

# [MS-IPFF]: InfoPath Form Template Format

---

## Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft's Open Specification Promise (available here: <http://www.microsoft.com/interop/osp>) or the Community Promise (available here: <http://www.microsoft.com/interop/cp/default.aspx>). If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting [iplq@microsoft.com](mailto:iplq@microsoft.com).
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

**Reservation of Rights.** All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

**Tools.** The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

## Revision Summary

Date	Revision History	Revision Class	Comments
04/04/2008	0.01	Major	Initial Availability
06/27/2008	1.0	Minor	Revised and edited technical content
07/13/2009	1.01	Major	Revised and edited the technical content
08/28/2009	1.02	Editorial	Revised and edited the technical content
11/06/2009	1.03	Editorial	Revised and edited the technical content
02/19/2010	2.0	Minor	Updