker control) DTPicker_DTText (specifies the editable text field in the date picker control)
Drop-down list box (section 2.4.1.9)	DropDown
Expression box (section 2.4.1.10)	ExpressionBox
File attachment (section 2.4.1.11)	FileAttachment
Hyperlink (section 2.4.1.12)	Hyperlink
List box (section 2.4.1.13)	ListBox
Option button (section 2.4.1.14)	OptionButton
Optional section (section 2.4.1.18)	Section
Repeating section (section 2.4.1.15)	RepeatingSection
Repeating table (section 2.4.1.16)	RepeatingTable
Rich text box (section 2.4.1.17)	RichText

Control	"xctname" value
Section (section 2.4.1.18)	Section
Text box (section 2.4.1.20)	PlainText

For custom controls, the value of the **xctname** attribute MUST be the control's **class identifier (CLSID)**, and MUST follow this structure:

```
{{clsid}}
```

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.2.36 xmlToEdit

The following controls MAY contain the **xmlToEdit** attribute in their XSLT representation:

- Optional section (section 2.4.1.18)
- Repeating section (section 2.4.1.15)
- Repeating table (section 2.4.1.16)
- Section (section 2.4.1.18)

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **xmlToEdit** attribute MUST be equal to the name of an **xmlToEdit** element (section 2.2.124) that exists in the form.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

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

2.4.3 XSL Function Extensions

This section describes the additional functions provided with the XSL function extension to be used in the XSL file.

Namespace	Description
Msxsl (section 2.4.3.1)	A **string** comparison function.
xdDate (section 2.4.3.2	A set of date- and time-related functions.
xdEnvironment (section 2.4.3.3)	A set of functions that are related to the environment in which the form is being filled out.
xdFormatting (section 2.4.3.4)	A set of string formatting functions..

Namespace	Description
xdImage (section 2.4.3.5)	A set of image related functions.
xdMath (section 2.4.3.6)	A set of mathematical functions.
xdUser (section 2.4.3.7)	A set of functions that are related to the user who is filling out the form (1).
xdUtil (section 2.4.3.8	Generic helper tools.
xdXDocument (section 2.4.3.9)	A set of functions that are related to the data of the form (1) being filled out.

2.4.3.1 msxsl

Microsoft XPath Extension Functions, as specified in [MSDN-XPATH], specify a number of functions, one of which is supported by **msxsl**.

2.4.3.1.1 string-compare

Function signature: **number** msxsl:string-compare(*First String*, *Second String*)

This function MUST take two parameters:

- *First String***:** This parameter MUST be a **string**. **String** is specified in [XPATH], section 3.6.
- *Second String***:** This parameter MUST be a **string**. **String** is specified in [XPATH], section 3.6.

The function MUST compare the lexicographical order of the two **strings** passed as parameters. It MUST return zero if they are equivalent strings; it MUST return 1 if the second **string** comes before the first in lexicographical order, and it MUST return -1 if the first **string** comes before the second in lexicographical order. **Number** is specified in [XPATH], section 3.5. **String** is specified in [XPATH] section 3.6.

2.4.3.2 xdDate

xdDate contains a set of date- and time-related functions.

2.4.3.2.1 AddDays

Function Signature: **string** xdDate:addDays(*date*, *days*)

This function MUST take two parameters:

- *date***:** This parameter MUST be either a **string** or an XPath expression that returns a node. The **string** or the value of the node MUST be a date in ISO 8601 format, as specified in [ISO-8601], to be a valid parameter. XPath expression is specified in [XPATH].
- *days***:** This parameter MUST be either a **string** or an XPath expression that returns a node. The **string** or the value of the node MUST be a **number**, as specified in [XPATH], section 3.5 to be a valid parameter.

The function MUST increase the given date by the given number of days and return the resulting date in ISO format. It MUST return an empty string if both parameters are empty strings. It MUST return a **string** with the value of "#ERR?" if either of the parameters have an invalid value. **String** is specified in [XPATH] section 3.6.

2.4.3.2.2 AddSeconds

Function Signature: **string** xdDate:addSeconds(*time*, *seconds*)

This function MUST take two parameters:

- *Time*: This parameter MUST be either a **string** or an XPath expression that returns a node. The **string** or the value of the node MUST be a time in ISO 8601 format, as specified in [ISO-8601], to be a valid parameter. XPath expression is specified in [XPATH].
- *Seconds*: This parameter MUST be either a **string** or an XPath expression that returns a node. The **string** or the value of the node MUST be a **number**, as specified in [XPATH] section 3.5, to be a valid parameter.

The function MUST increase the given time by the given number of seconds and return the resulting time in ISO format. It MUST return an empty string if both parameters are empty strings. It MUST return a **string** with the value of "#ERR?" if either of the parameters have an invalid value. **String** is specified in [XPATH] section 3.6.

2.4.3.2.3 Now

Function Signature: **string** xdDate:now()

This function MUST NOT take any parameters. It MUST return the current system date and time in ISO 8601 format, as specified in [ISO-8601]. **String** is specified in [XPATH] section 3.6.

2.4.3.2.4 Today

Function Signature: **string** xdDate:today()

This function MUST NOT take any parameters. It MUST return the current system date in ISO 8601 format, as specified in [ISO-8601]. **String** is specified [XPATH] section 3.6.

2.4.3.3 xdEnvironment

xdEnvironment contains a set of functions that are related to the environment in which the form is being filled out.

2.4.3.3.1 IsBrowser

Function Signature: **boolean** xdEnvironment:IsBrowser()

This function MUST NOT take any parameters. It MUST return TRUE if the form is being filled out with a Web browser, FALSE otherwise. **Boolean** is specified in [XPATH] section 3.4.

2.4.3.3.2 IsMobile

Function Signature: **boolean** xdEnvironment:IsMobile()

This function MUST NOT take any parameters. It MUST return TRUE if the form is being filled out with a **mobile device**, FALSE otherwise. **Boolean** is specified in [XPATH] section 3.4.

2.4.3.4 xdFormatting

xdFormatting contains a set of **string** formatting functions. It is not supported and MUST be ignored.

2.4.3.5 xdImage

xdImage contains a set of image-related functions. It is not supported and MUST be ignored.

2.4.3.6 xdMath

xdMath contains a set of mathematical functions.

2.4.3.6.1 Avg

Function Signature: **number** xdMath:avg(*XPath Expression*)**)**

This function MUST take one parameter:

- *XPath Expression*: This parameter MUST be an XPath expression that returns a **node-set**. XPath expression is specified in [XPATH]. **Node-set** is specified in [XPATH] section 3.3.

The **string** value of every node in the **node-set** MUST be calculated using the **string** function specified in [XPATH] section 4.2. The output of the **string** function MUST be converted to a **number** using the **number** function specified in [XPATH] section 4.4. If the number function returns **NAN** for any node of the node-set, the **avg** function MUST return **NAN**. The output of the **number** function MUST be used as the numerical value of the node.

The **avg** function MUST return the average value of all of the numerical values in the given **node-set**. **Number** and **NAN** are specified in [XPATH] section 3.5.

2.4.3.6.2 Eval

Function Signature: **node-set** xdMath:eval(*XPath Expression, XSLT Expression*)

This function MUST take two parameters:

- *XPath Expression*: MUST be an XPath expression that returns a **node-set**. XPath expression is specified in [XPATH]. **Node-set** is specified in [XPATH] section 3.3.
- *XSLT Expression*: MUST be a valid XSLT expression, as specified in [W3C-XSLT] section 4.

The function MUST apply the XSLT expression to every node in the **node-set** and return the resulting **node-set**.

2.4.3.6.3 Max

Function Signature: **number** xdMath:max(*XPath Expression*)

This function MUST take one parameter:

- *XPath Expression*: MUST be an XPath expression that returns a **node-set**. XPath expression is specified in [XPATH]. **Node-set** is specified in [XPATH] section 3.3.

The **string** value of every node of the **node-set** MUST be calculated using the **string** function specified in [XPATH] section 4.2. The output of the **string** function MUST be converted to a **number** using the **number** function specified in [XPATH] section 4.4. If the **number** function returns **NAN** for any node of the **node-set**, the **max** function MUST return **NAN**. The output of the **number** function MUST be used as the numerical value of the node.

The **max** function MUST return the numerical value that is greater than or equal to the value of every other item in the **node-set**. **Number** and **NAN** are specified in [XPATH] section 3.5.

2.4.3.6.4 Min

Function Signature: **number** xdMath:min(*XPath Expression*)

This function MUST take one parameter:

- *XPath Expression*: MUST be an XPath expression that returns a **node-set**. XPath expression is specified in [XPATH]. **Node-set** is specified in [XPATH] section 3.3.

The **string** value of every node of the **node-set** MUST be calculated using the **string** function specified in [XPATH] section 4.2. The output of the **string** function MUST be converted to a **number** using the **number** function specified in [XPATH] section 4.4. If the **number** function returns **NAN** for any node of the **node-set**, the **min** function MUST return **NAN**. The output of the **number** function MUST be used as the numerical value of the node.

The **min** function MUST return the numerical value that is smaller than or equal to the value of every other item in the given **node-set**. **Number** and **NAN** are specified in [XPATH] section 3.5.

2.4.3.6.5 Nz

Function Signature: **node-set** xdMath:nz(*XPath Expression*)

This function MUST take one parameter:

- *XPath Expression*: This parameter MUST be an XPath expression that returns a **node-set**. XPath expression is specified in [XPATH]. **Node-set** is specified in [XPATH] section 3.3.

The function MUST return a **node-set** that is identical to the given **node-set** with the exception that empty nodes are given the value zero ("0").

2.4.3.7 xdUser

xdUser contains a set of functions that are related to the user who is filling out the form.

2.4.3.7.1 get-UserName

Function Signature: **string** xdUser:get-UserName()

This function MUST NOT take any parameters. It MUST return the **user name** for the current user.

2.4.3.8 xdUtil

xdUtil contains generic helper tools.

2.4.3.8.1 Match

Function Signature: **boolean** xdUtil:match(*string, Regular Expression*)

This function MUST take two parameters:

- *String*: This parameter MUST be a **string**.
- *Regular Expression*: This parameter MUST be a valid XML regular expression.

This function MUST return TRUE if the input **string** conforms to the specified regular expression, or FALSE otherwise. **String** is specified in [XPATH] section 3.6. Regular expression is specified in [XMLSCHEMA1] Appendix F.

2.4.3.9 xdXDocument

xdXDocument contains a set of functions that are related to the data of the form being filled out.

2.4.3.9.1 get-dom

Function Signature: **node-set** xdXDocument:get-dom()

This function MUST NOT take any parameters. It MUST return a **node-set** that contains the main data source.

2.4.3.9.2 getDOM

Function Signature: **node-set** xdXDocument:getDOM(*Name*)

This function MUST take one parameter:

- **Name**: This parameter MUST be a **string**. It MUST be the name of a secondary data source.

This function MUST return the data object with the given name. **Node-set** is specified in [XPATH] section 3.3. **String** is specified in [XPATH] section 3.6.

2.4.3.9.3 getnamednodeproperty

Function Signature: **string** xdXDocument:getnamednodeproperty(*MainDOMNode, PropertyName, DefaultValue*)

The function MUST take the following parameters:

- *MainDOMNode*: This parameter MUST be an XPath expression that returns a non-attribute node in the main data source, for which a **named property** is to be set.
- *PropertyName*: This parameter MUST be a **string**. It specifies the name of the property whose value is to be returned.
- *DefaultValue*: This parameter MUST be a **string**. It specifies the default value to be returned if the property has not been set.

This function provides a mechanism to retrieve string data that is stored on the non-attribute nodes of the main data source. The protocol only defines a mechanism to retrieve the data. The protocol server SHOULD provide a mechanism to store **string** data.

This function MUST return the value of the named property that is stored in the specified XML node. **String** is specified in [XPATH] section 3.6.

2.5 Print View Files (XSLT) Specification

The XSLT file representing a print view MUST conform to the format specified in section 2.4. A print view MUST be associated with a form view using the **printView** attribute of the **view** element (section 2.2.123) in the form definition (.xsf) file. See section 3.5 for an example.

2.6 Submit Files (XML) Specification

For each **input** element (section 2.2.43) inside of a **webServiceAdapter** element (section 2.2.39) in the form definition (.xsf) file, there MUST be a corresponding XML file defined. This SHOULD be accomplished by naming the files according to the pattern "Submit[0-9]*.xml". The first file SHOULD be "Submit.xml", and the subsequent files SHOULD be "Submit1.xml", "Submit2.xml", "Submit3.xml", and so on, with the number increasing by one for each additional file. Each of these files MUST be referenced at the **input** element inside of the form definition (.xsf) file. All of these files MUST be contained inside of the form template.

Each Submit.xml file MUST contain only the following types:

- myFields

- dataFields, including the Web service method template specified in 2.6.2.

2.6.1 myFields

The **myFields** element MUST be the top-level element in this XML file. Additionally, it MUST do the following:

- Specify ‘xmlns:dfs="http://schemas.microsoft.com/office/infopath/2003/dataFormSolution"’ as a namespace.
- Specify any additional namespaces required for the Web service method template specified in section 2.6.2.
- Have a single child **dataFields** element.

Child Elements
dataFields

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xs:element name="myFields" type="dfs:MyFieldType"/>
<xs:complexType name="MyFieldType">
  <xs:sequence>
    <xs:element ref="dfs:dataFields" minOccurs="1" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>
```

2.6.2 dataFields

The **dataFields** element MUST have exactly one child node that is the Web service method template. This template specifies the method and parameter fields names used when submitting to the Web service. The template MUST validate against the XML schema of the method in the Web service. This XML schema is defined in the Web service **Web Services Description Language (WSDL)**.

Parent Elements
myFields

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xs:element name="dataFields" type="dfs:DataFieldType"/>
<xs:complexType name="DataFieldType">
  <xs:sequence>
    <!-- Web Service Template -->
    <xs:any processContents="skip" minOccurs="1" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>
```

2.7 Template.XML Specification

The template.xml file MUST be an instance of the XML schema document, as specified in section 2.3, and MUST be a valid form file, as specified in [MS-IPFFX] section 2.1.

Initial values for the fields in a new form file MUST be stored in and loaded from this file. If a pre-existing form file is opened, contents of this file MUST be ignored.

2.8 Upgrade.XSL Specification

The upgrade.xsl file is an XSLT that MUST conform to the XSLT specification, as specified in [W3C-XSLT], with the exception of the **msxsl:node-set** function. Upgrade.xsl MUST be applied by the form server when opening an existing form file if upgrade.xsl is present within the form template (.xsn) file.

The upgrade.xsl file MUST use the **msxsl:node-set** function to create new empty XML node sets in cases that a new XML node set is required. The **msxsl:node-set** function is specified in the following section.

2.8.1 MSXSL:Node-Set()

The **msxsl:node-set** function converts a result tree fragment into an XML node set. The resulting XML node set always contains a single XML node and is the root XML node of the tree. It MUST take one argument, *$var*, which is the result tree fragment to be converted.

3 Structure Examples

This section contains examples of the following:

- InfoPath form template
- Form definition (XSF) files
- XML schema (XSD) files
- Form view (XLS) files
- Print view (XLST) files
- Submit (XML) files
- Template.XML
- Upgrade.XSL

3.1 The InfoPath Form Template Format

3.1.1 Simple Form Template

This example describes a simple form template (.xsn) file.

The contents of the Simple.xsn file are as follows:

- manifest.xsf
- myschema.xsd
- template.xml
- sampledata.xml
- view1.xsl

The preceding files represent a very simple form with just one form view of the data:

- The manifest.xsf file is the first file in the cabinet (.cab) file, and within it lists the other four files in the form template (.xsn) file.
- The myschema.xsd file is an example of the primaryschema.xsd file, which is required to define the XML schema for the data in the form.
- The sampledata.xml file needs to be present, but the form server ignores it.
- The template.xml file also needs to be present. It provides the default values for the form (1).
- The view1.xsl is a view.xsl file. At least one is required. It represents how the form (1) is displayed, including which fields appear and in what order.

3.1.2 Complex Form Template

This example describes a slightly more complex form template (.xsn) file.

The contents of the Complex.xsn file are as follows:

- manifest.xsf
- myschema.xsd
- template.xml
- sampledata.xml
- view1.xsl
- view2.xsl
- IPTemplate_bkgd.gif
- 741C3E77.gif
- upgrade.xsl
- 70482F6B.gif

The files listed here include more than just the minimum for a form template (.xsn) file:

- The manifest.xsf file is the first file in the cabinet (.cab) file, and within it lists the other files in the form template (.xsn) file. Myschema.xsd, template.xml and sampledata.xml are also all present as required. See section 3.1.
- This form uses two view.xsl files to represent the form:
  - view1.xsl
  - view2.xsl
- There is an upgrade.xsl file, which is used by the form server to upgrade older form files to the newest XML schema.
- There are three resource files. These images are used when displaying the form (1):
  - IPTemplate_bkgd.gif
  - 741C3E77.gif
  - 70482F6B.gif

3.2 Form Definition File (XSF) Examples

This sample form definition (.xsf) file specifies that this is a browser-compatible form template containing an ActiveX Data Objects (ADO) data adapter that queries a SQL database. The contained **files** element (section 2.2.97) and **documentSchemas** element (section 2.2.60) specify that there are three XML schema documents that are used to verify the form definition (.xsf) file and form file. There is also an [W3C-XML] file containing sample data for creating the form file and an [W3C-XSLT] file specifying how the form view is generated and displayed.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
This file is automatically created and modified by Microsoft Office InfoPath.
Changes made to the file outside of InfoPath might be lost if the form template is modified
in InfoPath.
-->
<xsf:xDocumentClass trustSetting="automatic" solutionFormatVersion="2.0.0.0"
dataFormSolution="yes" solutionVersion="1.0.0.6" productVersion="12.0.0" publishUrl=""
name="urn:schemas-microsoft-com:office:infopath:Unpacked:-dataFormSolution"
xmlns:xsf="http://schemas.microsoft.com/office/infopath/2003/solutionDefinition"
xmlns:xsf2="http://schemas.microsoft.com/office/infopath/2006/solutionDefinition/extensions
" xmlns:msxsl="urn:schemas-microsoft-com:xslt"
```

```
xmlns:xd="http://schemas.microsoft.com/office/infopath/2003"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xdUtil="http://schemas.microsoft.com/office/infopath/2003/xslt/Util"
xmlns:xdXDocument="http://schemas.microsoft.com/office/infopath/2003/xslt/xDocument"
xmlns:xdMath="http://schemas.microsoft.com/office/infopath/2003/xslt/Math"
xmlns:xdDate="http://schemas.microsoft.com/office/infopath/2003/xslt/Date"
xmlns:xdExtension=
xmlns:xdEnvironment="http://schemas.microsoft.com/office/infopath/2006/xslt/environment"
xmlns:xdUser="http://schemas.microsoft.com/office/infopath/2006/xslt/User"
xmlns:q="http://schemas.microsoft.com/office/infopath/2003/ado/queryFields"
xmlns:d="http://schemas.microsoft.com/office/infopath/2003/ado/dataFields"
xmlns:dfs="http://schemas.microsoft.com/office/infopath/2003/dataFormSolution"
xmlns:my="http://schemas.microsoft.com/office/infopath/2003/myXSD/2007-03-19T15:02:59"
xmlns:xdado="http://schemas.microsoft.com/office/infopath/2003/adomapping">
  <xsf:package>
    <xsf:files>
      <xsf:file name="schema.xsd">
        <xsf:fileProperties>
          <xsf:property name="editability" type="string" value="none"></xsf:property>
          <xsf:property name="namespace" type="string"
value="http://schemas.microsoft.com/office/infopath/2003/dataFormSolution"></xsf:property>
          <xsf:property name="rootElement" type="string" value="myFields"></xsf:property>
          <xsf:property name="useOnDemandAlgorithm" type="string"
value="yes"></xsf:property>
        </xsf:fileProperties>
      </xsf:file>
      <xsf:file name="schema1.xsd">
        <xsf:fileProperties>
          <xsf:property name="namespace" type="string"
value="http://schemas.microsoft.com/office/infopath/2003/ado/dataFields"></xsf:property>
          <xsf:property name="editability" type="string" value="none"></xsf:property>
        </xsf:fileProperties>
      </xsf:file>
      <xsf:file name="schema2.xsd">
        <xsf:fileProperties>
          <xsf:property name="namespace" type="string"
value="http://schemas.microsoft.com/office/infopath/2003/ado/queryFields"></xsf:property>
          <xsf:property name="editability" type="string" value="none"></xsf:property>
        </xsf:fileProperties>
      </xsf:file>
      <xsf:file name="template.xml"></xsf:file>
      <xsf:file name="sampledata.xml">
        <xsf:fileProperties>
          <xsf:property name="fileType" type="string" value="sampleData"></xsf:property>
        </xsf:fileProperties>
      </xsf:file>
      <xsf:file name="view1.xsl">
        <xsf:fileProperties>
          <xsf:property name="lang" type="string" value="1033"></xsf:property>
          <xsf:property name="queryView" type="string" value="yes"></xsf:property>
          <xsf:property name="componentId" type="string" value="12"></xsf:property>
          <xsf:property name="xmlToEditName" type="string" value="12"></xsf:property>
          <xsf:property name="mode" type="string" value="1"></xsf:property>
        </xsf:fileProperties>
      </xsf:file>
    </xsf:files>
  </xsf:package>
  <xsf:importParameters enabled="yes"></xsf:importParameters>
  <xsf:extensions>
    <xsf:extension name="SolutionDefinitionExtensions">
      <xsf2:solutionDefinition runtimeCompatibility="client server"
allowClientOnlyCode="no">
        <xsf2:offline openIfQueryFails="yes" cacheQueries="yes"></xsf2:offline>
        <xsf2:server isPreSubmitPostBackEnabled="no" isMobileEnabled="no" formLocale="en-
US"></xsf2:server>
      </xsf2:solutionDefinition>
    </xsf:extension>
  </xsf:extensions>
  <xsf:views default="View 1">
    <xsf:view name="View 1" caption="View 1">
```

```
      <xsf:mainpane transform="view1.xsl"></xsf:mainpane>
      <xsf:editing>
        <xsf:xmlToEdit name="DimCustomer_7"
item="/dfs:myFields/dfs:dataFields/d:DimCustomer" container="/dfs:myFields">
          <xsf:editWith caption="DimCustomer" xd:autogeneration="template"
component="xCollection">
            <xsf:fragmentToInsert>
              <xsf:chooseFragment parent="dfs:dataFields" innerFragment="d:DimCustomer">
                <d:DimCustomer CustomerKey="" Title="" FirstName="" MiddleName=""
LastName="" BirthDate="" MaritalStatus=""

Suffix="" Gender="" EmailAddress="" AddressLine1="" AddressLine2=""></d:DimCustomer>
              </xsf:chooseFragment>
            </xsf:fragmentToInsert>
          </xsf:editWith>
        </xsf:xmlToEdit>
      </xsf:editing>
      <xsf:menuArea name="msoInsertMenu">
        <xsf:menu caption="&amp;Section">
          <xsf:button action="xCollection::insert" xmlToEdit="DimCustomer_7"
caption="DimCustomer"></xsf:button>
        </xsf:menu>
      </xsf:menuArea>
      <xsf:menuArea name="msoStructuralEditingContextMenu">
        <xsf:button action="xCollection::insertBefore" xmlToEdit="DimCustomer_7"
caption="Insert DimCustomer before" showIf="immediate"></xsf:button>
        <xsf:button action="xCollection::insertAfter" xmlToEdit="DimCustomer_7"
caption="Insert DimCustomer after" showIf="immediate"></xsf:button>
        <xsf:button action="xCollection::remove" xmlToEdit="DimCustomer 7" caption="Remove
DimCustomer" showIf="immediate"></xsf:button>
        <xsf:button action="xCollection::insert" xmlToEdit="DimCustomer 7" caption="Insert
DimCustomer" showIf="immediate"></xsf:button>
      </xsf:menuArea>
    </xsf:view>
  </xsf:views>
  <xsf:applicationParameters application="InfoPath Design Mode">
    <xsf:solutionProperties
fullyEditableNamespace="http://schemas.microsoft.com/office/infopath/2003/myXSD/2007-03-
19T15:02:59" lastOpenView="view1.xsl"

lastVersionNeedingTransform="1.0.0.3"></xsf:solutionProperties>
  </xsf:applicationParameters>
  <xsf:documentSchemas>
    <xsf:documentSchema rootSchema="yes"
location="http://schemas.microsoft.com/office/infopath/2003/dataFormSolution
schema.xsd"></xsf:documentSchema>
    <xsf:documentSchema
location="http://schemas.microsoft.com/office/infopath/2003/ado/dataFields
schema1.xsd"></xsf:documentSchema>
    <xsf:documentSchema
location="http://schemas.microsoft.com/office/infopath/2003/ado/queryFields
schema2.xsd"></xsf:documentSchema>
  </xsf:documentSchemas>
  <xsf:fileNew>
    <xsf:initialXmlDocument caption="Unpacked"
href="template.xml"></xsf:initialXmlDocument>
  </xsf:fileNew>
  <xsf:query>
    <xsf:adoAdapter connectionString="Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist
Security Info=True;Initial Catalog=AdventureWorksDW;Data

Source=[Source];Use Procedure for Prepare=1;Auto Translate=True;Packet
Size=4096;Workstation ID=[ID];Use Encryption for Data=False;Tag with

column collation when possible=False" commandText="select

&amp;quot;CustomerKey&amp;quot;,&amp;quot;Title&amp;quot;,&amp;quot;FirstName&amp;quot;,&amp;quot;MiddleName&amp;quot;,&amp;quo
t;LastName&amp;quot;,&amp;quot;BirthDate&amp;quot;,&amp;quot;MaritalStatus&amp;quot;,&amp;quot;Suffix&amp;quot;,&q
```

```
uot;Gender&quot;,&quot;EmailAddress&quot;,&quot;AddressLine1&quot;,&quot;AddressLine2&quot;
from &quot;dbo&quot;.&quot;DimCustomer&quot; as &quot;DimCustomer&quot;"

queryAllowed="yes" name="Main connection" submitAllowed="no"></xsf:adoAdapter>
  </xsf:query>
</xsf:xDocumentClass>
```

3.2.1 XSF Extension Examples

This sample **solutionDefinition** element (section 2.2.147.8) specifies that this is a browser-compatible form template and its contents are verified at http://www.someserver.com/verificationService. The contained **offline** element (section 2.2.147.24) specifies that the form is loaded even if contained online queries fail and that the results of any queries are cached locally. The contained **server** element (section 2.2.147.9) specifies:

- The form template is not compatible with mobile Web browsers.
- The form view is rendered in US English.
- The form does not postback to the protocol server before submitting the form file.

Note that all elements are in the **xsf2** namespace.

```
<xsf2:solutionDefinition runtimeCompatibility="client server"
runtimeCompatibilityURL="http://www.someserver.com/verificationService"
verifyOnServer="yes">
  <xsf2:offline openIfQueryFails="yes" cacheQueries="yes"></xsf2:offline>
  <xsf2:server isPreSubmitPostBackEnabled="no" isMobileEnabled="no" formLocale="en-
US"></xsf2:server>
</xsf2:solutionDefinition>
```

3.3 XML Schema Files (XSD) Examples

Section 2.3 provides sample XSD constructs for supported controls.
The following example is an XSD:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xsd:schema targetNamespace="http://schemas.microsoft.com/office/infopath/2003/myXSD/2008-03-
17T22:37:33" xmlns:my="http://schemas.microsoft.com/office/infopath/2003/myXSD/2008-03-
17T22:37:33" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
            <xsd:element name="myFields">
                        <xsd:complexType>
                                    <xsd:sequence>
                                                <xsd:element ref="my:field1" minOccurs="0"/>
                                                <xsd:element ref="my:group1" minOccurs="0"/>
                                                <xsd:element ref="my:field3" minOccurs="0"/>
                                    </xsd:sequence>
                                    <xsd:anyAttribute processContents="lax"
namespace="http://www.w3.org/XML/1998/namespace"/>
                        </xsd:complexType>
            </xsd:element>
            <xsd:element name="field1" type="xsd:string"/>
            <xsd:element name="group1">
                        <xsd:complexType>
                                    <xsd:sequence>
                                                <xsd:element ref="my:group2" minOccurs="0"
maxOccurs="unbounded"/>
                                    </xsd:sequence>
                        </xsd:complexType>
            </xsd:element>
            <xsd:element name="group2">
                        <xsd:complexType>
                                    <xsd:sequence>
```

```
                                                <xsd:element ref="my:field2" minOccurs="0"/>
                                            </xsd:sequence>
                                        </xsd:complexType>
                                    </xsd:element>
                                    <xsd:element name="field2" type="xsd:string"/>
                                    <xsd:element name="field3" nillable="true" type="xsd:base64Binary"/>
</xsd:schema>
```

The first element represented in the XSD is **myFields**, which is the root element for all the other elements that represent a control in the XSD. **myFields** contains a reference to **my:field1**, **my:group1** and **my:field3**, which are defined as follows:

- **my:field1** represents a text box control, as specified in section 2.3.1.16, in the XSD that can have a **string** content.
- **my:group1** contains another group, **my:group2**. **my:group2** is a repeating element. This is used to represent repeating controls, such as a repeating section control specified in section 2.3.1.11.
- **my:field2** represents the control inside the repeating control, which is a text box control.
- **my:field3** is a file attachment control, as specified in section 2.3.1.7.

3.4 Form View Files (XSL) Examples

This section contains XSL examples for controls, attributes, style definitions, and function extensions, as specified in section 2.4.

3.4.1 Control representation

This section contains example XSL fragments for all of the controls specified in section 2.4.1. Each fragment provides an example of how a control can be structured with features such as conditional formatting, data formatting, or retrieving selection options from a data source.

3.4.1.1 Button Control

The following XSL examples are button controls, as specified in section 2.3.1.1.

The following example is a button control with conditional formatting. The **name** attribute is set to the value of **my:field1**. This means that the button's display text is the value of **my:field1**. Conditional formatting is set such that if the value of **my:field3** is equal to "true", the control is hidden.

```
<input class="langFont" title="" type="button" xd:xctname="Button" xd:CtrlId="CTRL1_5"
tabIndex="0">
    <xsl:attribute name="style">
        <xsl:choose>
            <xsl:when test="my:field3 = string(true())">DISPLAY: none</xsl:when>
        </xsl:choose>
    </xsl:attribute>
    <xsl:attribute name="value">
        <xsl:value-of select="my:field1"/>
    </xsl:attribute>
</input>
```

The following example is a button control that is used to update the form content in the Web browser. The button display text is the value of **my:field1**. Conditional formatting is set such that if the value of **my:field2** is equal to "Red", the control has a red background color.

```
<input class="langFont" title="" style="BEHAVIOR: url(#default#ActionButton)" type="button"
xd:xctname="Button" xd:CtrlId="CTRL1_5" xd:action="updateForm" tabIndex="0">
    <xsl:attribute name="style">BEHAVIOR: url(#default#ActionButton);<xsl:choose>
            <xsl:when test="not(xdEnvironment:IsBrowser())">DISPLAY: none</xsl:when>
            <xsl:when test="my:field2 = &quot;Red&quot;">BACKGROUND-COLOR: #ff0000</xsl:when>
        </xsl:choose>
    </xsl:attribute>
    <xsl:attribute name="value">
        <xsl:value-of select="my:field1"/>
    </xsl:attribute>
</input>
```

The following example is a button control that is used to submit the form (1) data. The button display text is statically set to "Submit". This control has two conditional formatting settings, as follows:

- If the value of **my:field1** is "1", the control is disabled and has a yellow background color.
- If the value of **my:field2** is "abc", the button display text is bold and the control has an orange background.

```
<input class="langFont" title="Press to submit this form" style="BEHAVIOR:
url(#default#ActionButton)" accessKey="S" type="button" value="Submit" xd:xctname="Button"
xd:CtrlId="CTRL1_5" xd:action="submit" xd:postbackModel="always" tabIndex="0">
    <xsl:attribute name="style">BEHAVIOR: url(#default#ActionButton);<xsl:choose>
            <xsl:when test="my:field1 = 1">BACKGROUND-COLOR: #ffff00</xsl:when>
            <xsl:when test="my:field2 = &quot;abc&quot;">FONT-WEIGHT: bold; COLOR:
#ff6600</xsl:when>
        </xsl:choose>
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="my:field1 = 1">
            <xsl:attribute name="disabled">true</xsl:attribute>
        </xsl:when>
        <xsl:when test="my:field2 = &quot;abc&quot;"/>
    </xsl:choose>
</input>
```

The following example is a button control that is used to refresh the content of a secondary data source. The button display text is statically set to "Refresh". Conditional formatting is set such that if the value of **my:field1** is "1", the control is disabled and has a yellow background color.

```
<input class="langFont" title="" style="BEHAVIOR: url(#default#ActionButton)" type="button"
value="Refresh" xd:xctname="Button" xd:CtrlId="CTRL1_5" xd:action="refresh"
xd:auxDom="UserNameList" tabIndex="0">
    <xsl:attribute name="style">BEHAVIOR: url(#default#ActionButton);<xsl:choose>
            <xsl:when test="my:field1 = 1">BACKGROUND-COLOR: #ffff00</xsl:when>
        </xsl:choose>
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="my:field1 = 1">
            <xsl:attribute name="disabled">true</xsl:attribute>
        </xsl:when>
    </xsl:choose>
</input>
```

3.4.1.2 Check Box Control

The following XSL examples are check box controls, as specified in section 2.3.1.2.

The following example is a check box control with the value "1" if the control is not checked, and zero ("0") if the control is checked. When the user hovers over the control with the cursor, it displays the message "this is a checkbox".

```
<input class="xdBehavior_Boolean" title="this is a checkbox" type="checkbox"
xd:binding="my:field1" xd:boundProp="xd:value" xd:offValue="1" xd:onValue="0" tabIndex="0"
xd:xctname="CheckBox" xd:CtrlId="CTRL1">
    <xsl:attribute name="xd:value">
        <xsl:value-of select="my:field1" />
    </xsl:attribute>
    <xsl:if test="my:field1=&amp;quot;true&amp;quot;">
        <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
    </xsl:if>
</input>
```

The following example is a check box control with the value "false" if the control is not checked, and "true" if the control is checked. Conditional formatting is set such that if the control is checked, the control is disabled.

```
<input class="xdBehavior_Boolean" title="" type="checkbox" xd:binding="my:field2"
xd:boundProp="xd:value" xd:offValue="false" xd:onValue="true" tabIndex="0"
xd:xctname="CheckBox" xd:CtrlId="CTRL2">
    <xsl:choose>
        <xsl:when test="my:field2 = string(true())">
            <xsl:attribute name="disabled">true</xsl:attribute>
        </xsl:when>
    </xsl:choose>
    <xsl:attribute name="xd:value">
        <xsl:value-of select="my:field2" />
    </xsl:attribute>
    <xsl:if test="my:field2=&amp;quot;true&amp;quot;">
        <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
    </xsl:if>
</input>
```

3.4.1.3 Contact Selector Control

The following XSL example is a contact selector control, as specified in section 2.3.1.3, with conditional formatting. Conditional formatting is set such that if the value of **my:field1** is "false", the control is disabled.

```
<object class="xdActiveX" hideFocus="1" style="WIDTH: 288px; HEIGHT: 22px"
classid="clsid:61e40d31-993d-4777-8fa0-19ca59b6d0bb" tabIndex="0" tabStop="true"
xd:xctname="{{61e40d31-993d-4777-8fa0-19ca59b6d0bb}}" xd:CtrlId="CTRL1"
xd:bindingType="xmlNode" xd:bindingProperty="Value" xd:boundProp="xd:inline"
contentEditable="false" xd:binding="my:group1">
    <xsl:if test="function-available('xdImage:getImageUrl')">
        <xsl:attribute name="src"><xsl:value-of
select="xdImage:getImageUrl(my:group1)"/></xsl:attribute>
    </xsl:if>
    <xsl:choose>
        <xsl:when test="my:field1 = string(false())">
            <xsl:attribute name="xd:disableEditing">yes</xsl:attribute>
        </xsl:when>
    </xsl:choose>
    <param NAME="ButtonFont" VALUE="Verdana,10,0,400,0,0,0"/>
    <param NAME="ButtonText" VALUE="To..."/>
    <param NAME="DisplayNameXPath" VALUE="my:DisplayName"/>
    <param NAME="ObjectIdXPath" VALUE="my:AccountId"/>
    <param NAME="ObjectTypeXPath" VALUE="my:AccountType"/>
    <param NAME="SiteUrlXPath" VALUE="/Context/@siteUrl"/>
    <param NAME="SiteUrlDataSource" VALUE="Context"/>
```

```
    <param NAME="NewNodeTemplate"
VALUE="&lt;my:Person&gt;&#xA;&lt;my:DisplayName&gt;&lt;/my:DisplayName&gt;&#xA;&lt;my:Account
Id&gt;&lt;/my:AccountId&gt;&#xA;&lt;my:AccountType&gt;&lt;/my:AccountType&gt;&#xA;&lt;/my:Per
son&gt;"/>
    <param NAME="BackgroundColor" VALUE="2147483653"/>
    <param NAME="MaxLines" VALUE="4"/>
    <param NAME="Direction" VALUE="0"/>
</object>
```

3.4.1.4 Date Picker Control

The following XSL examples are date picker controls, as specified in section 2.3.1.4.

The following example is a date picker control where **xd:datafmt** is equal to ""date","dateFormat:Short Date;"". This formats the value of **my:field1** to be a short date.

```
<div class="xdDTPicker" title="" style="WIDTH: 130px" noWrap="1" xd:CtrlId="CTRL1"
xd:xctname="DTPicker">
    <span class="xdDTText xdBehavior FormattingNoBUI" hideFocus="1" contentEditable="true"
xd:xctname="DTPicker_DTText" xd:datafmt="&amp;quot;date&amp;quot;,&amp;quot;dateFormat:Short Date;&amp;quot;"
xd:boundProp="xd:num" xd:binding="my:field1" tabIndex="0" xd:innerCtrl=" DTText">
        <xsl:attribute name="xd:num">
            <xsl:value-of select="my:field1" />
        </xsl:attribute>
        <xsl:choose>
            <xsl:when test="function-available('xdFormatting:formatString')">
                <xsl:value-of
select="xdFormatting:formatString(my:field1,&amp;quot;date&amp;quot;,&amp;quot;dateFormat:Short
Date;&amp;quot;)" />
            </xsl:when>
            <xsl:otherwise>
                <xsl:value-of select="my:field1" />
            </xsl:otherwise>
        </xsl:choose>
    </span>
    <button class="xdDTButton" xd:xctname="DTPicker_DTButton" xd:innerCtrl="_DTButton"
tabIndex="-1">
        <img src="res://infopath.exe/calendar.gif" />
    </button>
</div>
```

The following example is a date picker control with conditional formatting. Conditional formatting is set such that if the value of **my:field3** is "1900-01-01", the text for the control is bold and strikethrough.

```
<div class="xdDTPicker" title="" style="WIDTH: 130px" noWrap="1" xd:CtrlId="CTRL3"
xd:xctname="DTPicker">
    <span class="xdDTText xdBehavior FormattingNoBUI" hideFocus="1" contentEditable="true"
xd:xctname="DTPicker_DTText" xd:datafmt="&amp;quot;date&amp;quot;,&amp;quot;dateFormat:Short Date;&amp;quot;"
xd:boundProp="xd:num" xd:binding="my:field3" tabIndex="0" xd:innerCtrl="_DTText">
        <xsl:attribute name="style">
            <xsl:choose>
                <xsl:when test="my:field3 = &amp;quot;1900-01-01&amp;quot;">FONT-WEIGHT: bold; TEXT-
DECORATION: line-through</xsl:when>
            </xsl:choose>
        </xsl:attribute>
        <xsl:attribute name="xd:num">
            <xsl:value-of select="my:field3" />
        </xsl:attribute>
        <xsl:choose>
            <xsl:when test="function-available('xdFormatting:formatString')">
                <xsl:value-of
select="xdFormatting:formatString(my:field3,&amp;quot;date&amp;quot;,&amp;quot;dateFormat:Short
Date;&amp;quot;)" />
            </xsl:when>
```

```
            <xsl:otherwise>
                <xsl:value-of select="my:field3" />
            </xsl:otherwise>
        </xsl:choose>
    </span>
    <button class="xdDTButton" xd:xctname="DTPicker DTButton" xd:innerCtrl=" DTButton"
tabIndex="-1">
        <img src="res://infopath.exe/calendar.gif" />
    </button>
</div>
```

3.4.1.5 Drop-Down List Control

The following XSL examples are drop-down list controls, as specified in section 2.3.1.5.

The following example is a drop-down list control with the static values of "Select...", "1", "2", and "3".

```
<select class="xdComboBox xdBehavior_Select" title="" size="1" xd:binding="my:field1"
xd:boundProp="value" xd:xctname="dropdown" tabIndex="0" xd:CtrlId="CTRL1" style="WIDTH:
130px">
    <xsl:attribute name="value">
        <xsl:value-of select="my:field1" />
    </xsl:attribute>
    <option>
        <xsl:if test="my:field1=&quot;&quot;">
            <xsl:attribute name="selected">selected</xsl:attribute>
        </xsl:if>
        Select...
    </option>
    <option value="1">
        <xsl:if test="my:field1=&quot;1&quot;">
            <xsl:attribute name="selected">selected</xsl:attribute>
        </xsl:if>
        1
    </option>
    <option value="2">
        <xsl:if test="my:field1=&quot;2&quot;">
            <xsl:attribute name="selected">selected</xsl:attribute>
        </xsl:if>
        2
    </option>
    <option value="3">
        <xsl:if test="my:field1=&quot;3&quot;">
            <xsl:attribute name="selected">selected</xsl:attribute>
        </xsl:if>
        3
    </option>
</select>
```

The following example is a drop-down list control with values that are dynamically generated from an external data source called "sample".

```
<select class="xdComboBox xdBehavior Select" title="" style="WIDTH: 130px" size="1"
xd:binding="my:field2" xd:boundProp="value" value="" xd:xctname="dropdown" xd:CtrlId="CTRL2"
tabIndex="0">
    <xsl:attribute name="value">
        <xsl:value-of select="my:field2" />
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="function-available('xdXDocument:GetDOM')">
            <option />
            <xsl:variable name="val" select="my:field2" />
```

```
            <xsl:if
test="not(xdXDocument:GetDOM(&quot;sample&quot;)/main/small/big/name[.=$val] or $val='')">
                <option selected="selected">
                    <xsl:attribute name="value">
                        <xsl:value-of select="$val" />
                    </xsl:attribute>
                    <xsl:value-of select="$val" />
                </option>
            </xsl:if>
            <xsl:for-each
select="xdXDocument:GetDOM(&quot;sample&quot;)/main/small/big/name">
                <option>
                    <xsl:attribute name="value">
                        <xsl:value-of select="." />
                    </xsl:attribute>
                    <xsl:if test="$val=.">
                        <xsl:attribute name="selected">selected</xsl:attribute>
                    </xsl:if>
                    <xsl:value-of select="." />
                </option>
            </xsl:for-each>
        </xsl:when>
        <xsl:otherwise>
            <option>
                <xsl:value-of select="my:field2" />
            </option>
        </xsl:otherwise>
    </xsl:choose>
</select>
```

The following example is a drop-down list control with values that are dynamically generated from an external data source (2) called "sample", displaying only unique entries.

```
<select class="xdComboBox xdBehavior_Select" title="" style="WIDTH: 130px" size="1"
xd:binding="my:field2" xd:boundProp="value" value="" xd:xctname="dropdown" xd:CtrlId="CTRL2"
tabIndex="0">
    <xsl:attribute name="value">
        <xsl:value-of select="my:field2" />
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="function-available('xdXDocument:GetDOM')">
            <option />
            <xsl:variable name="val" select="my:field2" />
            <xsl:if
test="not(xdXDocument:GetDOM(&quot;sample&quot;)/main/small/big/name[.=$val] or $val='')">
                <option selected="selected">
                    <xsl:attribute name="value">
                        <xsl:value-of select="$val" />
                    </xsl:attribute>
                    <xsl:value-of select="$val" />
                </option>
            </xsl:if>
            <xsl:variable name="items">
                <xsl:copy-of
select="xdXDocument:GetDOM(&quot;sample&quot;)/main/small/big/name" />
            </xsl:variable>
            <xsl:variable name="uniqueItems" select="msxsl:node-set($items)/*[not(. =
preceding::name)]" />
            <xsl:for-each select="$uniqueItems">
                <option>
                    <xsl:attribute name="value">
                        <xsl:value-of select="." />
                    </xsl:attribute>
                    <xsl:if test="$val=.">
                        <xsl:attribute name="selected">selected</xsl:attribute>
                    </xsl:if>
                    <xsl:value-of select="." />
```

```
                </option>
            </xsl:for-each>
        </xsl:when>
        <xsl:otherwise>
            <option>
                <xsl:value-of select="my:field2" />
            </option>
        </xsl:otherwise>
    </xsl:choose>
</select>
```

The following example is a drop-down list control with conditional formatting and values that are dynamically generated from an external data source (2) called "sample", displaying only unique entries.

```
<select class="xdComboBox xdBehavior_Select" title="" style="WIDTH: 130px" size="1"
xd:binding="my:field2" xd:boundProp="value" value="" xd:xctname="dropdown" xd:CtrlId="CTRL2"
tabIndex="0">
    <xsl:attribute name="style">
        WIDTH: 130px;
        <xsl:choose>
            <xsl:when test="my:field2 = &quot;bob&quot;">FONT-WEIGHT: bold; COLOR: #808000;
FONT-STYLE: italic; BACKGROUND-COLOR: #800000; TEXT-DECORATION: underline line-
through</xsl:when>
            <xsl:when test="my:field2 = &quot;theodore&quot;" />
        </xsl:choose>
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="my:field2 = &quot;bob&quot;" />
        <xsl:when test="my:field2 = &quot;theodore&quot;">
            <xsl:attribute name="disabled">true</xsl:attribute>
        </xsl:when>
    </xsl:choose>
    <xsl:attribute name="value">
        <xsl:value-of select="my:field2" />
        </xsl:attribute>
            <xsl:choose>
        <xsl:when test="function-available('xdXDocument:GetDOM')">
            <option />
            <xsl:variable name="val" select="my:field2" />
            <xsl:if
test="not(xdXDocument:GetDOM(&quot;sample&quot;)/main/small/big/name[.=$val] or $val='')">
                <option selected="selected">
                    <xsl:attribute name="value">
                        <xsl:value-of select="$val" />
                    </xsl:attribute>
                    <xsl:value-of select="$val" />
                </option>
            </xsl:if>
            <xsl:variable name="items">
                <xsl:copy-of
select="xdXDocument:GetDOM(&quot;sample&quot;)/main/small/big/name" />
            </xsl:variable>
            <xsl:variable name="uniqueItems" select="msxsl:node-set($items)/*[not(. =
preceding::name)]" />
            <xsl:for-each select="$uniqueItems">
                <option>
                    <xsl:attribute name="value">
                        <xsl:value-of select="." />
                    </xsl:attribute>
                    <xsl:if test="$val=.">
                        <xsl:attribute name="selected">selected</xsl:attribute>
                    </xsl:if>
                    <xsl:value-of select="." />
                </option>
            </xsl:for-each>
```

```
        </xsl:when>
        <xsl:otherwise>
            <option>
                <xsl:value-of select="my:field2" />
            </option>
        </xsl:otherwise>
    </xsl:choose>
</select>
```

3.4.1.6 Expression Box Control

The following XSL examples are expression box controls, as specified in section 2.3.1.6.

The following example is an expression box control that is displaying the value of **my:field1**.

```
<span class="xdExpressionBox xdDataBindingUI" title="" xd:xctname="ExpressionBox" tabIndex="-
1" xd:CtrlId="CTRL3" xd:disableEditing="yes" style="WIDTH: 145px">
    <xsl:value-of select="my:field1" />
</span>
```

The following example is an expression box control with conditional formatting that is displaying the value of **my:field1**.**xd:datafmt** is equal to ""datetime","dateFormat:Short Date;timeFormat:none;"". This formats the value of **my:field1** to be a short date.

```
<span class="xdExpressionBox xdDataBindingUI xdBehavior Formatting" title="texas"
xd:binding="my:field1" xd:xctname="ExpressionBox" tabIndex="-1" xd:CtrlId="CTRL4"
xd:disableEditing="yes" xd:datafmt="&amp;quot;datetime&amp;quot;,&amp;quot;dateFormat:Short
Date;timeFormat:none;&amp;quot;" xd:num="">
    <xsl:attribute name="style">
        WIDTH: 145px;
        <xsl:choose>
            <xsl:when test="my:field1 = &amp;quot;1&amp;quot;">DISPLAY: none</xsl:when>
        </xsl:choose>
    </xsl:attribute>
    <xsl:attribute name="xd:num">
        <xsl:value-of select="my:field1" />
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="function-available('xdFormatting:formatString')">
            <xsl:value-of
select="xdFormatting:formatString(my:field1,&amp;quot;datetime&amp;quot;,&amp;quot;dateFormat:Short
Date;timeFormat:none;&amp;quot;)" />
        </xsl:when>
        <xsl:otherwise>
            <xsl:value-of select="my:field1" />
        </xsl:otherwise>
    </xsl:choose>
</span>
```

3.4.1.7 File Attachment Control

The following XSL example is a file attachment control, as specified in section 2.3.1.7.

```
<span class="xdFileAttachment" hideFocus="1" style="WIDTH: 161px; HEIGHT: 30px"
tabStop="true" xd:binding="my:field1" xd:boundProp="xd:inline" tabIndex="0"
xd:xctname="FileAttachment" xd:CtrlId="CTRL1">
    <xsl:if test="function-available('xdImage:getImageUrl')">
        <xsl:attribute name="src"><xsl:value-of
select="xdImage:getImageUrl(my:field1)"/></xsl:attribute>
    </xsl:if>
</span>
```

3.4.1.8 Hyperlink Control

The following XSL examples are hyperlink controls, as specified in section 2.3.1.8.

The following example is a static hyperlink control.

```
<a href="http://www.contoso.com" xd:disableEditing="yes">http://www.contoso.com</a>
```

The following example is a hyperlink control that dynamically changes its target, as well as its display text. The target of the hyperlink is the value of **my:field1** and the display text is the value of **my:field2**. This control also contains **border formatting** and shading formatting.

```
<span class="xdHyperlink" hideFocus="1" title="" style="BORDER-RIGHT: #cbd8eb 4.5pt dotted;
BORDER-TOP: #cbd8eb 4.5pt dotted; OVERFLOW: visible; BORDER-LEFT: #cbd8eb 4.5pt dotted;
WIDTH: 130px; BORDER-BOTTOM: #cbd8eb 4.5pt dotted; BACKGROUND-COLOR: #ffff00; TEXT-ALIGN:
left" xd:xctname="hyperlink">
    <a class="xdDataBindingUI" xd:CtrlId="CTRL5" xd:disableEditing="yes">
        <xsl:attribute name="href">
            <xsl:value-of select="my:field1" />
        </xsl:attribute>
        <xsl:value-of select="my:field2" />
    </a>
</span>
```

3.4.1.9 List Box Control

The following XSL examples are list box controls, as specified in section 2.3.1.9.

The following example is a list box control with three selection entries.

```
<select class="xdListBox xdBehavior Select" title="" size="3" xd:binding="my:field1"
xd:boundProp="value" tabIndex="0" xd:xctname="ListBox" xd:CtrlId="CTRL1" style="WIDTH:
130px">
    <xsl:attribute name="value">
        <xsl:value-of select="my:field1"/>
    </xsl:attribute>
    <option value="a">
        <xsl:if test="my:field1=&quot;a&quot;">
            <xsl:attribute name="selected">selected</xsl:attribute>
        </xsl:if>A</option>
    <option value="b">
        <xsl:if test="my:field1=&quot;b&quot;">
            <xsl:attribute name="selected">selected</xsl:attribute>
        </xsl:if>B</option>
    <option value="c">
        <xsl:if test="my:field1=&quot;c&quot;">
            <xsl:attribute name="selected">selected</xsl:attribute>
        </xsl:if>C</option>
</select>
```

The following example is a list box control that looks up the selection options from a repeating group within the main data source. The control only displays unique selection options. Conditional formatting is set such that if the value of **my:field1** is "a", the control has a red background color.

```
<select class="xdListBox xdBehavior Select" title="" style="WIDTH: 130px" size="3"
xd:binding="my:field1" xd:boundProp="value" value="a" xd:xctname="ListBox" xd:CtrlId="CTRL1"
tabIndex="0">
    <xsl:attribute name="style">WIDTH: 130px;<xsl:choose>
            <xsl:when test="my:field1 = &quot;a&quot;">BACKGROUND-COLOR: #ff0000</xsl:when>
        </xsl:choose>
    </xsl:attribute>
    <xsl:attribute name="value">
```

```
            <xsl:value-of select="my:field1"/>
        </xsl:attribute>
        <xsl:choose>
            <xsl:when test="function-available('xdXDocument:GetDOM')">
            <option/>
                <xsl:variable name="val" select="my:field1"/>
                    <xsl:if test="not(my:group1/my:group2[my:field2=$val] or $val='')">
                        <option selected="selected">
                            <xsl:attribute name="value">
                                <xsl:value-of select="$val"/>
                            </xsl:attribute>
                            <xsl:value-of select="$val"/>
                        </option>
                    </xsl:if>
                <xsl:variable name="items">
                    <xsl:copy-of select="my:group1/my:group2"/>
                </xsl:variable>
                <xsl:variable name="uniqueItems" select="msxsl:node-set($items)/*[not(my:field3 =
preceding::my:group2/my:field3)]"/>
                <xsl:for-each select="$uniqueItems">
                    <option>
                        <xsl:attribute name="value">
                            <xsl:value-of select="my:field2"/>
                        </xsl:attribute>
                        <xsl:if test="$val=my:field2">
                            <xsl:attribute name="selected">selected</xsl:attribute>
                        </xsl:if>
                        <xsl:value-of select="my:field3"/>
                    </option>
                </xsl:for-each>
            </xsl:when>
            <xsl:otherwise>
                <option>
                    <xsl:value-of select="my:field1"/>
                </option>
            </xsl:otherwise>
        </xsl:choose>
    </select>
```

The following example is a list box control that looks up the selection options from a repeating group in a secondary data source called "UserNameList". Conditional formatting is set such that if the value of **my:field1** is "a", the control has a red background color.

```
    <select class="xdListBox xdBehavior Select" title="" style="WIDTH: 130px" size="3"
    xd:binding="my:field1" xd:boundProp="value" value="a" xd:xctname="ListBox" xd:CtrlId="CTRL1"
    tabIndex="0">
        <xsl:attribute name="style">WIDTH: 130px;<xsl:choose>
                <xsl:when test="my:field1 = &amp;quot;a&amp;quot;">BACKGROUND-COLOR: #ff0000</xsl:when>
            </xsl:choose>
        </xsl:attribute>
        <xsl:attribute name="value">
            <xsl:value-of select="my:field1"/>
        </xsl:attribute>
        <xsl:choose>
            <xsl:when test="function-available('xdXDocument:GetDOM')">
                <option/>
                <xsl:variable name="val" select="my:field1"/>
                <xsl:if
    test="not(xdXDocument:GetDOM(&amp;quot;UserNameList&amp;quot;)/dfs:myFields/dfs:dataFields/dfs:UserNa
    meList[@E-mail Address=$val] or $val='')">
                    <option selected="selected">
                        <xsl:attribute name="value">
                            <xsl:value-of select="$val"/>
                        </xsl:attribute>
                        <xsl:value-of select="$val"/>
                    </option>
```

```
            </xsl:if>
            <xsl:for-each
select="xdXDocument:GetDOM(&quot;UserNameList&quot;)/dfs:myFields/dfs:dataFields/dfs:UserName
List">
                <option>
                    <xsl:attribute name="value">
                        <xsl:value-of select="@E-mail_Address"/>
                    </xsl:attribute>
                    <xsl:if test="$val=@E-mail_Address">
                        <xsl:attribute name="selected">selected</xsl:attribute>
                    </xsl:if>
                    <xsl:value-of select="@Last_Name"/>
                </option>
            </xsl:for-each>
        </xsl:when>
        <xsl:otherwise>
            <option>
                <xsl:value-of select="my:field1"/>
            </option>
        </xsl:otherwise>
    </xsl:choose>
</select>
```

3.4.1.10 Option Button Control

The following XSL examples are option button controls, as specified in section 2.3.1.10.

The following example is an option button control with three option buttons.

```
<div>
    <input class="xdBehavior_Boolean" title="" type="radio" name="{generate-id(my:field3)}"
xd:binding="my:field3" xd:boundProp="xd:value" xd:xctname="OptionButton" tabIndex="0"
xd:CtrlId="CTRL6" xd:onValue="1">
        <xsl:attribute name="xd:value">
            <xsl:value-of select="my:field3" />
        </xsl:attribute>
        <xsl:if test="my:field3=&quot;1&quot;">
            <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
        </xsl:if>
    </input>
    1
</div>
<div>
    <input class="xdBehavior_Boolean" title="" type="radio" name="{generate-id(my:field3)}"
xd:binding="my:field3" xd:boundProp="xd:value" xd:xctname="OptionButton" tabIndex="0"
xd:CtrlId="CTRL7" xd:onValue="2">
        <xsl:attribute name="xd:value">
            <xsl:value-of select="my:field3" />
        </xsl:attribute>
        <xsl:if test="my:field3=&quot;2quot;">
            <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
        </xsl:if>
    </input>
    2
</div>
<div>
    <input class="xdBehavior Boolean" title="" type="radio" name="{generate-id(my:field3)}"
xd:binding="my:field3" xd:boundProp="xd:value" xd:xctname="OptionButton" tabIndex="0"
xd:CtrlId="CTRL8" xd:onValue="3">
        <xsl:attribute name="xd:value">
            <xsl:value-of select="my:field3" />
        </xsl:attribute>
        <xsl:if test="my:field3=quot;3quot;">
            <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
        </xsl:if>
    </input>
```

```
    3
</div>
```

The following example is an option button control with conditional formatting. Conditional formatting is set such that if the value of **my:field3** is "2", the control is disabled.

```
<input class="xdBehavior_Boolean" title="" type="radio" name="{generate-id(my:field3)}"
xd:binding="my:field3" xd:boundProp="xd:value" xd:xctname="OptionButton" tabIndex="0"
xd:CtrlId="CTRL7" xd:onValue="2">
    <xsl:choose>
        <xsl:when test="my:field3 = 2">
            <xsl:attribute name="disabled">true</xsl:attribute>
        </xsl:when>
    </xsl:choose>
    <xsl:attribute name="xd:value">
        <xsl:value-of select="my:field3" />
    </xsl:attribute>
    <xsl:if test="my:field3=&amp;quot;2&amp;quot;">
        <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
    </xsl:if>
</input>
1
```

3.4.1.11 Repeating Section Control

The following XSL examples are **repeating section** controls, as specified in section 2.3.1.11.

The repeating section control contains a call to another section control. The repeating section call is surrounded by a **span** that shows how conditional formatting is set such that if the value of **my:field1** is "true", the control is disabled.

The following example is a repeating section call.

```
<span>
    <xsl:attribute name="style">
        <xsl:if test="my:field1 = string(true())">msos-xCollection-group2_1-
editing:disabled;</xsl:if>
    </xsl:attribute>
    <div><xsl:apply-templates select="my:group1/my:group2" mode="_1"/>
        <div class="optionalPlaceholder" xd:xmlToEdit="group2_1" tabIndex="0"
xd:action="xCollection::insert" align="left" style="WIDTH: 651px">Insert item</div>
    </div>
    <div> </div>
    <div> </div>
</span>
```

The following example is a repeating section body.

```
<xsl:template match="my:group2" mode="_1">
    <div class="xdRepeatingSection xdRepeating" title="" style="MARGIN-BOTTOM: 6px; WIDTH:
651px" align="left" xd:CtrlId="CTRL1" xd:xctname="RepeatingSection" tabIndex="-1">
        <div> </div>
        <div><xsl:apply-templates select="my:group3" mode="_2"/>
        </div>
        <div> </div>
    </div>
</xsl:template>
```

3.4.1.12 Repeating Table Control

The following XSL examples are repeating table controls, as specified in section 2.3.1.12.

The following example is a repeating table control that has three columns containing a text box control, as specified in section 2.3.1.16, inside each column (2). The repeating table also outputs a link with the text "Insert Item" that adds an additional row to the repeating table after clicking this link.

```
<div>
    <table class="xdRepeatingTable msoUcTable" title="" style="TABLE-LAYOUT: fixed; WIDTH:
651px; BORDER-TOP-STYLE: none; BORDER-RIGHT-STYLE: none; BORDER-LEFT-STYLE: none; BORDER-
COLLAPSE: collapse; WORD-WRAP: break-word; BORDER-BOTTOM-STYLE: none" border="1"
xd:CtrlId="CTRL12">
        <colgroup>
            <col style="WIDTH: 210px" />
            <col style="WIDTH: 211px" />
            <col style="WIDTH: 230px" />
        </colgroup>
        <tbody class="xdTableHeader">
            <tr>
                <td>
                    <div>
                        <strong />
                    </div>
                </td>
                <td>
                    <div>
                        <strong />
                    </div>
                </td>
                <td>
                    <div>
                        <strong />
                    </div>
                </td>
            </tr>
        </tbody>
        <tbody xd:xctname="RepeatingTable">
            <xsl:for-each select="my:group1/my:group2">
                <tr>
                    <td>
                        <span class="xdTextBox" hideFocus="1" title="" xd:binding="my:field5"
xd:xctname="PlainText" tabIndex="0" xd:CtrlId="CTRL13" style="WIDTH: 100%">
                            <xsl:value-of select="my:field5" />
                        </span>
                    </td>
                    <td>
                        <span class="xdTextBox" hideFocus="1" title="" xd:binding="my:field6"
xd:xctname="PlainText" tabIndex="0" xd:CtrlId="CTRL14" style="WIDTH: 100%">
                            <xsl:value-of select="my:field6" />
                        </span>
                    </td>
                    <td>
                        <span class="xdTextBox" hideFocus="1" title="" xd:binding="my:field7"
xd:xctname="PlainText" tabIndex="0" xd:CtrlId="CTRL15" style="WIDTH: 100%">
                            <xsl:value-of select="my:field7" />
                        </span>
                    </td>
                </tr>
            </xsl:for-each>
        </tbody>
</table>
<div class="optionalPlaceholder" xd:xmlToEdit="group2_8" tabIndex="0"
xd:action="xCollection::insert" style="WIDTH: 651px">Insert item</div>
</div>
```

The following example is a repeating table control that has three columns containing a text box control inside each column. This repeating table also contains a footer. Conditional formatting is set such that if the value of **my:field8** is "2", the control has a different background color. This control also causes a postback whenever a table row is inserted or removed.

```
<div>
    <table class="xdRepeatingTable msoUcTable" title="" style="TABLE-LAYOUT: fixed; WIDTH:
651px; BORDER-TOP-STYLE: none; BORDER-RIGHT-STYLE: none; BORDER-LEFT-STYLE: none; BORDER-
COLLAPSE: collapse; WORD-WRAP: break-word; BORDER-BOTTOM-STYLE: none" border="1"
xd:CtrlId="CTRL16" xd:postbackModel="always">
        <colgroup>
            <col style="WIDTH: 210px" />
            <col style="WIDTH: 211px" />
            <col style="WIDTH: 230px" />
        </colgroup>
        <tbody class="xdTableHeader">
            <tr>
                <td>
                    <div>
                        <strong />
                    </div>
                </td>
                <td>
                    <div>
                        <strong />
                    </div>
                </td>
                <td>
                    <div>
                        <strong />
                    </div>
                </td>
            </tr>
        </tbody>
        <tbody xd:xctname="RepeatingTable">
            <xsl:for-each select="my:group3/my:group4">
                <xsl:if test="not((my:field8 = quot;1quot;))">
                    <tr>
                        <xsl:attribute name="style">
                            <xsl:choose>
                                <xsl:when test="my:field8 = 2">BACKGROUND-COLOR:
#ff00ff</xsl:when>
                            </xsl:choose>
                        </xsl:attribute>
                        <td>
                            <span class="xdTextBox" hideFocus="1" title=""
xd:binding="my:field8" xd:xctname="PlainText" tabIndex="0" xd:CtrlId="CTRL17" style="WIDTH:
100%">
                                <xsl:value-of select="my:field8" />
                            </span>
                        </td>
                        <td>
                            <span class="xdTextBox" hideFocus="1" title=""
xd:binding="my:field9" xd:xctname="PlainText" tabIndex="0" xd:CtrlId="CTRL18" style="WIDTH:
100%">
                                <xsl:value-of select="my:field9" />
                            </span>
                        </td>
                        <td>
                            <span class="xdTextBox" hideFocus="1" title=""
xd:binding="my:field10" xd:xctname="PlainText" tabIndex="0" xd:CtrlId="CTRL19" style="WIDTH:
100%">
                                <xsl:value-of select="my:field10" />
                            </span>
                        </td>
                    </tr>
                </xsl:if>
            </xsl:for-each>
        </tbody>
```

```
        <tbody class="xdTableFooter">
            <tr>
                <td>
                    <div> </div>
                </td>
                <td>
                    <div> </div>
                </td>
                <td>
                    <div> </div>
                </td>
            </tr>
        </tbody>
    </table>
</div>
```

The following example is a repeating table control that has one column with a text box inside. Conditional formatting is set such that if the value of **my:field1** is "1", the control does not allow the user to insert or delete rows from the table. Note that this conditional formatting is placed outside the repeating table element.

```
<span>
    <xsl:attribute name="style">
        <xsl:if test="my:field1 = &quot;1&quot;">msos-xCollection-group8 16-
editing:disabled;</xsl:if>
    </xsl:attribute>
    <div>
        <table class="xdRepeatingTable msoUcTable" title="" style="TABLE-LAYOUT: fixed;
WIDTH: 651px; BORDER-TOP-STYLE: none; BORDER-RIGHT-STYLE: none; BORDER-LEFT-STYLE: none;
BORDER-COLLAPSE: collapse; WORD-WRAP: break-word; BORDER-BOTTOM-STYLE: none" border="1"
xd:CtrlId="CTRL24">
            <colgroup>
                <col style="WIDTH: 651px" />
            </colgroup>
        <tbody class="xdTableHeader">
            <tr>
                <td>
                    <div>
                        <strong />
                    </div>
                </td>
            </tr>
        </tbody>
        <tbody xd:xctname="RepeatingTable">
            <xsl:for-each select="my:group7/my:group8">
                <tr>
                    <td>
                        <span class="xdTextBox" hideFocus="1" title=""
xd:binding="my:field14" xd:xctname="PlainText" tabIndex="0" xd:CtrlId="CTRL25" style="WIDTH:
100%">
                            <xsl:value-of select="my:field14" />
                        </span>
                    </td>
                </tr>
            </xsl:for-each>
        </tbody>
    </table>
    <div class="optionalPlaceholder" xd:xmlToEdit="group8_16" tabIndex="0"
xd:action="xCollection::insert" style="WIDTH: 651px">Insert item</div>
    </div>
</span>
```

3.4.1.13 Rich Text Box Control

The following XSL example is a rich text box control, as specified in section 2.3.1.13, with conditional formatting set such that if the value of **my:field2** is "false", the control is disabled.

```
<span class="xdRichTextBox" hideFocus="1" title="" xd:binding="my:field1" tabIndex="0"
xd:xctname="RichText" xd:CtrlId="CTRL1" style="WIDTH: 651px; HEIGHT: 50px">
    <xsl:choose>
        <xsl:when test="my:field2 = string(true())">
            <xsl:attribute name="contentEditable">false</xsl:attribute>
        </xsl:when>
    </xsl:choose>
    <xsl:copy-of select="my:field1/node()"/>
</span>
```

3.4.1.14 Section Control and Optional Section Control

The following XSL examples are **section and optional section** controls, as specified in section 2.4.1.18.

The following example is a section control that can be digitally signed. This control contains a text box control, as specified in section 2.3.1.16.

The section call is as follows:

```
<xsl:apply-templates select="my:group1" mode="_1"/>
```

The section body is as follows:

```
<xsl:template match="my:group1" mode="_1">
    <div class="xdSection xdRepeating" title="" style="MARGIN-BOTTOM: 6px; WIDTH: 651px"
align="left" xd:xctname="Section" xd:CtrlId="CTRL1" xd:SignedSectionName="group1" tabIndex="-
1">
        <div> </div>
        <div><span class="xdTextBox" hideFocus="1" title="" xd:xctname="PlainText"
xd:CtrlId="CTRL2" tabIndex="0" xd:binding="my:field1" style="WIDTH: 130px">
            <xsl:value-of select="my:field1"/>
        </span>
        </div>
        <div> </div>
    </div>
    <div xd:disableEditing="yes" xd:SignatureBlock="group1"
xd:SignedSectionDisplaySignatures="true" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 6px;
BEHAVIOR: url (#default#SignaturesInDocUI); WIDTH: 651px">
        <xsl:if test="function-available('xdXDocument:GetNamedNodeProperty')">
            <xsl:if
test="xdXDocument:GetNamedNodeProperty(/my:myFields/my:signatures1/my:signatures2,
'CanAddSignature', 'false') = 'true'">
                <button title="" style="width: 100%; height: 100%; text-align: left; border:
0px solid; padding: 2px; background-color: window; cursor: hand;">
                    <table style="color: windowtext;" class="defaultInDocUI">
                        <tbody>
                            <tr>
                                <td><img src="res://infopath.exe/AddSignatureButton.gif"
width="12" height="12"/></td>
                                <td>Click here to sign this section</td>
                            </tr>
                        </tbody>
                    </table>
                </button>
            </xsl:if>
            <xsl:for-each select="/my:myFields/my:signatures1/my:signatures2">
                <xsl:for-each select="sig:Signature">
                    <xsl:choose>
```

```
                        <xsl:when test="xdXDocument:GetNamedNodeProperty(.,
'IsValidSignature', 'false') = 'true'">
                            <button title="" style="width: 100%; height: 100%; text-align:
left; border: 0px solid; padding: 2px; background-color: window; cursor: hand;">
                                <table style="color: windowtext;" class="defaultInDocUI">
                                    <tbody>
                                        <tr>
                                            <xsl:choose>
                                                <xsl:when test="function-
available('xdImage:getImageUrl') and
sig:Object/sig:SignatureProperties/sig:SignatureProperty/xdSignatureProperties:NonRepudiation
/xdSignatureProperties:ValidSignedImage">
                                                    <td style="display: none;"><img
src="res://infopath.exe/signaturevalid.gif"/></td>
                                                    <td><img
src="{xdImage:getImageUrl(sig:Object/sig:SignatureProperties/sig:SignatureProperty/xdSignatur
eProperties:NonRepudiation/xdSignatureProperties:ValidSignedImage)}"/> </td>
                                                </xsl:when>
                                                <xsl:otherwise>
                                                    <td><img
src="res://infopath.exe/signaturevalid.gif" width="12" height="12"/></td>
                                                    <td style="color: gray;">
                                                        <div><b><xsl:value-of
select="xdXDocument:GetNamedNodeProperty(., 'SignedBy', '???')"/></b><span style="margin: 0pt
20pt">View details</span></div>
                                                        <div><xsl:value-of
select="xdXDocument:GetNamedNodeProperty(., 'SignedOn', '???')"/></div>
                                                    </td>
                                                </xsl:otherwise>
                                            </xsl:choose>
                                        </tr>
                                    </tbody>
                                </table>
                            </button>
                        </xsl:when>
                        <xsl:otherwise>
                            <button title="" style="width: 100%; height: 100%; text-align:
left; border: 0px solid; padding: 2px; background-color: window; cursor: hand;">
                                <table style="font: message-box; color: windowtext;">
                                    <tbody>
                                        <tr>
                                            <xsl:choose>
                                                <xsl:when test="function-
available('xdImage:getImageUrl') and
sig:Object/sig:SignatureProperties/sig:SignatureProperty/xdSignatureProperties:NonRepudiation
/xdSignatureProperties:InvalidSignedImage">
                                                    <td style="display: none;"><img
src="res://infopath.exe/signaturevalid.gif"/></td>
                                                    <td><img
src="{xdImage:getImageUrl(sig:Object/sig:SignatureProperties/sig:SignatureProperty/xdSignatur
eProperties:NonRepudiation/xdSignatureProperties:InvalidSignedImage)}"/> </td>
                                                </xsl:when>
                                                <xsl:otherwise>
                                                    <td><img
src="res://infopath.exe/signatureerror.gif" width="12" height="12"/></td>
                                                    <td style="color: red;"><b>There is a
problem with this signature</b><span style="margin: 0pt 20pt">View details</span></td>
                                                </xsl:otherwise>
                                            </xsl:choose>
                                        </tr>
                                    </tbody>
                                </table>
                            </button>
                        </xsl:otherwise>
                    </xsl:choose>
                </xsl:for-each>
            </xsl:for-each>
        </xsl:if>
    </div>
```

```
</xsl:template>
```

The following example is an optional section control. This control contains a date picker control, as specified in section 2.3.1.4.

The optional section call is as follows:

```
<xsl:choose>
    <xsl:when test="my:group1">
        <xsl:apply-templates select="my:group1" mode="_1"/>
    </xsl:when>
    <xsl:otherwise>
        <div class="optionalPlaceholder" xd:xmlToEdit="group1_1" tabIndex="0" align="left"
style="WIDTH: 651px">Click here to insert</div>
    </xsl:otherwise>
</xsl:choose>
```

The optional section body is as follows:

```
<xsl:template match="my:group1" mode="_1">
    <div class="xdSection xdRepeating" title="" style="MARGIN-BOTTOM: 6px; WIDTH: 651px"
align="left" xd:xctname="Section" xd:CtrlId="CTRL1" tabIndex="-1">
        <div> </div>
        <div>
            <div class="xdDTPicker" title="" style="WIDTH: 130px" noWrap="1"
xd:xctname="DTPicker" xd:CtrlId="CTRL3"><span class="xdDTText xdBehavior_FormattingNoBUI"
hideFocus="1" contentEditable="true" xd:xctname="DTPicker DTText" tabIndex="0"
xd:binding="my:field2" xd:datafmt="&amp;quot;date&amp;quot;,&amp;quot;dateFormat:Short Date;&amp;quot;"
xd:boundProp="xd:num" xd:innerCtrl=" DTText">
                    <xsl:attribute name="xd:num">
                        <xsl:value-of select="my:field2"/>
                    </xsl:attribute>
                    <xsl:choose>
                        <xsl:when test="function-available('xdFormatting:formatString')">
                            <xsl:value-of
select="xdFormatting:formatString(my:field2,&amp;quot;date&amp;quot;,&amp;quot;dateFormat:Short
Date;&amp;quot;)"/>
                        </xsl:when>
                        <xsl:otherwise>
                            <xsl:value-of select="my:field2"/>
                        </xsl:otherwise>
                    </xsl:choose>
                </span>
                <button class="xdDTButton" xd:xctname="DTPicker DTButton"
xd:innerCtrl="_DTButton" tabIndex="-1">
                    <img src="res://infopath.exe/calendar.gif"/>
                </button>
            </div>
        </div>
        <div> </div>
    </div>
</xsl:template>
```

3.4.1.15 Table Control

The following XSL example is a table control, as specified in section 2.3.1.15, that is two rows by two columns that has the value "1" in three of the four cells and a button control, as specified in section 2.3.1.1, in the remaining cell.

```
<table class="xdLayout" style="BORDER-RIGHT: medium none; TABLE-LAYOUT: fixed; BORDER-TOP:
medium none; BORDER-LEFT: medium none; WIDTH: 260px; BORDER-BOTTOM: medium none; BORDER-
COLLAPSE: collapse; WORD-WRAP: break-word" borderColor="buttontext" border="1">
    <colgroup>
        <col style="WIDTH: 130px" />
        <col style="WIDTH: 130px" />
    </colgroup>
    <tbody vAlign="top">
        <tr>
            <td>
                <div>
                    <font face="Verdana" size="2">1</font>
                </div>
            </td>
            <td>
                <div>
                    <font face="Verdana" size="2">
                        <input class="langFont" title="" type="button" value="Button"
xd:xctname="Button" xd:CtrlId="CTRL10 5" tabIndex="0" />
                    </font>
                </div>
            </td>
        </tr>
        <tr>
            <td>
                <div>
                    <font face="Verdana" size="2">1</font>
                </div>
            </td>
            <td>
                <div>
                    <font face="Verdana" size="2">1</font>
                </div>
            </td>
        </tr>
    </tbody>
</table>
```

3.4.1.16 Text Box Control

The following XSL examples are text box controls, as specified in section 2.3.1.16.

The following example is a text box control that is bound to **my:field1**.

```
<div>
  <span class="xdTextBox" hideFocus="1" title="" xd:binding="my:field1" tabIndex="0"
xd:xctname="PlainText" xd:CtrlId="CTRL1" style="WIDTH: 130px">
    <xsl:value-of select="my:field1"/>
  </span>
</div>
```

The following example is a text box control with **multi-line** enabled that is bound to **my:field2**.

```
<div>
 <span class="xdTextBox" hideFocus="1" title="" xd:binding="my:field2" tabIndex="0"
xd:xctname="PlainText" xd:CtrlId="CTRL2"
xd:datafmt="&quot;string&quot;,&quot;plainMultiline&quot;" style="OVERFLOW-Y: auto; OVERFLOW-
X: auto; WIDTH: 130px; WHITE-SPACE: normal; WORD-WRAP: break-word">
    <xsl:choose>
      <xsl:when test="function-available('xdFormatting:formatString')">
        <xsl:value-of
select="xdFormatting:formatString(my:field2,&quot;string&quot;,&quot;plainMultiline&quot;)"
disable-output-escaping="yes"/>
      </xsl:when>
```

```
      <xsl:otherwise>
        <xsl:value-of select="my:field2" disable-output-escaping="yes"/>
      </xsl:otherwise>
    </xsl:choose>
  </span>
</div>
```

The following example is a text box control that is bound to **my:field3**. The value of **xd:datafmt** is ""date","locale:1061; dateFormat:d.MM.yyyy;"". This formats the value of **my:field3** to be a date and specifies the locale as "Estonian". The specific date format is "dateFormat:d.MM.yyyy". This formats the date and specifies how the day, month, and year are displayed. For this example, the date could be displayed as 14.03.2001 corresponding to March 14th, 2001.

```
<div>
  <span class="xdTextBox xdBehavior_Formatting" hideFocus="1" title="" contentEditable="true"
xd:binding="my:field3" tabIndex="0" xd:xctname="PlainText" xd:CtrlId="CTRL3"
xd:datafmt="&amp;quot;date&amp;quot;,&amp;quot;locale:1061; dateFormat:d.MM.yyyy;&amp;quot;"
xd:boundProp="xd:num" style="WIDTH: 130px">
    <xsl:attribute name="xd:num">
      <xsl:value-of select="my:field3"/>
    </xsl:attribute>
    <xsl:choose>
      <xsl:when test="function-available('xdFormatting:formatString')">
        <xsl:value-of
select="xdFormatting:formatString(my:field3,&amp;quot;date&amp;quot;,&amp;quot;locale:1061;dateFormat:d.
MMMM yyyy'. a.';&amp;quot;)"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="my:field3"/>
      </xsl:otherwise>
    </xsl:choose>
  </span>
</div>
```

The following example is a text box control that has conditional formatting and is bound to **my:field4**. Conditional formatting is set such that if the value of **my:field4** is "abc", the text in the control is bold.

```
<div>
  <span class="xdTextBox" hideFocus="1" title="" xd:binding="my:field4" tabIndex="0"
xd:xctname="PlainText" xd:CtrlId="CTRL4">
    <xsl:attribute name="style">WIDTH: 130px;
      <xsl:choose>
        <xsl:when test="my:field4 = &amp;quot;abc&amp;quot;">FONT-WEIGHT: bold</xsl:when>
      </xsl:choose>
    </xsl:attribute>
    <xsl:value-of select="my:field4"/>
  </span>
</div>
```

The following example is a text box control that has conditional formatting and data formatting and is bound to **my:field5**. Conditional formatting is set such that if the value of **my:field5** is "def", the text in the control is underlined. The value of **xd:datafmt** is ""time","locale:1033;timeFormat:hh:mm:ss tt;"". This formats the value of **my:field5** to be a time and specifies the locale as "English". The specific time format is "timeFormat:hh:mm:ss tt;". This formats the time and specifies how the hour, minutes, and seconds are displayed. For this example, the time could be displayed as 09:46:55 AM.

```
<div>
```

```
  <span class="xdTextBox xdBehavior_Formatting" hideFocus="1" title="" contentEditable="true"
xd:binding="my:field5" tabIndex="0" xd:xctname="PlainText" xd:CtrlId="CTRL5"
xd:datafmt="&quot;time&quot;,&quot;locale:1033;timeFormat:hh:mm:ss tt;&quot;"
xd:boundProp="xd:num">
    <xsl:attribute name="style">WIDTH: 130px;
      <xsl:choose>
        <xsl:when test="my:field5 = &quot;def&quot;">TEXT-DECORATION: underline</xsl:when>
      </xsl:choose>
    </xsl:attribute>
    <xsl:attribute name="xd:num">
      <xsl:value-of select="my:field5"/>
    </xsl:attribute>
    <xsl:choose>
      <xsl:when test="function-available('xdFormatting:formatString')">
        <xsl:value-of
select="xdFormatting:formatString(my:field5,&quot;time&quot;,&quot;locale:1033;timeFormat:hh:
mm:ss tt;&quot;)"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="my:field5"/>
      </xsl:otherwise>
    </xsl:choose>
  </span>
</div>
```

3.4.2 Control-Specific Attributes

xd:action

The following XSLT fragment is an example of a button control, as specified in section 2.3.1.1, with a submit action, as specified in section 2.2.147.12.

```
<input class="langFont" title="" style="BEHAVIOR: url(#default#ActionButton)" type="button"
value="Submit" xd:xctname="Button" xd:CtrlId="CTRL3_5" xd:action="submit" tabIndex="0"/>
```

The following XSLT fragment is an example of a repeating section control, as specified in section 2.4.1.15, that allows insertion of sections.

```
<xsl:apply-templates select="my:group3/my:group4" mode="_2"/>
<div class="optionalPlaceholder" xd:xmlToEdit="group4 3" tabIndex="0"
xd:action="xCollection::insert" align="left" style="WIDTH: 651px">Insert item</div>
...
```

The following XSLT fragment is an example of a text box control, as specified in section 2.4.1.20, with the **autoAdvance** attribute, as specified in section 2.4.2.3, set to "yes".

```
<span class="xdTextBox" hideFocus="1" title="" contentEditable="true" xd:binding="my:field3"
tabIndex="0" xd:xctname="PlainText" xd:CtrlId="CTRL8" xd:autoAdvance="yes" style="WIDTH:
130px; WHITE-SPACE: nowrap">
  <xsl:value-of select="my:field3"/>
</span>
```

The following XSLT fragment is an example of a button control with a "refresh" **action**, as specified in section 2.2.147.12, with the **auxDom** attribute, as specified in section 2.4.2.4.

```
<input class="langFont" title="" style="BEHAVIOR: url(#default#ActionButton)" type="button"
value="Refresh" xd:xctname="Button" xd:CtrlId="CTRL7_5" xd:action="refresh"
xd:auxDom="Notification List" tabIndex="0"/>
```

The following XSLT is an example of a text box control bound to an XML field with the **binding** attribute, as specified in section 2.4.2.6.

```
<span class="xdTextBox" hideFocus="1" title="" xd:binding="my:field6" tabIndex="0"
xd:xctname="PlainText" xd:CtrlId="CTRL10" style="WIDTH: 130px">
  <xsl:value-of select="my:field6"/>
</span>
```

The following XSLT fragment is an example of a custom control with a **bindingProperty** attribute, as specified in section 2.4.2.7, set to "Value".

```
<object class="xdActiveX" hideFocus="1" style="WIDTH: 288px; HEIGHT: 192px"
classid="clsid:8e27c92b-1264-101c-8a2f-040224009c02" tabIndex="0" tabStop="true"
xd:xctname="{{8e27c92b-1264-101c-8a2f-040224009c02}}" xd:CtrlId="CTRL9" xd:bindingType="text"
xd:bindingProperty="Value" xd:boundProp="xd:inline" contentEditable="false"
xd:binding="my:field5">
...
</object>
```

The following XSLT fragment is an example of a custom control with a **bindingType** attribute, as specified in section 2.4.2.8, set to "text".

```
<object class="xdActiveX" hideFocus="1" style="WIDTH: 288px; HEIGHT: 192px"
classid="clsid:8e27c92b-1264-101c-8a2f-040224009c02" tabIndex="0" tabStop="true"
xd:xctname="{{8e27c92b-1264-101c-8a2f-040224009c02}}" xd:CtrlId="CTRL9" xd:bindingType="text"
xd:bindingProperty="Value" xd:boundProp="xd:inline" contentEditable="false"
xd:binding="my:field5">
...
</object>
```

The following XSLT fragment is an example of a text box control with a **boundProp** attribute, as specified in section 2.4.2.9, set to "xd:num".

```
<span class="xdTextBox xdBehavior_Formatting" hideFocus="1" title="" contentEditable="true"
xd:binding="my:field12" tabIndex="0" xd:xctname="PlainText" xd:CtrlId="CTRL17"
xd:boundProp="xd:num"
xd:datafmt="&amp;quot;currency&amp;quot;,&amp;quot;numDigits:0;negativeOrder:0;positiveOrder:0;currencyLo
cale:1033;&amp;quot;" style="WIDTH: 130px">
    <xsl:attribute name="xd:num">
        <xsl:value-of select="my:field12"/>
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="function-available('xdFormatting:formatString')">
            <xsl:value-of
select="xdFormatting:formatString(my:field12,&amp;quot;currency&amp;quot;,&amp;quot;numDigits:0;negativeO
rder:0;positiveOrder:0;currencyLocale:1033;&amp;quot;)"/>
        </xsl:when>
        <xsl:otherwise>
            <xsl:value-of select="my:field12"/>
        </xsl:otherwise>
    </xsl:choose>
</span>
```

The following XSLT fragment is an example of a text box control with a **CtrlId** attribute, as specified in section 2.4.2.10, set to "CTRL1":

```
<span class="xdTextBox" hideFocus="1" title="" xd:binding="my:field1" tabIndex="0"
xd:xctname="PlainText" xd:CtrlId="CTRL1" style="WIDTH: 130px">
    <xsl:value-of select="my:field1"/>
```

```
</span>
```

The following XSLT fragment is an example of a text box control with currency data formatting specified using a **datafmt** attribute, as specified in section 2.4.2.11.

```
<span class="xdTextBox xdBehavior_Formatting" hideFocus="1" title="" contentEditable="true"
xd:binding="my:field7" tabIndex="0" xd:xctname="PlainText" xd:CtrlId="CTRL11"
xd:boundProp="xd:num"
xd:datafmt="&quot;currency&quot;,&quot;numDigits:0;negativeOrder:0;positiveOrder:0;currencyLo
cale:1033;&quot;" style="WIDTH: 130px">
...
</span>
```

The following XSLT fragment is an example of a hyperlink control, as specified in section 2.4.1.12, with the **disableEditing** attribute, as specified in section 2.4.2.12, set to "yes".

```
<span class="xdHyperlink" hideFocus="1" title="" style="OVERFLOW: visible; WIDTH: 130px;
TEXT-ALIGN: left" xd:xctname="hyperlink">
    <a class="xdDataBindingUI" xd:CtrlId="CTRL13" xd:disableEditing="yes">
        <xsl:attribute name="href">
            <xsl:value-of select="my:field1"/>
        </xsl:attribute>
        <xsl:value-of select="my:field1"/>
    </a>
</span>
```

The following XSLT fragment is an example of a text box control with the **disableEditing** attribute, as specified in section 2.4.2.12, set to "yes":

```
<span class="xdTextBox" hideFocus="1" title="" xd:binding="my:field8" tabIndex="-1"
xd:xctname="PlainText" xd:CtrlId="CTRL15" xd:disableEditing="yes" style="WIDTH: 130px; WHITE-
SPACE: nowrap">
    <xsl:value-of select="my:field8"/>
</span>
```

The following XSLT fragment is an example of a custom control with the **enabledProperty** attribute, as specified in section 2.4.2.13, set to "Enabled".

```
<object
class="xdActiveX"
hideFocus="1"
style="WIDTH: 288px; HEIGHT: 192px"
classid="clsid:8e27c92b-1264-101c-8a2f-040224009c02"
tabIndex="0"
tabStop="true"
xd:xctname="{{8e27c92b-1264-101c-8a2f-040224009c02}}"
xd:CtrlId="CTRL2"
xd:bindingType="text"
xd:bindingProperty="Value"
xd:boundProp="xd:inline"
xd:enabledValue="true"
xd:enabledProperty="Enabled"
contentEditable="false"
xd:binding="my:field1">
    <xsl:if test="function-available('xdImage:getImageUrl')">
        <xsl:attribute name="src">
           <xsl:value-of select="xdImage:getImageUrl(my:field1)"/>
        </xsl:attribute>
    </xsl:if>
```

```
</object>
```

The following XSLT fragment is an example of a custom control with the **enabledValue** attribute, as specified in section 2.4.2.14, set to "true".

```
<object
class="xdActiveX"
hideFocus="1"
style="WIDTH: 288px; HEIGHT: 192px"
classid="clsid:8e27c92b-1264-101c-8a2f-040224009c02"
tabIndex="0"
tabStop="true"
xd:xctname="{{8e27c92b-1264-101c-8a2f-040224009c02}}"
xd:CtrlId="CTRL2"
xd:bindingType="text"
xd:bindingProperty="Value"
xd:boundProp="xd:inline"
xd:enabledValue="true"
xd:enabledProperty="Enabled"
contentEditable="false"
xd:binding="my:field1">
    <xsl:if test="function-available('xdImage:getImageUrl')">
        <xsl:attribute name="src">
            <xsl:value-of select="xdImage:getImageUrl(my:field1)"/>
        </xsl:attribute>
    </xsl:if>
</object>
```

The following XSLT is an example of a date picker control, as specified in section 2.4.1.8, with an **innerCtrl** attribute, as specified in section 2.4.2.19.

```
<div class="xdDTPicker" title="" style="WIDTH: 130px" noWrap="1" xd:xctname="DTPicker"
xd:CtrlId="CTRL15">
    <span class="xdDTText xdBehavior FormattingNoBUI" hideFocus="1" contentEditable="true"
xd:binding="my:field10" tabIndex="0" xd:xctname="DTPicker_DTText" xd:boundProp="xd:num"
xd:datafmt="&amp;quot;date&amp;quot;,&amp;quot;dateFormat:Short Date;&amp;quot;" xd:innerCtrl="_DTText">
        ...
    </span>
    <button class="xdDTButton" xd:xctname="DTPicker_DTButton" xd:innerCtrl="_DTButton"
tabIndex="-1">
        <img src="res://infopath.exe/calendar.gif"/>
    </button>
</div>
```

The following XSLT fragment is an example of a text box control with a **num** attribute, as specified in section 2.4.2.26.

```
<span class="xdTextBox xdBehavior_Formatting" hideFocus="1" title="" contentEditable="true"
xd:binding="my:field12" tabIndex="0" xd:xctname="PlainText" xd:CtrlId="CTRL17"
xd:boundProp="xd:num"
xd:datafmt="&amp;quot;currency&amp;quot;,&amp;quot;numDigits:0;negativeOrder:0;positiveOrder:0;currencyLo
cale:1033;&amp;quot;" style="WIDTH: 130px">
    <xsl:attribute name="xd:num">
        <xsl:value-of select="my:field12"/>
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="function-available('xdFormatting:formatString')">
            <xsl:value-of
select="xdFormatting:formatString(my:field12,&amp;quot;currency&amp;quot;,&amp;quot;numDigits:0;negativeO
rder:0;positiveOrder:0;currencyLocale:1033;&amp;quot;)"/>
        </xsl:when>
        <xsl:otherwise>
```

```
            <xsl:value-of select="my:field12"/>
        </xsl:otherwise>
    </xsl:choose>
</span>
```

The following XSLT fragment is an example of a check box control with the **offValue** attribute, as specified in section 2.4.2.27, set to "1".

```
<input class="xdBehavior_Boolean" title="" type="checkbox" xd:binding="my:field4"
tabIndex="0" xd:xctname="CheckBox" xd:CtrlId="CTRL9" xd:boundProp="xd:value" xd:offValue="1"
xd:onValue="true">
    <xsl:attribute name="xd:value">
        <xsl:value-of select="my:field4"/>
    </xsl:attribute>
    <xsl:if test="my:field4=&quot;true&quot;">
        <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
    </xsl:if>
</input> Field 4
```

The following XSLT fragment is an example of a check box control with the **onValue** attribute, as specified in section 2.4.2.28, set to "true".

```
<input class="xdBehavior_Boolean" title="" type="checkbox" xd:binding="my:field4"
tabIndex="0" xd:xctname="CheckBox" xd:CtrlId="CTRL9" xd:boundProp="xd:value" xd:offValue="1"
xd:onValue="true">
    <xsl:attribute name="xd:value">
        <xsl:value-of select="my:field4"/>
    </xsl:attribute>
    <xsl:if test="my:field4=&quot;true&quot;">
        <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
    </xsl:if>
</input> Field 4
```

The following XSLT fragment is an example of an option button control with two options, with the **onValue** attribute set to "value1" and "value2" respectively.

```
<input class="xdBehavior_Boolean" title="" type="radio" name="{generate-id(my:field5)}"
xd:binding="my:field5" tabIndex="0" xd:xctname="OptionButton" xd:CtrlId="CTRL10"
xd:boundProp="xd:value" xd:onValue="value1">
    <xsl:attribute name="xd:value">
        <xsl:value-of select="my:field5"/>
    </xsl:attribute>
    <xsl:if test="my:field5=&quot;value1&quot;">
        <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
    </xsl:if>
</input> Field 5
</div>
<div>
<input class="xdBehavior_Boolean" title="" type="radio" name="{generate-id(my:field5)}"
xd:binding="my:field5" tabIndex="0" xd:xctname="OptionButton" xd:CtrlId="CTRL11"
xd:boundProp="xd:value" xd:onValue="value2">
    <xsl:attribute name="xd:value">
        <xsl:value-of select="my:field5"/>
    </xsl:attribute>
    <xsl:if test="my:field5=&quot;value2&quot;">
        <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
    </xsl:if>
</input> Field 5
```

The following XSLT fragment is an example of a text box control with the **postbackModel** attribute, as specified in section 2.4.2.29, set to "always".

```
<span class="xdTextBox" hideFocus="1" title="" xd:binding="my:field1" tabIndex="0"
xd:xctname="PlainText" xd:CtrlId="CTRL1" xd:postbackModel="always" style="WIDTH: 130px">
    <xsl:value-of select="my:field1"/>
</span>
```

The following XSLT fragment is an example of a section control, as specified in section 2.4.1.18, with the **SignatureBlock** attribute, as specified in section 2.4.2.31, set to "group3".

```
<div xd:disableEditing="yes" xd:SignatureBlock="group3"
xd:SignedSectionDisplaySignatures="true" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 6px;
BEHAVIOR: url (#default#SignaturesInDocUI); WIDTH: 651px">
...
</div>
```

The following XSLT fragment is an example of a section control with the **SignedSectionDisplaySignatures** attribute, as specified in section 2.4.2.32, set to "true".

```
<div xd:disableEditing="yes" xd:SignatureBlock="group3"
xd:SignedSectionDisplaySignatures="true" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 6px;
BEHAVIOR: url (#default#SignaturesInDocUI); WIDTH: 651px">
...
</div>
```

The following XSLT fragment is an example of a section control with the **SignedSectionName** attribute, as specified in section 2.4.2.33, set to "group3".

```
<div class="xdSection xdRepeating" title="" style="MARGIN-BOTTOM: 6px; WIDTH: 651px"
align="left" xd:xctname="Section" xd:CtrlId="CTRL4" xd:SignedSectionName="group3" tabIndex="-
1">
</div>
```

The following XSLT fragment is an example of a check box control with a **value** attribute, as specified in section 2.4.2.34.

```
<input class="xdBehavior_Boolean" title="" type="checkbox" xd:binding="my:field4"
tabIndex="0" xd:xctname="CheckBox" xd:CtrlId="CTRL9" xd:boundProp="xd:value" xd:offValue="1"
xd:onValue="true">
    <xsl:attribute name="xd:value">
        <xsl:value-of select="my:field4"/>
    </xsl:attribute>
    <xsl:if test="my:field4=&amp;quot;true&amp;quot;">
        <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
    </xsl:if>
</input> Field 4
```

xd:xctname

The following XSLT fragment is an example of a text box control with an **xctname** attribute, as specified in section 2.4.2.35, set to "PlainText".

```
<span class="xdTextBox" hideFocus="1" title="" xd:binding="my:field1" tabIndex="0"
xd:xctname="PlainText" xd:CtrlId="CTRL1" style="WIDTH: 130px">
    <xsl:value-of select="my:field1"/>
```

```
</span>
```

The following XSLT fragment is an example of a custom control with an **xctname** attribute set to "{{8e27c92b-1264-101c-8a2f-040224009c02}}".

```
<object class="xdActiveX" hideFocus="1" style="WIDTH: 288px; HEIGHT: 192px"
classid="clsid:8e27c92b-1264-101c-8a2f-040224009c02" tabIndex="0" tabStop="true"
xd:xctname="{{8e27c92b-1264-101c-8a2f-040224009c02}}" xd:CtrlId="CTRL2" xd:bindingType="text"
xd:bindingProperty="Value" xd:boundProp="xd:inline" contentEditable="false"
xd:binding="my:field2">
    <xsl:if test="function-available('xdImage:getImageUrl')">
        <xsl:attribute name="src"><xsl:value-of
select="xdImage:getImageUrl(my:field2)"/></xsl:attribute>
    </xsl:if>
...
</object>
```

The following XSLT fragment is an example of an optional section control, as specified in section 2.4.1.18, with the **xmlToEdit** attribute, as specified in section 2.4.2.36, set to "group3_2".

```
<xsl:choose>
    <xsl:when test="my:group3">
        <xsl:apply-templates select="my:group3" mode="_2"/>
    </xsl:when>
    <xsl:otherwise>
        <div class="optionalPlaceholder" xd:xmlToEdit="group3_2" tabIndex="0" align="left"
style="WIDTH: 651px">Click here to insert</div>
    </xsl:otherwise>
</xsl:choose>
```

3.4.3 XSL Function Extensions

The following examples demonstrate the use of a selection of function extensions. XSL function extensions are specified in section 2.4.3.

The following example XML is used as the target for the given XSL snippets in the first four examples:

SampleXML1.xml

```
<my:myFields>
      <my:group1>
            <my:group2>
                  <my:Values>6</my:Values>
            </my:group2>
            <my:group2>
                  <my:Values>12</my:Values>
            </my:group2>
<my:group2>
                  <my:Values>15</my:Values>
            </my:group2>
</my:group1>
      <my:Average>11</my:Average>
      <my:Maximum>15</my:Maximum>
      <my:Minimum>6</my:Minimum>
      <my:Date>2008-02-04</my:Date>
      <my:Seconds>3</my:Seconds>
      <my:ZipCode>98052x</my:ZipCode>
</my:myFields>
```

1. Conditional formatting with xdMath:Avg

xdMath:Avg can be used to conditionally change the **style** attribute of an HTML tag.

```
<xsl:attributename="style">WIDTH: 130px;
    <xsl:choose>
        <xsl:when test="my:Average = xdMath:Avg(my:group1/my:group2)">FONT-WEIGHT:
bold</xsl:when>
    </xsl:choose>
</xsl:attribute>
```

The **xdMath:Avg** function is used to calculate the average of the values in the **my:group2** nodes, which is 6+12+15 / 3 = 11. Because this value is equal to the value of the **my:Average** node, the test evaluates to true and **xsl:when** outputs **FONT-WEIGHT: bold** and makes the font bold.

2. Conditional formatting with xdDate:AddDays, xdDate:Today

This example is similar to the first example in that it shows how to perform conditional formatting using XSL function extensions.

```
<xsl:attributename="style">
    <xsl:choose>
        <xsl:when test = "my:Date = xdDate:AddDays(xdDate:Today() , 3)">FONT-WEIGHT: bold;
COLOR: #ff0000
        </xsl:when>
    </xsl:choose>
</xsl:attribute>
```

The **xsl:when** clause is used to test if the date in the **my:Date** node is three days past today. **xdDate:Today()** is used to get today's date and the output is passed in to the **AddDays** function. **AddDays** adds three days to today's date and outputs the date three days past today. In the SampleXML1.xml snippet, **my:Date** is "2008-02-04". If today is 2008-02-01, **test** evaluates to true and the **style** attribute has the value "FONT-WEIGHT: bold; COLOR: #ff0000".

3. Outputting time with xdDate:AddSeconds

```
<xsl:value-of select="xdDate:AddSeconds(xdDate:Now(), my:Seconds)"/>
```

The **xsl:value-of** clause outputs the value returned by the expression in the **select** attribute. **xdDate:AddSeconds** returns the time equivalent to now plus the number of seconds indicated in **my:Seconds**. **Xsl:value-of** outputs the resulting time.

4. Conditional formatting with xdUtil:Match

```
<spanclass="xdTextBox" hideFocus="1" title="" xd:CtrlId="CTRL12" xd:xctname="PlainText"
tabIndex="0" xd:binding="my:ZipCode">
    <xsl:if test="function-available('xdXDocument:GetDOM')">
        <xsl:attributename="style">
            <xsl:choose>
                <xsl:when test = "not( xdUtil:Match( string(my:ZipCode),
&amp;quot;\d\d\d\d\d&amp;quot; ) )">COLOR: #ff0000; TEXT-DECORATION: line-through
                </xsl:when>
            </xsl:choose>
            </xsl:attribute>
         <xsl:value-of select="my:ZipCode"/>
    </xsl:if>
</span>
```

In this example, the XSL representation of a text box control is given. This textbox is used to enter zip codes. To make users aware of the case where they enter an invalid zip code, the **xdUtil:Match**

function is used to check that the entered value conforms to the zip code pattern. The first parameter of the **xdUtil:Match** function takes the value of the **my:ZipCode** node. The second parameter is the regular expression "\d\d\d\d\d", which means five consecutive digits. If the given zip code is not composed of five digits, the textbox has the **style** "COLOR: #ff0000; TEXT-DECORATION: line-through", which makes it more visible to the user that the entered zip code is not valid. In SampleXML1.xml, **my:ZipCode** is "98052x", which does not match the given regular expression. Therefore, **test** evaluates to TRUE and the text box control's **style** attribute has the given **style** attributes.

The following example XML is used as the target for the given XSL snippets in the next example.

SampleXML2.xml

```
<root>
      <value>12</value>
      <value>14</value>
      <value>20</value>
</root>.
```

1. Getting data from a secondary data source using xdXDocument:GetDOM

This example uses SampleXML2.xml as the secondary data source of a form template (.xsn) file. This secondary data source is registered with a data source name "example". Data is retrieved from this data source for use within the XSL.

The example makes the font of a text box control bold if the average of the values in the secondary data source is less than 15.

```
<spanclass="xdTextBox"hideFocus="1"title=""tabIndex="0"xd:binding="my:field2"xd:xctname="Plai
nText"xd:CtrlId="CTRL3">
    <xsl:if test="function-available('xdXDocument:GetDOM')">
        <xsl:attribute name="style">WIDTH: 130px;
            <xsl:choose>
                <xsl:when test="xdMath:Avg(
xdXDocument:getDOM(&quot;example&quot;)/root/value )">FONT-WEIGHT: bold; caption: Conditional
Formatting 1
</xsl:when>
            </xsl:choose>
        </xsl:attribute>
        <xsl:value-ofselect="my:field2"/>
    </xsl:if>
</span>
```

In the preceding XSL snippet, secondary data source content is queried using **xdXDocument:getDom** and then an XPath expression is built on it. This XPath expression returns a **node-set** containing all the **value** nodes in the "example" data source. Then **xdMath:Avg** is used to calculate the average.

The following example XML is used as the target for the given XSL snippets in the next example.

SampleXML3.xml

```
<my:myFields>
      <my:group1>
            <my:group2>
                  <my:Count>3</my:Count>
                  <my:Value>4</my:Value>
                  <my:Total>12</my:Total>
            </my:group2>
            <my:group2>
                  <my:Count>12</my:Count>
```

```
                  <my:Value>20</my:Value>
                  <my:Total>240</my:Total>
            </my:group2>
            <my:group2>
                  <my:Count>4</my:Count>
                  <my:Value>8</my:Value>
                  <my:Total>12</my:Total>
            </my:group2>
      </my:group1>
</my:myFields>
```

1. Using xdMath:Eval

```
<span class="xdExpressionBox xdDataBindingUI" title="" xd:CtrlId="CTRL9"
xd:xctname="ExpressionBox" tabIndex="-1" xd:disableEditing="yes" style="WIDTH: 145px">
<xsl:if test="function-available('xdXDocument:GetDOM')">
<xsl:value-of select="sum(xdMath:Eval(my:group1/my:group2, &amp;quot;xdMath:Nz(my:Count) *
xdMath:Nz(my:Value)&amp;quot;))"/>
    </xsl:if>
</span>
```

This XSL example is an expression box control. The value of the expression box is calculated from SampleXML3.xml using **xdMath:Eval**. The XPath expression passed to **xdMath:Eval** returns a **node-set** containing all the **my:group2** nodes under **my:group1**. The **my:group2** nodes contain the child elements **my:Count** and **my:Value**. The expression passed in as the second parameter to the **xdMath:Eval** calculates the sum of the multiplication of the nodes from the first parameter. **xdMath:Eval** calculates this multiplication for every **group2** node and returns the result as a **node-set**.

Count	Value
3	4
12	20
4	8

The output is 3*4 + 12 * 20 + 4*8 = 284.

3.5 Print View Files (XSLT) Examples

The following example shows how a form view can be set as a print view of another form view. In this example, the first form view, "View 1", has two text box controls, as specified in section 2.4.1.20. The second form view, "Print Version View 1" has only one of the text box controls that "View 2" has and is set as a print view of "View 1".

The following XSLT excerpt shows two textbox controls for "View 1" from view1.xsl.

```
…
<div>
  <span class="xdTextBox" hideFocus="1" title="" xd:binding="my:field1" tabIndex="0"
xd:xctname="PlainText" xd:CtrlId="CTRL1" style="WIDTH: 130px">
    <xsl:value-of select="my:field1"/>
  </span>
  <span class="xdTextBox" hideFocus="1" title="" xd:binding="my:field2" tabIndex="0"
xd:xctname="PlainText" xd:CtrlId="CTRL2" style="WIDTH: 130px">
    <xsl:value-of select="my:field2"/>
  </span>
</div>
…
```

The following XSL excerpt shows only one control for "Print Version View 1" from PrintVersionView1.xsl.

```
…
<div>
  <span class="xdTextBox" hideFocus="1" title="" xd:CtrlId="CTRL2" xd:xctname="PlainText"
tabIndex="0" xd:binding="my:field2" style="WIDTH: 130px">
    <xsl:value-of select="my:field2"/>
  </span>
</div>
…
```

The following excerpt shows how the "Print Version View 1" form view is set as a print view of "View 1" in the manifest from manifest.xsf.

```
…
<xsf:views default="View 1">
  <xsf:view showMenuItem="yes" name="View 1" caption="View 1" printView="Print Version View
1">
    <xsf:mainpane transform="view1.xsl">
    </xsf:mainpane>
  </xsf:view>
  <xsf:view showMenuItem="yes" name="Print Version View 1" caption="Print Version View 1">
    <xsf:mainpane transform="PrintVersionView1.xsl">
    </xsf:mainpane>
  </xsf:view>
</xsf:views>
…
```

The **printView** attribute of the **view** element, as specified in section 2.2.123, for "View 1" points to "Print Version View 1". This notation defines "Print Version View 1" as a print version view of "View 1".

```
…
<xsf:view showMenuItem="yes" name="View 1" caption="View 1" printView="Print Version View 1">
…
```

3.6 Submit Files (XML) Examples

Data from an XML document is be consumed by the Web service method **GetRandom**. Following is a fragment from the WSDL file containing the XML schema for the **GetRandom** method.

```
<?xml version="1.0" encoding="utf-8" ?>
<wsdl:definitions xmlns:s1="http://microsoft.com/wsdl/types/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:i0="http://tempuri.org/twrSchema.xsd" xmlns:tns="http://webserviceserver/Everett"
targetNamespace="http://webserviceserver/Everett"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
    <wsdl:import namespace="http://tempuri.org/twrSchema.xsd"
location="http://webserviceserver/anon/Service1.asmx?schema=typedDataSet" />
    <wsdl:types>
        <s:schema elementFormDefault="qualified"
targetNamespace="http://webserviceserver/Everett">
            <s:import namespace="http://tempuri.org/twrSchema.xsd" />
            <s:import namespace="http://www.w3.org/2001/XMLSchema" />
            <s:import namespace="http://microsoft.com/wsdl/types/" />
            <s:element name="GetRandom">
                <s:complexType>
                    <s:sequence>
                        <s:element minOccurs="1" maxOccurs="1" name="seed" type="s:int" />
```

```
                        <s:element minOccurs="1" maxOccurs="1" name="min" type="s:int" />
                        <s:element minOccurs="1" maxOccurs="1" name="max" type="s:int" />
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:element name="GetRandomResponse">
                <s:complexType>
                    <s:sequence>
                        <s:element minOccurs="1" maxOccurs="1" name="GetRandomResult"
type="s:int" />
                    </s:sequence>
                </s:complexType>
            </s:element>
        </s:schema>
    </wsdl:types>
</wsdl:definitions>
```

The following example is how a Submit.xml looks when using the **GetRandom** method.

```
<dfs:myFields xmlns:dfs="http://schemas.microsoft.com/office/infopath/2003/dataFormSolution"
xmlns:tns="http://webserviceserver/Everett">
    <dfs:dataFields>
        <tns:GetRandom>
            <tns:seed></tns:seed>
            <tns:min></tns:min>
            <tns:max></tns:max>
        </tns:GetRandom>
    </dfs:dataFields>
</dfs:myFields>
```

As specified before, Submit.xml can broadly be categorized into two parts

- The first part is static and contains **<dfs:myFields>** and **<dfs:dataFields>**.
- The second part is the template for the **GetRandom** Web service method based on the Web service's WSDL.

The child of **<dfs:dataFields>** is **<tns:GetRandom>**, which is the template of the **GetRandom** method. The three parameters to this Web service method are *<tns:seed>*, *<tns:min>*, and *<tns:max>*. This subtree validates against the XML schema element **<s:element name="GetRandom">** in the WSDL example.

3.7 Template.XML Examples

The following example shows two textbox controls with default values. When the form template containing this template.xml file is opened for editing, the first textbox, which is bound to the **my:field1** field is populated with the initial value of "Jean Philippe", and the second text box, which is bound to the **my:field2** field is populated with the initial value of "Bagel".

```
<?xml version="1.0" encoding="UTF-8"?>
<?mso-infoPathSolution name="urn:schemas-microsoft-com:office:infopath:Template:-myXSD-2008-
02-15T23-26-48" href="manifest.xsf" solutionVersion="1.0.0.1" productVersion="12.0.0"
PIVersion="1.0.0.0" ?>
<?mso-application progid="InfoPath.Document" versionProgid="InfoPath.Document.2"?>
<my:myFields xmlns:my="http://schemas.microsoft.com/office/infopath/2003/myXSD/2008-02-
15T23:26:48">
  <my:field1>Jean Philippe</my:field1>
  <my:field2>Bagel</my:field2>
</my:myFields>
```

3.8 Upgrade.XSL Examples

There are two examples. One is upgrade.xsl and one is focused on the **MSXSL:node-set()** function.

3.8.1 Upgrade.XSL Example

The following example is a simple upgrade.xsl. The following example is a form file based on the original form template.

```
<?xml version="1.0" encoding="UTF-8"?>
<?mso-infoPathSolution solutionVersion="1.0.0.1" productVersion="12.0.0" PIVersion="1.0.0.0"
href="file:///C:\upgradeexample.xsn" name="urn:schemas-microsoft-
com:office:infopath:upgradeexample:-myXSD-2008-02-06T18-48-21" ?>
<?mso-application progid="InfoPath.Document" versionProgid="InfoPath.Document.2"?>
<my:myFields xmlns:my="http://schemas.microsoft.com/office/infopath/2003/myXSD/2008-02-
06T18:48:21" xml:lang="en-us">
  <my:field1>exampletext</my:field1>
  <my:field2>true</my:field2>
  <my:myRepeatingGroup>
    <my:fieldNumber xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">20000</my:fieldNumber>
  </my:myRepeatingGroup>
  <my:myRepeatingGroup>
    <my:fieldNumber>30000</my:fieldNumber>
  </my:myRepeatingGroup>
</my:myFields>
```

A new version of the form template is published, and the upgrade.xsl in the form template (.xsn) file is as follows.:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:msxsl="urn:schemas-
microsoft-com:xslt" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:my="http://schemas.microsoft.com/office/infopath/2003/myXSD/2008-02-06T18:48:21"
xmlns:xd="http://schemas.microsoft.com/office/infopath/2003" version="1.0">
  <xsl:output encoding="UTF-8" method="xml"/>
  <xsl:template match="/">
    <xsl:copy-of select="processing-instruction() | comment()"/>
    <xsl:choose>
      <xsl:when test="my:myFields">
        <xsl:apply-templates select="my:myFields" mode="_0"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:variable name="var">
          <xsl:element name="my:myFields"/>
        </xsl:variable>
        <xsl:apply-templates select="msxsl:node-set($var)/*" mode=" 0"/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
  <xsl:template match="my:myRepeatingGroup" mode="_1">
    <xsl:copy>
      <xsl:element name="my:fieldNumber">
        <xsl:choose>
          <xsl:when test="my:fieldNumber/text()[1]">
            <xsl:copy-of select="my:fieldNumber/text()[1]"/>
          </xsl:when>
          <xsl:otherwise>
            <xsl:attribute name="xsi:nil">true</xsl:attribute>
          </xsl:otherwise>
        </xsl:choose>
      </xsl:element>
      <xsl:element name="my:fieldText">
        <xsl:copy-of select="my:fieldText/text()[1]"/>
      </xsl:element>
```

```
    </xsl:copy>
  </xsl:template>
  <xsl:template match="my:myFields" mode="_0">
    <xsl:copy>
      <xsl:element name="my:field1">
        <xsl:copy-of select="my:field1/text()[1]"/>
      </xsl:element>
      <xsl:choose>
        <xsl:when test="my:myRepeatingGroup">
          <xsl:apply-templates select="my:myRepeatingGroup" mode="_1"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:variable name="var">
            <xsl:element name="my:myRepeatingGroup"/>
          </xsl:variable>
          <xsl:apply-templates select="msxsl:node-set($var)/*" mode="_1"/>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:copy>
  </xsl:template>
</xsl:stylesheet>
```

After upgrade.xsl is applied, the form file is as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<?mso-infoPathSolution productVersion="12.0.0" PIVersion="1.0.0.0" href="
file:///C:\upgradeexample.xsn" name="urn:schemas-microsoft-
com:office:infopath:upgradeexample:-myXSD-2008-02-06T18-48-21" solutionVersion="1.0.0.3" ?>
<?mso-application progid="InfoPath.Document" versionProgid="InfoPath.Document.2"?>
<my:myFields xmlns:my="http://schemas.microsoft.com/office/infopath/2003/myXSD/2008-02-
06T18:48:21" xml:lang="en-us">
  <my:field1>exampletext</my:field1>
  <my:myRepeatingGroup>
    <my:fieldNumber>20000</my:fieldNumber>
    <my:fieldText></my:fieldText>
  </my:myRepeatingGroup>
  <my:myRepeatingGroup>
    <my:fieldNumber>30000</my:fieldNumber>
    <my:fieldText></my:fieldText>
  </my:myRepeatingGroup>
</my:myFields>
```

The upgrade.xsl file has the following effects:

- **my:field2** was removed. That data was discarded.
- **my:myRepeatingGroup** has a new field, **my:fieldText**, which is empty for all XML nodes under **my:myRepeatingGroup**.
- The data for all other fields was retained and moved to the new data source.

3.8.2 MSXSL:Node-Set() Example

In the following example, **$var** is a variable that is an XML node tree in the style sheet. The **for-each** statement combined with the **node-set** function allows the user to iterate over this XML node tree as an XML node set.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:msxsl="urn:schemas-microsoft-com:xslt"
                xmlns:user="http://www.contoso.com"
                version="1.0">
   <xsl:variable name="books">
```

```
         <book author="Michael Howard">Writing Secure Code</book>
         <book author="Michael Kay">XSLT Reference</book>
      </xsl:variable>

      <xsl:template match="/">
         <authors>
            <xsl:for-each select="msxsl:node-set($books)/book">
            <author><xsl:value-of select="@author"/)</author>
            </xsl:for-each>
         </authors>
      </xsl:template>
   </xsl:stylesheet>
```

This transformation outputs the following.

```
<?xml version="1.0" encoding="utf-8"?>

<authors><author>Michael Howard</author><author>Michael Kay</author></authors>
```

4 Security Considerations

This section describes security considerations related to the InfoPath form template format and the template.XML specification.

4.1 The InfoPath Form Template Format

Because a form template (.xsn) file is a cabinet (.cab) file, as described in [MC-MCF], any and all security considerations including, but not limited to, digital signatures affecting cabinet (.cab) files affect form template (.xsn) files.

4.2 Template.XML Specification

Because template.xml is a form file, as described in [MS-IPFFX], any and all security considerations including, but not limited to, code signatures affecting form files affect template.xml files.

5 Appendix A: Full XML Schemas

For ease of implementation, this section provides the full XML schemas for the InfoPath XSF and XSF2 namespaces, as specified in section 2.2.

5.1 The InfoPath XSF XSD

The XML schema for the form definition (.xsf) file can also be found at the location described by [MSDN-XSF].

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsf="http://schemas.microsoft.com/office/infopath/2003/solutionDefinition"
  targetNamespace="http://schemas.microsoft.com/office/infopath/2003/solutionDefinition"
elementFormDefault="qualified"
  attributeFormDefault="unqualified">
  <!-- xdTitle type -->
  <xsd:simpleType name="xdTitle">
    <xsd:restriction base="xsd:string">
      <xsd:minLength value="1" />
      <xsd:maxLength value="255" />
      <xsd:pattern
value="([^\p{Z}\p{Cc}\p{Cf}\p{Cn}])(([^\p{Zl}\p{Zp}\p{Cc}])*([^\p{Z}\p{Cc}\p{Cf}\p{Cn}]))?"
/>
    </xsd:restriction>
  </xsd:simpleType>
  <!-- xdViewName type -->
  <xsd:simpleType name="xdViewName">
    <xsd:restriction base="xsd:string">
      <xsd:minLength value="1" />
      <xsd:maxLength value="255" />
      <xsd:pattern
value="([^\p{Z}\p{C}/\\#&amp;&quot;&gt;&lt;])(([^\p{Zl}\p{Zp}\p{C}/\\#&amp;&quot;&gt;&lt;])*(
[^\p{Z}\p{C}/\\#&amp;&quot;&gt;&lt;]))?" />
    </xsd:restriction>
  </xsd:simpleType>
  <!-- xdRoleName type -->
  <!-- uses xdViewName as base -->
  <xsd:simpleType name="xdRoleName">
    <xsd:restriction base="xsf:xdViewName"></xsd:restriction>
  </xsd:simpleType>
  <!-- xdYesNo type -->
  <xsd:simpleType name="xdYesNo">
    <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="yes" />
      <xsd:enumeration value="no" />
    </xsd:restriction>
  </xsd:simpleType>
  <!-- xdEnabledDisabled type -->
  <xsd:simpleType name="xdEnabledDisabled">
    <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="enabled" />
      <xsd:enumeration value="disabled" />
    </xsd:restriction>
  </xsd:simpleType>
  <!-- xdManualAuto type -->
  <xsd:simpleType name="xdManualAuto">
    <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="manual" />
      <xsd:enumeration value="automatic" />
    </xsd:restriction>
  </xsd:simpleType>
  <!-- xdExpressionLiteral type -->
  <xsd:simpleType name="xdExpressionLiteral">
    <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="expression" />
```

```
        <xsd:enumeration value="literal" />
      </xsd:restriction>
    </xsd:simpleType>
    <!-- xdFileName type -->
    <xsd:simpleType name="xdFileName">
      <xsd:restriction base="xsd:string">
        <xsd:minLength value="1" />
        <xsd:maxLength value="64" />
      </xsd:restriction>
    </xsd:simpleType>
    <!-- xdScriptLanguage type -->
    <xsd:simpleType name="xdScriptLanguage">
      <xsd:restriction base="xsd:NMTOKEN">
        <xsd:pattern
  value="((([Jj][Aa][Vv][Aa]|(([Jj])|([Vv][Bb])))([Ss][Cc][Rr][Ii][Pp][Tt]))([.][Ee][Nn][Cc][Oo
  ][Dd][Ee]))|(([Jj][Aa][Vv][Aa]|(([Jj])|([Vv][Bb])))([Ss][Cc][Rr][Ii][Pp][Tt]))|([Mm][Aa][Nn][
  Aa][Gg][Ee][Dd][Cc][Oo][Dd][Ee])" />
      </xsd:restriction>
    </xsd:simpleType>
    <!-- xdSolutionVersion type -->
    <xsd:simpleType name="xdSolutionVersion">
      <xsd:restriction base="xsd:string">
        <xsd:pattern value="(([0-9]{1,4}.){3}[0-9]{1,4})" />
      </xsd:restriction>
    </xsd:simpleType>
    <!-- xdEmptyString type -->
    <xsd:simpleType name="xdEmptyString">
      <xsd:restriction base="xsd:string">
        <xsd:maxLength value="0" />
      </xsd:restriction>
    </xsd:simpleType>
    <!-- xdErrorMessage type -->
    <xsd:simpleType name="xdErrorMessage">
      <xsd:restriction base="xsd:string">
        <xsd:maxLength value="1023" />
      </xsd:restriction>
    </xsd:simpleType>
    <!-- xdDesignMode type -->
    <xsd:simpleType name="xdDesignMode">
      <xsd:restriction base="xsd:NMTOKEN">
        <xsd:enumeration value="normal" />
        <xsd:enumeration value="protected" />
      </xsd:restriction>
    </xsd:simpleType>
    <!-- xdTrustLevel type -->
    <xsd:simpleType name="xdTrustLevel">
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="restricted" />
        <xsd:enumeration value="domain" />
      </xsd:restriction>
    </xsd:simpleType>
    <!-- xdSignedDataBlockName type -->
    <xsd:simpleType name="xdSignedDataBlockName">
      <xsd:restriction base="xsd:ID">
        <xsd:minLength value="1" />
        <xsd:maxLength value="255" />
      </xsd:restriction>
    </xsd:simpleType>
    <!-- xdSignedDataBlockMessage type -->
    <xsd:simpleType name="xdSignedDataBlockMessage">
      <xsd:restriction base="xsd:string">
        <xsd:maxLength value="255" />
      </xsd:restriction>
    </xsd:simpleType>
    <!-- xdSignatureRelationEnum type -->
    <xsd:simpleType name="xdSignatureRelationEnum">
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="countersign" />
        <xsd:enumeration value="cosign" />
```

```
        <xsd:enumeration value="single" />
      </xsd:restriction>
    </xsd:simpleType>
    <!-- xdHWSname type -->
    <xsd:simpleType name="xdHWSname">
      <xsd:restriction base="xsd:NCName">
        <xsd:pattern value="[^-^\.^\\^\[^\]^\|^\+^?^\*^@^\{^\}^\(^\)^&gt;^&lt;^=^;^,]*" />
      </xsd:restriction>
    </xsd:simpleType>
    <!-- xdHWSCaption type -->
    <xsd:simpleType name="xdHWSCaption">
      <xsd:restriction base="xsd:string">
        <xsd:minLength value="1" />
        <xsd:maxLength value="255" />
      </xsd:restriction>
    </xsd:simpleType>
    <!-- xDocumentClass -->
    <xsd:element name="xDocumentClass">
      <xsd:complexType>
        <xsd:all>
          <xsd:element ref="xsf:package" minOccurs="1" />
          <xsd:element ref="xsf:permissions" minOccurs="0" />
          <xsd:element ref="xsf:views" minOccurs="1" />
          <xsd:element ref="xsf:hwsWorkflow" minOccurs="0" />
          <xsd:element ref="xsf:externalViews" minOccurs="0" />
          <xsd:element ref="xsf:scripts" minOccurs="0" />
          <xsd:element ref="xsf:schemaErrorMessages" minOccurs="0" />
          <xsd:element ref="xsf:documentSchemas" minOccurs="0" />
          <xsd:element ref="xsf:applicationParameters" minOccurs="0" />
          <xsd:element ref="xsf:featureRestrictions" minOccurs="0" />
          <xsd:element ref="xsf:fileNew" minOccurs="0" />
          <xsd:element ref="xsf:customValidation" minOccurs="0" />
          <xsd:element ref="xsf:domEventHandlers" minOccurs="0" />
          <xsd:element ref="xsf:importParameters" minOccurs="0" />
          <xsd:element ref="xsf:listProperties" minOccurs="0" />
          <xsd:element ref="xsf:taskpane" minOccurs="0" />
          <xsd:element ref="xsf:documentSignatures" minOccurs="0" />
          <xsd:element ref="xsf:dataObjects" minOccurs="0" />
          <xsd:element ref="xsf:dataAdapters" minOccurs="0" />
          <xsd:element ref="xsf:query" minOccurs="0" />
          <xsd:element ref="xsf:submit" minOccurs="0" />
          <xsd:element ref="xsf:save" minOccurs="0" />
          <xsd:element ref="xsf:roles" minOccurs="0" />
          <xsd:element ref="xsf:onLoad" minOccurs="0" />
          <xsd:element ref="xsf:documentVersionUpgrade" minOccurs="0" />
          <xsd:element ref="xsf:extensions" minOccurs="0" />
          <xsd:element ref="xsf:ruleSets" minOccurs="0" />
          <xsd:element ref="xsf:calculations" minOccurs="0" />
        </xsd:all>
        <xsd:attribute name="name" type="xsd:string" use="optional" />
        <xsd:attribute name="author" type="xsd:string" use="optional" />
        <xsd:attribute name="description" use="optional">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:maxLength value="255" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="solutionVersion" type="xsf:xdSolutionVersion" use="optional" />
        <xsd:attribute name="productVersion" type="xsd:string" use="optional" />
        <xsd:attribute name="solutionFormatVersion" type="xsf:xdSolutionVersion" use="required"
/>
        <xsd:attribute name="dataFormSolution" type="xsf:xdYesNo" use="optional" />
        <xsd:attribute name="requireFullTrust" type="xsf:xdYesNo" use="optional" />
        <xsd:attribute name="trustLevel" type="xsf:xdTrustLevel" use="optional" />
        <xsd:attribute name="trustSetting" type="xsf:xdManualAuto" use="optional" />
        <xsd:attribute name="publishUrl" type="xsd:string" use="optional" />
      </xsd:complexType>
      <xsd:key name="view_name_key">
```

```
        <xsd:selector xpath="./xsf:views/xsf:view" />
        <xsd:field xpath="@name" />
      </xsd:key>
      <xsd:key name="externalView_name_key">
        <xsd:selector xpath="./xsf:externalViews/xsf:externalView" />
        <xsd:field xpath="@name" />
      </xsd:key>
      <xsd:key name="view_or_externalView_name_key">
        <xsd:selector xpath="./xsf:views/xsf:view | ./xsf:externalViews/xsf:externalView" />
        <xsd:field xpath="@name" />
      </xsd:key>
      <xsd:key name="ruleset_name_key">
        <xsd:selector xpath="./xsf:ruleSets/xsf:ruleSet" />
        <xsd:field xpath="@name" />
      </xsd:key>
      <xsd:key name="dataObject_name_key">
        <xsd:selector xpath="./xsf:dataObjects/xsf:dataObject" />
        <xsd:field xpath="@name" />
      </xsd:key>
      <xsd:unique name="adapter_name_unique">
        <xsd:selector xpath="./xsf:dataObjects/xsf:dataObject/xsf:query/* | ./xsf:query/* |
./xsf:dataAdapters/* | ./xsf:submit/xsf:webServiceAdapter | ./xsf:submit/xsf:davAdapter |
./xsf:submit/xsf:emailAdapter | ./xsf:submit/xsf:submitToHostAdapter" />
        <xsd:field xpath="@name" />
      </xsd:unique>
      <xsd:key name="adapter_name_key">
        <xsd:selector xpath="./xsf:dataAdapters/*" />
        <xsd:field xpath="@name" />
      </xsd:key>
      <xsd:unique name="view_external_name_unique">
        <xsd:selector xpath="./xsf:views/xsf:view | ./xsf:externalViews/xsf:externalView" />
        <xsd:field xpath="@name" />
      </xsd:unique>
    </xsd:element>
    <!-- schemaErrorMessages -->
    <xsd:element name="schemaErrorMessages">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="xsf:override" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <!-- override -->
    <xsd:element name="override">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="xsf:errorMessage" />
        </xsd:sequence>
        <xsd:attribute name="match" type="xsd:string" use="required" />
      </xsd:complexType>
    </xsd:element>
    <!-- applicationParameters -->
    <xsd:element name="applicationParameters">
      <xsd:complexType>
        <xsd:all>
          <xsd:element ref="xsf:solutionProperties" minOccurs="0" />
        </xsd:all>
        <xsd:attribute name="application" use="required">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="InfoPath Design Mode" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
      </xsd:complexType>
    </xsd:element>
    <!-- solutionProperties -->
    <xsd:element name="solutionProperties">
      <xsd:complexType>
```

```
      <xsd:attribute name="allowCustomization" type="xsf:xdYesNo" use="optional" />
      <xsd:attribute name="lastOpenView" use="optional" />
      <xsd:attribute name="scriptLanguage" type="xsf:xdScriptLanguage" use="optional" />
      <xsd:attribute name="automaticallyCreateNodes" type="xsf:xdYesNo" use="optional" />
      <xsd:attribute name="lastVersionNeedingTransform" type="xsf:xdSolutionVersion"
use="optional" />
      <xsd:attribute name="fullyEditableNamespace" type="xsd:anyURI" use="optional" />
      <xsd:attribute name="publishSaveUrl" type="xsd:string" use="optional" />
    </xsd:complexType>
  </xsd:element>
  <!-- featureRestrictions -->
  <xsd:element name="featureRestrictions">
    <xsd:complexType>
      <xsd:all>
        <xsd:element name="save" minOccurs="0">
          <xsd:complexType>
            <xsd:attribute name="ui" type="xsf:xdEnabledDisabled" use="required" />
          </xsd:complexType>
        </xsd:element>
        <xsd:element ref="xsf:exportToWeb" minOccurs="0" />
        <xsd:element ref="xsf:exportToExcel" minOccurs="0" />
        <xsd:element ref="xsf:print" minOccurs="0" />
        <xsd:element ref="xsf:sendMail" minOccurs="0" />
        <xsd:element ref="xsf:autoRecovery" minOccurs="0" />
      </xsd:all>
    </xsd:complexType>
  </xsd:element>
  <!-- exportToWeb -->
  <xsd:element name="exportToWeb">
    <xsd:complexType>
      <xsd:attribute name="ui" type="xsf:xdEnabledDisabled" use="required" />
    </xsd:complexType>
  </xsd:element>
  <!-- exportToExcel -->
  <xsd:element name="exportToExcel">
    <xsd:complexType>
      <xsd:attribute name="ui" type="xsf:xdEnabledDisabled" use="required" />
    </xsd:complexType>
  </xsd:element>
  <!-- print -->
  <xsd:element name="print">
    <xsd:complexType>
      <xsd:attribute name="ui" type="xsf:xdEnabledDisabled" use="required" />
    </xsd:complexType>
  </xsd:element>
  <!-- sendMail -->
  <xsd:element name="sendMail">
    <xsd:complexType>
      <xsd:attribute name="ui" type="xsf:xdEnabledDisabled" use="required" />
    </xsd:complexType>
  </xsd:element>
  <!-- autoRecovery -->
  <xsd:element name="autoRecovery">
    <xsd:complexType>
      <xsd:attribute name="feature" type="xsf:xdEnabledDisabled" use="required" />
    </xsd:complexType>
  </xsd:element>
  <!-- query -->
  <xsd:element name="query">
    <xsd:complexType>
      <xsd:choice>
        <xsd:element ref="xsf:queryAction" />
        <xsd:element ref="xsf:adoAdapter" />
        <xsd:element ref="xsf:webServiceAdapter" />
        <xsd:element ref="xsf:xmlFileAdapter" />
        <xsd:element ref="xsf:sharepointListAdapter" />
      </xsd:choice>
    </xsd:complexType>
  </xsd:element>
```

```
    <!-- scripts -->
    <xsd:element name="scripts">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="xsf:script" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute name="language" type="xsf:xdScriptLanguage" use="required" />
        <xsd:attribute name="enforceScriptTimeout" type="xsf:xdYesNo" use="optional"
  default="yes" />
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="script">
      <xsd:complexType>
        <xsd:attribute name="src" type="xsf:xdFileName" use="required" />
      </xsd:complexType>
    </xsd:element>
    <!-- dataObjects -->
    <xsd:element name="dataObjects">
      <xsd:complexType>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
          <xsd:element ref="xsf:dataObject" />
        </xsd:choice>
      </xsd:complexType>
      <xsd:unique name="dataObjects_name_unique">
        <xsd:selector xpath="./xsf:dataObject" />
        <xsd:field xpath="@name" />
      </xsd:unique>
    </xsd:element>
    <xsd:element name="dataObject">
      <xsd:complexType>
        <xsd:choice>
          <xsd:element name="query">
            <xsd:complexType>
              <xsd:choice>
                <xsd:element ref="xsf:adoAdapter" />
                <xsd:element ref="xsf:webServiceAdapter" />
                <xsd:element ref="xsf:xmlFileAdapter" />
                <xsd:element ref="xsf:sharepointListAdapter" />
              </xsd:choice>
            </xsd:complexType>
          </xsd:element>
        </xsd:choice>
        <xsd:attribute name="name" type="xsf:xdTitle" use="required" />
        <xsd:attribute name="schema" type="xsd:string" use="optional" />
        <xsd:attribute name="initOnLoad" type="xsf:xdYesNo" use="optional" />
      </xsd:complexType>
    </xsd:element>
    <!-- dataAdapters -->
    <xsd:element name="adoAdapter">
      <xsd:complexType>
        <xsd:attribute name="name" type="xsf:xdTitle" use="optional" />
        <xsd:attribute name="connectionString" type="xsd:string" use="required" />
        <xsd:attribute name="commandText" type="xsd:string" use="required" />
        <xsd:attribute name="queryAllowed" type="xsf:xdYesNo" use="optional" />
        <xsd:attribute name="submitAllowed" type="xsf:xdYesNo" use="optional" />
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="webServiceAdapter">
      <xsd:complexType>
        <xsd:choice>
          <xsd:element ref="xsf:operation" />
        </xsd:choice>
        <xsd:attribute name="name" type="xsf:xdTitle" use="optional" />
        <xsd:attribute name="wsdlUrl" type="xsd:string" use="required" />
        <xsd:attribute name="queryAllowed" type="xsf:xdYesNo" use="optional" />
        <xsd:attribute name="submitAllowed" type="xsf:xdYesNo" use="optional" />
        <xsd:attribute name="useDataSet" type="xsf:xdYesNo" use="optional" />
      </xsd:complexType>
    </xsd:element>
```

```
<xsd:element name="hwsAdapter">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element ref="xsf:hwsOperation" />
    </xsd:choice>
    <xsd:attribute name="name" type="xsf:xdTitle" use="required" />
    <xsd:attribute name="wsdlUrl" type="xsd:string" use="required" />
    <xsd:attribute name="queryAllowed" type="xsf:xdYesNo" use="optional" />
    <xsd:attribute name="submitAllowed" type="xsf:xdYesNo" use="optional" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="operation">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element ref="xsf:input" minOccurs="0" />
    </xsd:choice>
    <xsd:attribute name="name" type="xsd:string" use="required" />
    <xsd:attribute name="soapAction" type="xsd:string" use="required" />
    <xsd:attribute name="serviceUrl" type="xsd:string" use="required" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="hwsOperation">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element ref="xsf:input" />
    </xsd:choice>
    <xsd:attribute name="type" type="xsd:string" use="required" />
    <xsd:attribute name="typeID" type="xsd:string" use="required" />
    <xsd:attribute name="serviceUrl" type="xsd:string" use="required" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="input">
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element ref="xsf:partFragment" />
    </xsd:choice>
    <xsd:attribute name="source" type="xsd:string" use="required" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="partFragment">
  <xsd:complexType>
    <xsd:attribute name="match" type="xsd:string" use="required" />
    <xsd:attribute name="replaceWith" type="xsd:string" use="required" />
    <xsd:attribute name="sendAsString" type="xsf:xdYesNo" use="optional" />
    <xsd:attribute name="dataObject" type="xsd:string" use="optional" />
    <xsd:attribute name="filter" type="xsd:string" use="optional" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="xmlFileAdapter">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsf:xdTitle" use="optional" />
    <xsd:attribute name="fileUrl" type="xsd:anyURI" use="required" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="sharepointListAdapter">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="field" minOccurs="0" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:attribute name="sharepointName" type="xsd:string" use="required" />
          <xsd:attribute name="infopathName" type="xsd:string" use="required" />
          <xsd:attribute name="isLookup" type="xsf:xdYesNo" use="optional" />
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsf:xdTitle" use="required" />
    <xsd:attribute name="siteUrl" type="xsd:string" use="required" />
    <xsd:attribute name="sharepointGuid" type="xsd:string" use="required" />
    <xsd:attribute name="infopathGroup" type="xsd:string" use="required" />
```

```
    <xsd:attribute name="queryAllowed" type="xsf:xdYesNo" use="optional" />
    <xsd:attribute name="submitAllowed" type="xsf:xdYesNo" use="optional" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="davAdapter">
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="folderURL">
        <xsd:complexType>
          <xsd:attribute name="value" type="xsd:string" use="required" />
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="fileName">
        <xsd:complexType>
          <xsd:attribute name="value" type="xsd:string" use="required" />
          <xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="optional" />
        </xsd:complexType>
      </xsd:element>
    </xsd:all>
    <xsd:attribute name="name" type="xsf:xdTitle" use="required" />
    <xsd:attribute name="overwriteAllowed" type="xsf:xdYesNo" use="optional" />
    <xsd:attribute name="queryAllowed" type="xsf:xdYesNo" use="optional" />
    <xsd:attribute name="submitAllowed" type="xsf:xdYesNo" use="optional" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="emailAdapter">
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="to" minOccurs="0">
        <xsd:complexType>
          <xsd:attribute name="value" type="xsd:string" use="required" />
          <xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="optional" />
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="cc" minOccurs="0">
        <xsd:complexType>
          <xsd:attribute name="value" type="xsd:string" use="required" />
          <xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="optional" />
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="bcc" minOccurs="0">
        <xsd:complexType>
          <xsd:attribute name="value" type="xsd:string" use="required" />
          <xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="optional" />
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="subject" minOccurs="0">
        <xsd:complexType>
          <xsd:attribute name="value" type="xsd:string" use="required" />
          <xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="optional" />
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="intro" minOccurs="0">
        <xsd:complexType>
          <xsd:attribute name="value" type="xsd:string" use="required" />
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="attachmentFileName" minOccurs="0">
        <xsd:complexType>
          <xsd:attribute name="value" type="xsd:string" use="required" />
          <xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="optional" />
        </xsd:complexType>
      </xsd:element>
    </xsd:all>
    <xsd:attribute name="name" type="xsf:xdTitle" use="required" />
    <xsd:attribute name="queryAllowed" type="xsf:xdYesNo" use="optional" />
    <xsd:attribute name="submitAllowed" type="xsf:xdYesNo" use="optional" />
  </xsd:complexType>
</xsd:element>
```

```
<xsd:element name="submitToHostAdapter">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsf:xdTitle" use="required" />
    <xsd:attribute name="queryAllowed" type="xsf:xdYesNo" use="optional" />
    <xsd:attribute name="submitAllowed" type="xsf:xdYesNo" use="optional" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="dataAdapters">
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element ref="xsf:adoAdapter" />
      <xsd:element ref="xsf:webServiceAdapter" />
      <xsd:element ref="xsf:xmlFileAdapter" />
      <xsd:element ref="xsf:sharepointListAdapter" />
      <xsd:element ref="xsf:davAdapter" />
      <xsd:element ref="xsf:emailAdapter" />
      <xsd:element ref="xsf:submitToHostAdapter" />
      <xsd:element ref="xsf:hwsAdapter" />
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
<!-- documentSchemas -->
<xsd:element name="documentSchemas">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:documentSchema" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="documentSchema">
  <xsd:complexType>
    <xsd:attribute name="location" type="xsd:string" use="required" />
    <xsd:attribute name="rootSchema" type="xsf:xdYesNo" />
  </xsd:complexType>
</xsd:element>
<!-- customValidation -->
<xsd:element name="customValidation">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:errorCondition" minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="errorCondition">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:errorMessage" />
    </xsd:sequence>
    <xsd:attribute name="match" type="xsd:string" use="required" />
    <xsd:attribute name="expression" type="xsd:string" use="required" />
    <xsd:attribute name="expressionContext" type="xsd:string" use="optional" />
    <xsd:attribute name="showErrorOn" type="xsd:string" use="optional" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="errorMessage">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsf:xdErrorMessage">
        <xsd:attribute name="type" use="optional">
          <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
              <xsd:enumeration value="modal" />
              <xsd:enumeration value="modeless" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="shortMessage" use="required">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
```

```
              <xsd:maxLength value="127" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
<!-- domEventHandlers -->
<xsd:element name="domEventHandlers">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:domEventHandler" minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:unique name="domEventHandler_handlerObject_unique">
    <xsd:selector xpath="." />
    <xsd:field xpath="@handlerObject" />
  </xsd:unique>
</xsd:element>
<xsd:element name="domEventHandler">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:ruleSetAction" minOccurs="0" maxOccurs="1" />
    </xsd:sequence>
    <xsd:attribute name="dataObject" type="xsd:string" use="optional" />
    <xsd:attribute name="match" type="xsd:string" use="required" />
    <xsd:attribute name="handlerObject" type="xsd:string" use="optional" />
  </xsd:complexType>
  <xsd:keyref name="domEventHandler_ruleSetAction" refer="xsf:ruleset_name_key">
    <xsd:selector xpath="./xsf:ruleSetAction" />
    <xsd:field xpath="@ruleSet" />
  </xsd:keyref>
</xsd:element>
<!-- importParameters -->
<xsd:element name="importParameters">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:importSource" minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:attribute name="enabled" type="xsf:xdYesNo" use="required" />
    <xsd:attribute name="useScriptHandler" type="xsf:xdYesNo" use="optional" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="importSource">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsd:string" use="required" />
    <xsd:attribute name="schema" type="xsf:xdFileName" use="required" />
    <xsd:attribute name="transform" type="xsf:xdFileName" use="required" />
    <xsd:attribute name="authoringOfTransform" type="xsf:xdManualAuto" use="optional" />
  </xsd:complexType>
</xsd:element>
<!-- listProperties -->
<xsd:element name="listProperties">
  <xsd:complexType>
    <xsd:all>
      <xsd:element ref="xsf:fields" />
    </xsd:all>
  </xsd:complexType>
</xsd:element>
<xsd:element name="fields">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:field" minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="field">
  <xsd:complexType>
```

```
      <xsd:attribute name="type" type="xsd:NMTOKEN" use="required" />
      <xsd:attribute name="name" type="xsf:xdTitle" use="required" />
      <xsd:attribute name="columnName" type="xsf:xdTitle" use="required" />
      <xsd:attribute name="required" type="xsf:xdYesNo" use="optional" />
      <xsd:attribute name="viewable" type="xsf:xdYesNo" use="optional" />
      <xsd:attribute name="node" type="xsd:string" use="required" />
      <xsd:attribute name="maxLength" type="xsd:byte" />
      <xsd:attribute name="aggregation" use="optional">
        <xsd:simpleType>
          <xsd:restriction base="xsd:NMTOKEN">
            <xsd:enumeration value="sum" />
            <xsd:enumeration value="count" />
            <xsd:enumeration value="average" />
            <xsd:enumeration value="min" />
            <xsd:enumeration value="max" />
            <xsd:enumeration value="first" />
            <xsd:enumeration value="last" />
            <xsd:enumeration value="merge" />
            <xsd:enumeration value="plaintext" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:complexType>
  </xsd:element>
  <!-- submit -->
  <xsd:element name="submit">
    <xsd:complexType>
      <xsd:all>
        <xsd:element name="submitAction" minOccurs="0">
          <xsd:complexType>
            <xsd:attribute name="adapter" type="xsf:xdTitle" use="required" />
          </xsd:complexType>
          <xsd:keyref name="submitAdapter_name_keyref" refer="xsf:adapter_name_key">
            <xsd:selector xpath="." />
            <xsd:field xpath="@adapter" />
          </xsd:keyref>
        </xsd:element>
        <xsd:element ref="xsf:useHttpHandler" minOccurs="0" />
        <xsd:element ref="xsf:useScriptHandler" minOccurs="0" />
        <xsd:element ref="xsf:ruleSetAction" minOccurs="0" />
        <xsd:element ref="xsf:useQueryAdapter" minOccurs="0" />
        <xsd:element ref="xsf:webServiceAdapter" minOccurs="0" />
        <xsd:element ref="xsf:davAdapter" minOccurs="0" />
        <xsd:element ref="xsf:emailAdapter" minOccurs="0" />
        <xsd:element ref="xsf:submitToHostAdapter" minOccurs="0" />
        <xsd:element name="successMessage" type="xsd:string" minOccurs="0" />
        <xsd:element name="errorMessage" type="xsd:string" minOccurs="0" />
      </xsd:all>
      <xsd:attribute name="caption" type="xsd:string" use="optional" />
      <xsd:attribute name="onAfterSubmit" use="optional">
        <xsd:simpleType>
          <xsd:restriction base="xsd:NMTOKEN">
            <xsd:enumeration value="close" />
            <xsd:enumeration value="keepOpen" />
            <xsd:enumeration value="openNew" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="showStatusDialog" type="xsf:xdYesNo" use="optional" />
      <xsd:attribute name="showSignatureReminder" type="xsf:xdYesNo" use="optional" />
      <xsd:attribute name="disableMenuItem" type="xsf:xdYesNo" use="optional" />
    </xsd:complexType>
    <xsd:keyref name="submit ruleSetAction" refer="xsf:ruleset name key">
      <xsd:selector xpath="./xsf:ruleSetAction" />
      <xsd:field xpath="@ruleSet" />
    </xsd:keyref>
  </xsd:element>
  <xsd:element name="useHttpHandler">
    <xsd:complexType>
```

```
      <xsd:attribute name="method" use="required">
        <xsd:simpleType>
          <xsd:restriction base="xsd:NMTOKEN">
            <xsd:enumeration value="POST" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="href" type="xsd:anyURI" use="required" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="useScriptHandler" />
  <xsd:element name="useQueryAdapter" />
  <!-- onLoad -->
  <xsd:element name="onLoad">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:ruleSetAction" minOccurs="1" maxOccurs="1" />
      </xsd:sequence>
    </xsd:complexType>
    <xsd:keyref name="load_ruleSetAction" refer="xsf:ruleset_name_key">
      <xsd:selector xpath="./xsf:ruleSetAction" />
      <xsd:field xpath="@ruleSet" />
    </xsd:keyref>
  </xsd:element>
  <!-- save -->
  <xsd:element name="save">
    <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="1">
        <xsd:element ref="xsf:useScriptHandler" />
      </xsd:choice>
    </xsd:complexType>
  </xsd:element>
  <!-- roles -->
  <xsd:element name="roles">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:role" minOccurs="1" maxOccurs="unbounded" />
        <xsd:element ref="xsf:membership" minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
      <xsd:attribute name="default" type="xsd:string" use="required" />
      <xsd:attribute name="initiator" type="xsd:string" use="optional" />
      <xsd:attribute name="hideStatusBarDisplay" type="xsf:xdYesNo" use="optional" />
    </xsd:complexType>
    <!-- role names must be unique -->
    <xsd:unique name="roles_name_unique">
      <xsd:selector xpath="./xsf:role" />
      <xsd:field xpath="@name" />
    </xsd:unique>
    <!-- fields must reference existing role -->
    <xsd:key name="role_name_key">
      <xsd:selector xpath="./xsf:role" />
      <xsd:field xpath="@name" />
    </xsd:key>
    <xsd:keyref name="role default" refer="xsf:role name key">
      <xsd:selector xpath="." />
      <xsd:field xpath="@default" />
    </xsd:keyref>
    <xsd:keyref name="role initiator" refer="xsf:role name key">
      <xsd:selector xpath="." />
      <xsd:field xpath="@initiator" />
    </xsd:keyref>
    <xsd:keyref name="role_membership" refer="xsf:role_name_key">
      <xsd:selector xpath="./xsf:membership/*" />
      <xsd:field xpath="@memberOf" />
    </xsd:keyref>
  </xsd:element>
  <xsd:element name="role">
    <xsd:complexType>
      <xsd:attribute name="name" type="xsf:xdRoleName" use="required" />
```

```
  </xsd:complexType>
</xsd:element>
<xsd:element name="membership">
  <xsd:complexType>
    <xsd:choice minOccurs="1" maxOccurs="unbounded">
      <xsd:element ref="xsf:getUserNameFromData" />
      <xsd:element ref="xsf:userName" />
      <xsd:element ref="xsf:group" />
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
<xsd:element name="getUserNameFromData">
  <xsd:complexType>
    <xsd:attribute name="dataObject" type="xsd:string" use="optional" />
    <xsd:attribute name="select" type="xsd:string" use="required" />
    <xsd:attribute name="memberOf" type="xsd:string" use="required" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="userName">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsd:string" use="required" />
    <xsd:attribute name="memberOf" type="xsd:string" use="required" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="group">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsd:string" use="required" />
    <xsd:attribute name="memberOf" type="xsd:string" use="required" />
  </xsd:complexType>
</xsd:element>
<!-- hwsWorkflow -->
<xsd:element name="hwsWorkflow">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:location" minOccurs="1" maxOccurs="1" />
      <xsd:element ref="xsf:allowedActions" minOccurs="1" maxOccurs="1" />
      <xsd:element ref="xsf:allowedTasks" minOccurs="0" maxOccurs="1" />
    </xsd:sequence>
    <xsd:attribute name="taskpaneVisible" type="xsf:xdYesNo" />
  </xsd:complexType>
  <xsd:unique name="hws_actiontask_name">
    <xsd:selector xpath="./xsf:allowedActions/xsf:action|./xsf:allowedTasks/xsf:task" />
    <xsd:field xpath="@name" />
  </xsd:unique>
</xsd:element>
<!-- location -->
<xsd:element name="location">
  <xsd:complexType>
    <xsd:attribute name="url" type="xsd:string" use="required" />
  </xsd:complexType>
</xsd:element>
<!-- allowedActions -->
<xsd:element name="allowedActions">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:action" minOccurs="1" maxOccurs="20" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:unique name="hws actionTypeID unique">
    <xsd:selector xpath="./xsf:action" />
    <xsd:field xpath="@actionTypeID" />
  </xsd:unique>
</xsd:element>
<!-- action -->
<xsd:element name="action">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsf:xdHWSname" use="required" />
    <xsd:attribute name="actionTypeID" type="xsd:string" use="required" />
    <xsd:attribute name="canInitiateWorkflow" type="xsf:xdYesNo" use="required" />
```

```
      <xsd:attribute name="caption" type="xsf:xdHWSCaption" use="optional" />
    </xsd:complexType>
  </xsd:element>
  <!-- allowedTasks -->
  <xsd:element name="allowedTasks">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:task" minOccurs="1" maxOccurs="20" />
      </xsd:sequence>
    </xsd:complexType>
    <xsd:unique name="hws taskID unique">
      <xsd:selector xpath="./xsf:task" />
      <xsd:field xpath="@taskTypeID" />
    </xsd:unique>
  </xsd:element>
  <!-- task -->
  <xsd:element name="task">
    <xsd:complexType>
      <xsd:attribute name="name" type="xsf:xdHWSname" use="required" />
      <xsd:attribute name="taskTypeID" type="xsd:string" use="required" />
      <xsd:attribute name="caption" type="xsf:xdHWSCaption" use="optional" />
    </xsd:complexType>
  </xsd:element>
  <!-- fileNew -->
  <xsd:element name="fileNew">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:initialXmlDocument" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="initialXmlDocument">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:customCategory" minOccurs="0" />
      </xsd:sequence>
      <xsd:attribute name="caption" type="xsf:xdTitle" use="required" />
      <xsd:attribute name="href" type="xsf:xdFileName" use="required" />
    </xsd:complexType>
  </xsd:element>
  <!-- customCategory -->
  <xsd:element name="customCategory">
    <xsd:complexType>
      <xsd:attribute name="name" type="xsf:xdTitle" use="required" />
    </xsd:complexType>
  </xsd:element>
  <!-- package -->
  <xsd:element name="package">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:files" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="files">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:file" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="file">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:fileProperties" minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
      <xsd:attribute name="name" type="xsf:xdFileName" use="required" />
    </xsd:complexType>
  </xsd:element>
```

```
<xsd:element name="fileProperties">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:property" minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="property">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsd:string" use="required" />
    <xsd:attribute name="value" type="xsd:string" use="required" />
    <xsd:attribute name="type" type="xsd:QName" use="required" />
  </xsd:complexType>
</xsd:element>
<!-- permissions -->
<xsd:element name="permissions">
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element ref="xsf:allowedControl" />
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
<xsd:element name="allowedControl">
  <xsd:complexType>
    <xsd:attribute name="cabFile" type="xsd:string" use="optional" />
    <xsd:attribute name="clsid" type="xsd:string" use="required" />
    <xsd:attribute name="version" type="xsd:string" use="optional" />
  </xsd:complexType>
</xsd:element>
<!-- View and Context-Driven Editing definitions -->
<!-- External Views -->
<xsd:element name="externalViews">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:externalView" minOccurs="1" maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:attribute name="default" type="xsd:string" />
  </xsd:complexType>
  <xsd:unique name="externalViews_name_unique">
    <xsd:selector xpath="./xsf:externalView" />
    <xsd:field xpath="@default" />
  </xsd:unique>
  <xsd:keyref name="external_views_printView" refer="xsf:externalView_name_key">
    <xsd:selector xpath="." />
    <xsd:field xpath="@default" />
  </xsd:keyref>
</xsd:element>
<xsd:element name="externalView">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:mainpane" />
    </xsd:sequence>
    <xsd:attribute name="target" type="xsd:string" />
    <xsd:attribute name="name" type="xsf:xdViewName" use="required" />
    <xsd:attribute name="designMode" type="xsf:xdDesignMode" />
  </xsd:complexType>
</xsd:element>
<!-- attributeData -->
<xsd:element name="attributeData">
  <xsd:complexType>
    <xsd:attribute name="attribute" type="xsd:string" use="required" />
    <xsd:attribute name="value" type="xsd:string" use="required" />
  </xsd:complexType>
</xsd:element>
<!-- button -->
<xsd:element name="button">
  <xsd:complexType>
    <xsd:attribute name="caption" type="xsf:xdTitle" />
    <xsd:attribute name="icon" type="xsd:string" />
```

```
      <xsd:attribute name="tooltip" type="xsf:xdTitle" />
      <xsd:attribute name="name" type="xsd:NMTOKEN" />
      <xsd:attribute name="xmlToEdit" type="xsd:NMTOKEN" />
      <xsd:attribute name="action">
        <xsd:simpleType>
          <xsd:restriction base="xsd:NMTOKEN">
            <xsd:enumeration value="xCollection::insert" />
            <xsd:enumeration value="xCollection::insertBefore" />
            <xsd:enumeration value="xCollection::insertAfter" />
            <xsd:enumeration value="xCollection::remove" />
            <xsd:enumeration value="xCollection::refreshFilter" />
            <xsd:enumeration value="xCollection::removeAll" />
            <xsd:enumeration value="xOptional::insert" />
            <xsd:enumeration value="xOptional::remove" />
            <xsd:enumeration value="xReplace::replace" />
            <xsd:enumeration value="xFileAttachment::attach" />
            <xsd:enumeration value="xFileAttachment::open" />
            <xsd:enumeration value="xFileAttachment::saveAs" />
            <xsd:enumeration value="xFileAttachment::remove" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="showIf">
        <xsd:simpleType>
          <xsd:restriction base="xsd:NMTOKEN">
            <xsd:enumeration value="always" />
            <xsd:enumeration value="enabled" />
            <xsd:enumeration value="immediate" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:complexType>
  </xsd:element>
  <!-- chooseFragment -->
  <xsd:element name="chooseFragment">
    <xsd:complexType mixed="true">
      <xsd:sequence>
        <xsd:any minOccurs="0" maxOccurs="unbounded" processContents="skip" />
      </xsd:sequence>
      <xsd:attribute name="parent" type="xsd:string" />
      <xsd:attribute name="followingSiblings" type="xsd:string" use="optional" />
      <xsd:attribute name="innerFragment" type="xsd:string" use="optional" />
    </xsd:complexType>
  </xsd:element>
  <!-- editWith -->
  <xsd:element name="editWith">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:masterDetail" minOccurs="0" maxOccurs="1" />
        <xsd:element ref="xsf:fragmentToInsert" minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
      <xsd:attribute name="component" use="required">
        <xsd:simpleType>
          <xsd:restriction base="xsd:NMTOKEN">
            <xsd:enumeration value="xCollection" />
            <xsd:enumeration value="xOptional" />
            <xsd:enumeration value="xReplace" />
            <xsd:enumeration value="xTextList" />
            <xsd:enumeration value="xField" />
            <xsd:enumeration value="xImage" />
            <xsd:enumeration value="xFileAttachment" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="caption" type="xsf:xdTitle" use="optional" />
      <xsd:attribute name="autoComplete" type="xsf:xdYesNo" use="optional" />
      <xsd:attribute name="proofing" type="xsf:xdYesNo" use="optional" />
      <xsd:attribute name="type" use="optional">
        <xsd:simpleType>
```

```
            <xsd:restriction base="xsd:NMTOKEN">
              <xsd:enumeration value="plain" />
              <xsd:enumeration value="formatted" />
              <xsd:enumeration value="plainMultiline" />
              <xsd:enumeration value="formattedMultiline" />
              <xsd:enumeration value="rich" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="useFilter" use="optional">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="yes" />
              <xsd:enumeration value="no" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="widgetIcon" use="optional">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="standard" />
              <xsd:enumeration value="filter" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="filterDependency" type="xsd:string" use="optional" />
        <xsd:attribute name="field" type="xsd:string" use="optional" />
        <xsd:attribute name="removeAncestors" type="xsd:nonNegativeInteger" use="optional" />
        <xsd:attribute name="maxLength" use="optional">
          <xsd:simpleType>
            <xsd:restriction base="xsd:integer">
              <xsd:minInclusive value="-1" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="allowedFileTypes" type="xsd:string" use="optional" />
        <xsd:anyAttribute namespace="http://schemas.microsoft.com/office/infopath/2003"
processContents="skip" />
      </xsd:complexType>
    </xsd:element>
    <!-- unboundControls -->
    <xsd:element name="unboundControls">
      <xsd:complexType>
        <xsd:sequence>
          <!-- button -->
          <xsd:element name="button" minOccurs="0" maxOccurs="unbounded">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element ref="xsf:ruleSetAction" minOccurs="0" maxOccurs="1" />
              </xsd:sequence>
              <xsd:attribute name="name" use="required">
                <xsd:simpleType>
                  <!-- type of name is non qualified name, but NCName also
accepts '.' and '-', so these characters are disabled by               pattern restriction -
->
                  <xsd:restriction base="xsd:NCName">
                    <xsd:pattern value="[^\.\^-]*" />
                  </xsd:restriction>
                </xsd:simpleType>
              </xsd:attribute>
            </xsd:complexType>
            <xsd:keyref name="button_ruleSetAction" refer="xsf:ruleset_name_key">
              <xsd:selector xpath="./xsf:ruleSetAction" />
              <xsd:field xpath="@ruleSet" />
            </xsd:keyref>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
```

```
<!-- editing -->
<xsd:element name="editing">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:xmlToEdit" minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<!-- Master Detail -->
<xsd:element name="masterDetail">
  <xsd:complexType>
    <xsd:attribute name="master" type="xsd:string" />
    <xsd:attribute name="masterViewContext" type="xsd:string" />
    <xsd:attribute name="masterKey" type="xsd:string" />
    <xsd:attribute name="detailKey" type="xsd:string" />
  </xsd:complexType>
</xsd:element>
<!-- fragmentToInsert -->
<xsd:element name="fragmentToInsert">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:chooseFragment" minOccurs="1" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<!-- mainpane -->
<xsd:element name="mainpane">
  <xsd:complexType>
    <xsd:attribute name="transform" type="xsf:xdFileName" use="required" />
  </xsd:complexType>
</xsd:element>
<!-- printSettings -->
<xsd:element name="printSettings">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:header" minOccurs="0" maxOccurs="1" />
      <xsd:element ref="xsf:footer" minOccurs="0" maxOccurs="1" />
    </xsd:sequence>
    <xsd:attribute name="orientation">
      <xsd:simpleType>
        <xsd:restriction base="xsd:NMTOKEN">
          <xsd:enumeration value="portrait" />
          <xsd:enumeration value="landscape" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="header">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="255" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="footer">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="255" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="marginUnitsType">
      <xsd:simpleType>
        <xsd:restriction base="xsd:NMTOKEN">
          <xsd:enumeration value="in" />
          <xsd:enumeration value="cm" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="rightMargin">
```

```
  <xsd:simpleType>
    <xsd:restriction base="xsd:float">
      <xsd:minInclusive value="0" />
      <xsd:maxInclusive value="100" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="leftMargin">
  <xsd:simpleType>
    <xsd:restriction base="xsd:float">
      <xsd:minInclusive value="0" />
      <xsd:maxInclusive value="100" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="topMargin">
  <xsd:simpleType>
    <xsd:restriction base="xsd:float">
      <xsd:minInclusive value="0" />
      <xsd:maxInclusive value="100" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="bottomMargin">
  <xsd:simpleType>
    <xsd:restriction base="xsd:float">
      <xsd:minInclusive value="0" />
      <xsd:maxInclusive value="100" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="printerName">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="255" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="paperSize">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="255" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="paperSource">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="255" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="copies">
  <xsd:simpleType>
    <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="1" />
      <xsd:maxInclusive value="9999" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="collate" type="xsf:xdYesNo" />
<xsd:attribute name="pageRangeStart">
  <xsd:simpleType>
    <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="1" />
      <xsd:maxInclusive value="32000" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
```

```
      <xsd:attribute name="pageRangeEnd">
        <xsd:simpleType>
          <xsd:restriction base="xsd:integer">
            <xsd:minInclusive value="1" />
            <xsd:maxInclusive value="32000" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="printerSpecificSettings">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="255" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="header">
    <xsd:complexType mixed="true">
      <xsd:sequence>
        <xsd:any minOccurs="0" maxOccurs="unbounded" processContents="skip" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="footer">
    <xsd:complexType mixed="true">
      <xsd:sequence>
        <xsd:any minOccurs="0" maxOccurs="unbounded" processContents="skip" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <!-- toolbar -->
  <xsd:element name="toolbar">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:group ref="xsf:UIItem" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
      <xsd:attribute name="name" type="xsf:xdTitle" use="required" />
      <xsd:attribute name="caption" type="xsf:xdTitle" use="required" />
    </xsd:complexType>
  </xsd:element>
  <!-- menu -->
  <xsd:element name="menu">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:group ref="xsf:UIItem" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
      <xsd:attribute name="caption" type="xsf:xdTitle" use="required" />
    </xsd:complexType>
  </xsd:element>
  <!-- menuArea -->
  <xsd:element name="menuArea">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:group ref="xsf:UIItem" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
      <xsd:attribute name="name" use="required">
        <xsd:simpleType>
          <xsd:restriction base="xsd:NMTOKEN">
            <xsd:enumeration value="msoFileMenu" />
            <xsd:enumeration value="msoEditMenu" />
            <xsd:enumeration value="msoInsertMenu" />
            <xsd:enumeration value="msoViewMenu" />
            <xsd:enumeration value="msoFormatMenu" />
            <xsd:enumeration value="msoToolsMenu" />
            <xsd:enumeration value="msoTableMenu" />
            <xsd:enumeration value="msoHelpMenu" />
            <xsd:enumeration value="msoStructuralEditingContextMenu" />
          </xsd:restriction>
```

```
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
</xsd:element>
<!-- UIContainer -->
<xsd:group name="UIContainer">
  <xsd:choice>
    <xsd:element ref="xsf:toolbar" />
    <xsd:element ref="xsf:menu" />
    <xsd:element ref="xsf:menuArea" />
  </xsd:choice>
</xsd:group>
<!-- UIItem -->
<xsd:group name="UIItem">
  <xsd:choice>
    <xsd:element ref="xsf:button" />
    <xsd:element ref="xsf:menu" />
  </xsd:choice>
</xsd:group>
<!-- taskpane -->
<xsd:element name="taskpane">
  <xsd:complexType>
    <xsd:attribute name="caption" type="xsd:string" use="required" />
    <xsd:attribute name="href" type="xsd:string" use="required" />
  </xsd:complexType>
</xsd:element>
<!-- views -->
<xsd:element name="views">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:view" minOccurs="1" maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:attribute name="default" type="xsd:string" />
  </xsd:complexType>
  <xsd:unique name="views_name_unique">
    <xsd:selector xpath="./xsf:view" />
    <xsd:field xpath="@name" />
  </xsd:unique>
  <xsd:keyref name="view_printView" refer="xsf:view_or_externalView_name_key">
    <xsd:selector xpath="./xsf:view" />
    <xsd:field xpath="@printView" />
  </xsd:keyref>
  <xsd:keyref name="views_default" refer="xsf:view_name_key">
    <xsd:selector xpath="." />
    <xsd:field xpath="@default" />
  </xsd:keyref>
</xsd:element>
<!-- ViewContent -->
<xsd:group name="ViewContent">
  <xsd:choice>
    <xsd:element ref="xsf:editing" minOccurs="0" />
    <xsd:element ref="xsf:mainpane" minOccurs="0" />
    <xsd:element ref="xsf:printSettings" minOccurs="0" />
    <xsd:group ref="xsf:UIContainer" minOccurs="0" maxOccurs="unbounded" />
    <xsd:element ref="xsf:unboundControls" minOccurs="0" />
  </xsd:choice>
</xsd:group>
<!-- view -->
<xsd:element name="view">
  <xsd:complexType>
    <xsd:group ref="xsf:ViewContent" minOccurs="0" maxOccurs="unbounded" />
    <xsd:attribute name="caption" type="xsf:xdViewName" />
    <xsd:attribute name="name" type="xsf:xdViewName" use="required" />
    <xsd:attribute name="showMenuItem" type="xsf:xdYesNo" use="optional" />
    <xsd:attribute name="printView" type="xsd:string" />
    <xsd:attribute name="designMode" type="xsf:xdDesignMode" />
  </xsd:complexType>
  <xsd:unique name="toolbar_name_unique">
    <xsd:selector xpath="./xsf:toolbar" />
```

```
      <xsd:field xpath="@name" />
    </xsd:unique>
    <xsd:unique name="menuArea_name_unique">
      <xsd:selector xpath="./xsf:menuArea" />
      <xsd:field xpath="@name" />
    </xsd:unique>
    <xsd:unique name="xmlToEdit_name_unique">
      <xsd:selector xpath="./xsf:editing/xsf:xmlToEdit" />
      <xsd:field xpath="@name" />
    </xsd:unique>
    <xsd:key name="xmlToEdit name key">
      <xsd:selector xpath="./xsf:editing/xsf:xmlToEdit" />
      <xsd:field xpath="@name" />
    </xsd:key>
    <xsd:keyref name="button_xmlToEdit_reference" refer="xsf:xmlToEdit_name_key">
      <xsd:selector xpath="./xsf:menuArea/xsf:button | ./xsf:menu/xsf:button |
./xsf:toolbar/xsf:button" />
      <xsd:field xpath="@xmlToEdit" />
    </xsd:keyref>
  </xsd:element>
  <!-- xmlToEdit -->
  <xsd:element name="xmlToEdit">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:editWith" minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:NMTOKEN" use="required" />
      <xsd:attribute name="item" type="xsd:string" use="required" />
      <xsd:attribute name="container" type="xsd:string" />
      <xsd:attribute name="viewContext">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:pattern value="((\.|\#|[a-zA-Z0-9_])[a-zA-Z0-9_]*)(\s((\.|\#|[a-zA-Z0-
9 ])[a-zA-Z0-9 ]*))*" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:complexType>
  </xsd:element>
  <!-- Digital Signatures -->
  <xsd:element name="documentSignatures">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:signedDataBlock" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
      <xsd:attribute name="signatureLocation" type="xsd:string" use="optional" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="signedDataBlock">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="message" type="xsf:xdSignedDataBlockMessage" minOccurs="0" />
      </xsd:sequence>
      <xsd:attribute name="name" type="xsf:xdSignedDataBlockName" use="required" />
      <xsd:attribute name="data" type="xsd:string" use="required" />
      <xsd:attribute name="signatureLocation" type="xsd:string" use="required" />
      <xsd:attribute name="mode" type="xsf:xdSignatureRelationEnum" use="required" />
    </xsd:complexType>
    <xsd:unique name="signedDataBlock name unique">
      <xsd:selector xpath="." />
      <xsd:field xpath="@name" />
    </xsd:unique>
  </xsd:element>
  <!-- Version Upgrade -->
  <xsd:element name="documentVersionUpgrade">
    <xsd:complexType>
      <xsd:choice>
        <xsd:element ref="xsf:useScriptHandler" />
        <xsd:element ref="xsf:useTransform" />
```

```
        </xsd:choice>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="useTransform">
      <xsd:complexType>
        <xsd:attribute name="transform" use="required">
          <xsd:simpleType>
            <xsd:union memberTypes="xsf:xdFileName xsf:xdEmptyString" />
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="minVersionToUpgrade" type="xsf:xdSolutionVersion" use="required"
/>
        <xsd:attribute name="maxVersionToUpgrade" type="xsf:xdSolutionVersion" />
      </xsd:complexType>
    </xsd:element>
    <!-- XSF Extensions -->
    <xsd:element name="extensions">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="xsf:extension" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="extension">
      <xsd:complexType mixed="true">
        <xsd:sequence>
          <xsd:any minOccurs="0" maxOccurs="unbounded" processContents="lax" />
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:NMTOKEN" use="required" />
        <xsd:anyAttribute processContents="skip" />
      </xsd:complexType>
    </xsd:element>
    <!-- Rules -->
    <xsd:element name="ruleSetAction">
      <xsd:complexType>
        <xsd:attribute name="ruleSet" type="xsd:string" use="required" />
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="rule">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="xsf:dialogBoxMessageAction" />
            <xsd:element ref="xsf:dialogBoxExpressionAction" />
            <xsd:element ref="xsf:switchViewAction" />
            <xsd:element ref="xsf:assignmentAction" />
            <xsd:element ref="xsf:queryAction" />
            <xsd:element name="submitAction">
              <xsd:complexType>
                <xsd:attribute name="adapter" type="xsf:xdTitle" use="required" />
              </xsd:complexType>
            </xsd:element>
            <xsd:element ref="xsf:openNewDocumentAction" />
            <xsd:element ref="xsf:closeDocumentAction" />
          </xsd:choice>
          <xsd:element name="exitRuleSet" minOccurs="0">
            <xsd:complexType />
          </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="caption" type="xsd:string" use="required" />
        <xsd:attribute name="condition" type="xsd:string" use="optional" />
        <xsd:attribute name="isEnabled" type="xsf:xdYesNo" use="optional" default="yes" />
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="dialogBoxMessageAction">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="1024" />
        </xsd:restriction>
```

```
  </xsd:simpleType>
</xsd:element>
<xsd:element name="dialogBoxExpressionAction" type="xsd:string" />
<xsd:element name="switchViewAction">
  <xsd:complexType>
    <xsd:attribute name="view" type="xsf:xdViewName" use="required" />
  </xsd:complexType>
  <xsd:keyref name="switchViewAction_view_keyref" refer="xsf:view_name_key">
    <xsd:selector xpath="." />
    <xsd:field xpath="@view" />
  </xsd:keyref>
</xsd:element>
<xsd:element name="assignmentAction">
  <xsd:complexType>
    <xsd:attribute name="targetField" type="xsd:string" use="required" />
    <xsd:attribute name="expression" type="xsd:string" use="required" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="queryAction">
  <xsd:complexType>
    <xsd:attribute name="adapter" type="xsd:string" use="required" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="openNewDocumentAction">
  <xsd:complexType>
    <xsd:attribute name="solutionURI" type="xsd:anyURI" use="required" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="closeDocumentAction">
  <xsd:complexType>
    <xsd:attribute name="promptToSaveChanges" type="xsf:xdYesNo" use="required" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="ruleSet">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:rule" minOccurs="1" maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="ruleSets">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:ruleSet" minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:unique name="ruleSets name unique">
    <xsd:selector xpath="./xsf:ruleSet" />
    <xsd:field xpath="@name" />
  </xsd:unique>
</xsd:element>
<!-- Declarative Calculations -->
<xsd:element name="calculations">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:calculatedField" minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:attribute name="treatBlankValueAsZero" type="xsf:xdYesNo" use="optional" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="calculatedField">
  <xsd:complexType>
    <xsd:attribute name="target" type="xsd:string" use="required" />
    <xsd:attribute name="expression" type="xsd:string" use="required" />
    <xsd:attribute name="refresh" type="xsd:string" use="required" />
  </xsd:complexType>
</xsd:element>
```

```
</xsd:schema>
```

5.2 The InfoPath XSF2 XSD

The XML schema for the form definition (.xsf) file can also be found at the location described by [MSDN-XSF].

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsf="http://schemas.microsoft.com/office/infopath/2003/solutionDefinition"

xmlns:xsf2="http://schemas.microsoft.com/office/infopath/2006/solutionDefinition/extensions"

targetNamespace="http://schemas.microsoft.com/office/infopath/2006/solutionDefinition/extensi
ons"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified">
  <xsd:import
namespace="http://schemas.microsoft.com/office/infopath/2003/solutionDefinition"
schemaLocation="xsfschema.xsd" />

  <!--
      Please note that any changes to this schema requires the documented xsd       to be
updated as well.
      It is more important because some of the types are open and do not list       all the
expected values and the documentation is expected list them.
   -->

  <!-- Allowed values: submit, print, view, save, saveAs, close, refresh -->
  <xsd:simpleType name="serverCommandActionType">
    <xsd:restriction base="xsd:NMTOKEN">
      <xsd:pattern value="[a-zA-Z0-9 ]*" />
    </xsd:restriction>
  </xsd:simpleType>
  <!-- Allowed values: none, xml, xmlXsn -->
  <xsd:simpleType name="emailAttachmentType">
    <xsd:restriction base="xsd:NMTOKEN">
      <xsd:pattern value="[a-zA-Z0-9 ]*" />
    </xsd:restriction>
  </xsd:simpleType>
  <!-- Allowed values: client, server -->
  <xsd:simpleType name="compatibilityModesType">
    <xsd:restriction base="xsd:NMTOKEN">
      <xsd:pattern value="[a-zA-Z0-9_]*" />
    </xsd:restriction>
  </xsd:simpleType>
  <!-- Allowed values: templatePart, formTemplate -->
  <xsd:simpleType name="solutionType">
    <xsd:restriction base="xsd:NMTOKEN">
      <xsd:pattern value="[a-zA-Z0-9]*" />
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="formDescriptionType">
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="1024" />
      <xsd:minLength value="1" />
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="formLocaleType">
    <xsd:restriction base="xsd:token">
      <xsd:minLength value="1" />
    </xsd:restriction>
  </xsd:simpleType>
  <!-- Allowed values: CSharp, VisualBasic -->
  <xsd:simpleType name="managedCodeType">
```

```
      <xsd:restriction base="xsd:string">
        <xsd:pattern value="[a-zA-Z0-9\.]*" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:attributeGroup name="queryKeyFile">
      <xsd:attribute name="queryFile" type="xsd:string" use="optional" />
      <xsd:attribute name="queryKey" type="xsd:string" use="optional" />
    </xsd:attributeGroup>

    <!-- Root element for the xsf extension elements -->
    <xsd:element name="solutionDefinition">
      <xsd:complexType>
        <xsd:all>
          <xsd:element ref="xsf2:server" minOccurs="0" />
          <xsd:element ref="xsf2:solutionPropertiesExtension" minOccurs="0" />
          <xsd:element ref="xsf2:mergedPrintView" minOccurs="0" />
          <xsd:element ref="xsf2:offline" minOccurs="0" />
          <xsd:element ref="xsf2:listPropertiesExtension" minOccurs="0" />
          <xsd:element ref="xsf2:dataConnections" minOccurs="0" />
          <xsd:element ref="xsf2:sendByMail" minOccurs="0" />
          <xsd:element ref="xsf2:warnings" minOccurs="0" />
          <xsd:element ref="xsf2:viewsExtension" minOccurs="0" />
          <xsd:element ref="xsf2:preview" minOccurs="0" />
          <xsd:element ref="xsf2:autoUpdatePrompt" minOccurs="0" />
          <xsd:element ref="xsf2:inputScopes" minOccurs="0" />
          <xsd:element ref="xsf2:managedCode" minOccurs="0" />
          <xsd:element ref="xsf2:submit" minOccurs="0" />
          <xsd:element ref="xsf2:featureRestrictionsExtension" minOccurs="0" />
        </xsd:all>
        <xsd:attribute name="runtimeCompatibility" use="required">
          <xsd:simpleType>
            <xsd:list itemType="xsf2:compatibilityModesType"/>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="solutionType" type="xsf2:solutionType" use="optional" />
        <xsd:attribute name="description" type="xsf2:formDescriptionType" use="optional" />
        <xsd:attribute name="allowClientOnlyCode" type="xsf:xdYesNo" use="optional"
  default="no" />
        <xsd:attribute name="runtimeCompatibilityURL" type="xsd:string" use="optional"/>
        <xsd:attribute name="verifyOnServer" type="xsf:xdYesNo" use="optional"/>
        <xsd:anyAttribute processContents="skip" />
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="server">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="xsf2:toolbar" minOccurs="0" />
        </xsd:sequence>
        <xsd:attribute name="formLocale" type="xsf2:formLocaleType" use="required" />
        <xsd:attribute name="isPreSubmitPostBackEnabled" type="xsf:xdYesNo" use="optional" />
        <xsd:attribute name="isMobileEnabled" type="xsf:xdYesNo" use="optional" />
        <xsd:anyAttribute processContents="skip" />
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="toolbar">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="xsf2:commands" minOccurs="0" />
        </xsd:sequence>
        <xsd:attribute name="enabledTop" type="xsf:xdYesNo" use="optional" default="no" />
        <xsd:attribute name="enabledBottom" type="xsf:xdYesNo" use="optional" default="no" />
        <xsd:anyAttribute  processContents="skip" />
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="commands">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="xsf2:command" maxOccurs="unbounded" minOccurs="0"/>
```

```
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="command">
  <xsd:complexType>
    <xsd:sequence />
    <xsd:attribute name="action" type="xsf2:serverCommandActionType" use="required" />
    <xsd:attribute name="caption" type="xsf:xdTitle" use="optional" />
  </xsd:complexType>
</xsd:element>

<xsd:element name="solutionPropertiesExtension">
  <xsd:complexType>
    <xsd:all>
      <xsd:element ref="xsf2:install" minOccurs="0" />
      <xsd:element ref="xsf2:wss" minOccurs="0" />
      <xsd:element ref="xsf2:contentType" minOccurs="0" />
      <xsd:element ref="xsf2:share" minOccurs="0" />
      <xsd:element ref="xsf2:mail" minOccurs="0" />
      <xsd:element ref="xsf2:admin" minOccurs="0" />
      <xsd:element ref="xsf2:contentTypeTemplate" minOccurs="0" />
    </xsd:all>
    <xsd:attribute name="branch" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="install" />
          <xsd:enumeration value="wss" />
          <xsd:enumeration value="contentType" />
          <xsd:enumeration value="share" />
          <xsd:enumeration value="mail" />
          <xsd:enumeration value="admin" />
          <xsd:enumeration value="contentTypeTemplate" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:anyAttribute  processContents="skip" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="install">
  <xsd:complexType>
    <xsd:attribute name="companyName" type="xsd:string" use="required" />
    <xsd:attribute name="language" type="xsd:string" use="required" />
    <xsd:attribute name="path" type="xsd:string" use="required" />
    <xsd:attribute name="updatePath" type="xsd:string" use="optional" />
    <xsd:anyAttribute  processContents="skip" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="wss">
  <xsd:complexType>
    <xsd:sequence />
    <xsd:attribute name="path" type="xsd:string" use="required" />
    <xsd:attribute name="name" type="xsd:string" use="required" />
    <xsd:attribute name="description" type="xsd:string" use="required" />
    <xsd:attribute name="browserEnable" type="xsf:xdYesNo" use="optional" />
    <xsd:anyAttribute  processContents="skip" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="contentType">
  <xsd:complexType>
    <xsd:sequence />
    <xsd:attribute name="path" type="xsd:string" use="required" />
    <xsd:attribute name="sharepointContentTypeId" type="xsd:string" use="required" />
    <xsd:anyAttribute  processContents="skip" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="contentTypeTemplate">
  <xsd:complexType>
    <xsd:sequence />
    <xsd:attribute name="site" type="xsd:string" use="required" />
```

```
      <xsd:attribute name="path" type="xsd:string" use="required" />
      <xsd:attribute name="name" type="xsd:string" use="required" />
      <xsd:attribute name="description" type="xsd:string" use="required" />
      <xsd:attribute name="browserEnable" type="xsf:xdYesNo" use="optional" />
      <xsd:anyAttribute  processContents="skip" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="share">
    <xsd:complexType>
      <xsd:sequence />
      <xsd:attribute name="formName" type="xsd:string" use="required" />
      <xsd:attribute name="path" type="xsd:string" use="required" />
      <xsd:attribute name="accessPath" type="xsd:string" use="required" />
      <xsd:anyAttribute  processContents="skip" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="mail">
    <xsd:complexType>
      <xsd:sequence />
      <xsd:attribute name="formName" type="xsd:string" use="required" />
      <xsd:anyAttribute  processContents="skip" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="admin">
    <xsd:complexType>
      <xsd:sequence />
      <xsd:attribute name="path" type="xsd:string" use="required" />
      <xsd:attribute name="site" type="xsd:string" use="required" />
      <xsd:anyAttribute  processContents="skip" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="mergedPrintView">
    <xsd:complexType>
      <xsd:all>
        <xsd:element ref="xsf:printSettings" minOccurs="0" />
        <xsd:element ref="xsf2:includedViews" minOccurs="0" />
      </xsd:all>
      <xsd:attribute name="isDefault" type="xsf:xdYesNo" use="optional" default="no" />
      <xsd:attribute name="isCustomizable" type="xsf:xdYesNo" use="optional" default="no" />
      <xsd:attribute name="viewBreak" type="xsd:string" use="required" />
      <xsd:anyAttribute  processContents="skip" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="includedViews">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf2:includedView" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="includedView">
    <xsd:complexType>
      <xsd:sequence />
      <xsd:attribute name="name" type="xsf:xdViewName" use="required" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="offline">
    <xsd:complexType>
      <xsd:attribute name="openIfQueryFails" type="xsf:xdYesNo" default="no" use="optional"
/>
      <xsd:attribute name="cacheQueries" type="xsf:xdYesNo" default="no" use="optional" />
      <xsd:attribute name="expirationTime" type="xsd:nonNegativeInteger" use="optional" />
      <xsd:anyAttribute  processContents="skip" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="listPropertiesExtension">
```

```
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="xsf2:fieldsExtension" minOccurs="0"></xsd:element>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="fieldsExtension">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="xsf2:fieldExtension" maxOccurs="unbounded" minOccurs="1" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="fieldExtension">
      <xsd:complexType>
        <xsd:sequence>
        </xsd:sequence>
        <xsd:attribute name="columnName" type="xsd:string" use="required" />
        <xsd:attribute name="readWrite" type="xsf:xdYesNo" use="optional" default="no" />
        <xsd:attribute name="columnId" type="xsd:string" use="optional" />
        <xsd:anyAttribute  processContents="skip" />
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="dataConnections">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="xsf2:useHttpHandlerExtension" minOccurs="0" />
          <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="xsf2:davAdapterExtension" minOccurs="0" maxOccurs="unbounded" />
            <xsd:element ref="xsf2:adoAdapterExtension" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="xsf2:webServiceAdapterExtension" minOccurs="0"
maxOccurs="unbounded" />
            <xsd:element ref="xsf2:emailAdapterExtension" minOccurs="0" maxOccurs="unbounded"
/>
            <xsd:element ref="xsf2:xmlFileAdapterExtension" minOccurs="0" maxOccurs="unbounded"
/>
            <xsd:element ref="xsf2:sharepointListAdapterExtension" minOccurs="0"
maxOccurs="unbounded" />
          </xsd:choice>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="useHttpHandlerExtension">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="xsf2:connectoid" minOccurs="0" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="connectoid">
      <xsd:complexType>
        <xsd:sequence />
        <xsd:attribute name="name" type="xsd:string" use="required" />
        <xsd:attribute name="siteCollection" type="xsd:string" use="required" />
        <xsd:attribute name="source" type="xsd:string" use="required" />
        <xsd:attribute name="connectionLinkType" type="xsd:string" use="required" />
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="davAdapterExtension">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="xsf2:connectoid" minOccurs="0" />
        </xsd:sequence>
        <xsd:attribute name="ref" type="xsf:xdTitle" use="required"></xsd:attribute>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="adoAdapterExtension">
      <xsd:complexType>
        <xsd:sequence>
```

```
        <xsd:element ref="xsf2:connectoid" minOccurs="0" />
      </xsd:sequence>
      <xsd:attribute name="ref" type="xsf:xdTitle" use="required" />
      <xsd:attribute name="submitAdapterName" type="xsf:xdTitle" use="optional" />
      <xsd:attributeGroup ref="xsf2:queryKeyFile" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="webServiceAdapterExtension">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf2:connectoid" minOccurs="0" />
        <xsd:element ref="xsf2:relativeQuery" minOccurs="0" />
      </xsd:sequence>
      <xsd:attribute name="trackDataSetChanges" type="xsf:xdYesNo" use="optional"
default="no" />
      <xsd:attribute name="ref" type="xsf:xdTitle" use="required" />
      <xsd:attributeGroup ref="xsf2:queryKeyFile" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="relativeQuery">
    <xsd:complexType>
      <xsd:sequence />
      <xsd:attribute name="replace" type="xsd:string" use="required" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="emailAdapterExtension">
    <xsd:complexType>
      <xsd:sequence />
      <xsd:attribute name="ref" type="xsf:xdTitle" use="required" />
      <xsd:attribute name="emailAttachmentType" type="xsf2:emailAttachmentType"
use="required" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="xmlFileAdapterExtension">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf2:connectoid" minOccurs="0" />
      </xsd:sequence>
      <xsd:attribute name="ref" type="xsf:xdTitle" use="required" />
      <xsd:attributeGroup ref="xsf2:queryKeyFile" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="sharepointListAdapterExtension">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf2:connectoid" minOccurs="0" />
      </xsd:sequence>
      <xsd:attribute name="ref" type="xsf:xdTitle" use="required" />
      <xsd:attribute name="queryThisFormOnly" type="xsf:xdYesNo" use="optional" default="no"
/>
      <xsd:attribute name="sharepointWebGuid" type="xsd:string" use="optional" />
      <xsd:attributeGroup ref="xsf2:queryKeyFile" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="sendByMail">
    <xsd:complexType>
      <xsd:sequence />
      <xsd:attribute name="emailAttachmentType" type="xsf2:emailAttachmentType"
use="optional" />
      <xsd:attribute name="disableEmailForms" type="xsf:xdYesNo" use="optional" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="warnings">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf2:warning" maxOccurs="unbounded" minOccurs="0" />
      </xsd:sequence>
```

```
        <xsd:anyAttribute  processContents="skip" />
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="warning">
      <xsd:complexType>
        <xsd:sequence />
        <xsd:attribute name="source" type="xsd:string" use="required" />
        <xsd:attribute name="hidden" type="xsf:xdYesNo" use="optional" default="no" />
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="viewsExtension">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="xsf2:viewExtension" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="viewExtension">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="xsf2:xmlToEditExtension" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute name="ref" type="xsf:xdTitle" use="required" />
        <xsd:attribute name="designMode" type="xsd:string" use="optional" />
        <xsd:attribute name="readOnly" type="xsf:xdYesNo" use="optional" />
        <xsd:attribute name="clientOnly" type="xsf:xdYesNo" use="optional" default="no"/>
        <xsd:anyAttribute  processContents="skip" />
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="xmlToEditExtension">
      <xsd:complexType>
        <xsd:sequence/>
        <xsd:attribute name="ref" type="xsf:xdTitle" use="required" />
        <xsd:attribute name="excludeEmbeddedImages" type="xsf:xdYesNo" use="optional"
  default="no" />
        <xsd:attribute name="allowLinkedImages" type="xsf:xdYesNo" use="optional" default="no"
  />
        <xsd:anyAttribute  processContents="skip" />
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="preview">
      <xsd:complexType>
        <xsd:sequence />
        <xsd:attribute name="sampleData" type="xsd:string" use="optional" />
        <xsd:attribute name="domain" type="xsd:string" use="optional" />
        <xsd:attribute name="userRole" type="xsd:string" use="optional" />
        <xsd:anyAttribute  processContents="skip" />
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="autoUpdatePrompt">
      <xsd:complexType>
        <xsd:sequence />
        <xsd:attribute name="showPrompt" type="xsf:xdYesNo" use="optional" />
        <xsd:anyAttribute  processContents="skip" />
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="inputScopes">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="xsf2:inputScope" maxOccurs="unbounded" minOccurs="0" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="inputScope">
      <xsd:complexType>
```

```
    <xsd:sequence>
      <xsd:element ref="xsf2:words" maxOccurs="unbounded" minOccurs="0" />
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required" />
    <xsd:attribute name="caption" type="xsf:xdTitle" use="optional" />
    <xsd:attribute name="expression" type="xsd:string" use="optional" />
    <xsd:anyAttribute  processContents="skip" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="words">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="word" maxOccurs="unbounded" minOccurs="1">
        <xsd:complexType>
          <xsd:sequence />
          <xsd:attribute name="value" type="xsd:string" use="optional" default="" />
          <xsd:anyAttribute  processContents="skip" />
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:anyAttribute  processContents="skip" />
  </xsd:complexType>
</xsd:element>

<xsd:element name="managedCode">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="projectPath" type="xsd:string" use="optional" />
    <xsd:attribute name="language" type="xsf2:managedCodeType" use="required" />
    <xsd:attribute name="version" type="xsd:string" use="required" />
    <xsd:attribute name="enabled" type="xsf:xdYesNo" use="optional" />
  </xsd:complexType>
</xsd:element>

<xsd:element name="submit">
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="submitAction" minOccurs="0">
        <xsd:complexType>
          <xsd:sequence />
          <xsd:attribute name="adapter" type="xsf:xdTitle" use="required" />
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="successMessage" type="xsd:string" minOccurs="0" />
      <xsd:element name="errorMessage" type="xsd:string" minOccurs="0" />
    </xsd:all>
    <xsd:attribute name="caption" type="xsd:string" use="optional" />
    <xsd:attribute name="onAfterSubmit" use="optional">
      <xsd:simpleType>
        <xsd:restriction base="xsd:NMTOKEN">
          <xsd:enumeration value="close" />
          <xsd:enumeration value="keepOpen" />
          <xsd:enumeration value="openNew" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="showStatusDialog" type="xsf:xdYesNo" use="optional" />
    <xsd:attribute name="showSignatureReminder" type="xsf:xdYesNo" use="optional" />
    <xsd:attribute name="disableMenuItem" type="xsf:xdYesNo" use="optional" />
  </xsd:complexType>
</xsd:element>

<xsd:element name="featureRestrictionsExtension">
  <xsd:complexType>
    <xsd:all>
      <xsd:element ref="xsf2:exportToPDForXPS" minOccurs="0" />
    </xsd:all>
  </xsd:complexType>
</xsd:element>
```

```
    <xsd:element name="exportToPDForXPS">
      <xsd:complexType>
        <xsd:attribute name="ui" type="xsf:xdEnabledDisabled" use="required" />
      </xsd:complexType>
    </xsd:element>

  </xsd:schema>
```

6 Appendix B: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs.

- Microsoft Office SharePoint Server 2007
- Microsoft Office InfoPath 2007

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 2.2.20: The value "12.0.0.0" specifies that the form template was created with Office InfoPath 2007. The value "11.0.0.0" specifies that the form template was created with Microsoft Office InfoPath 2003 or Microsoft Office InfoPath 2003 Service Pack 1.

<2> Section 2.2.20: The value "2.0.0.0" specifies that the form template is compatible with Office InfoPath 2007. The value "1.100.0.0" specifies that the form template is compatible with Office InfoPath 2003 SP1. The value "1.0.0.0" specifies that the form template is compatible with Office InfoPath 2003.

<3> Section 2.2.106: Office InfoPath 2007 has the following behavior: The value for this property is interpreted as "immediate".

<4> Section 2.4.1.1: Office InfoPath 2007 has the following behavior: The maximum value allowed for this property is 9999.

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

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

X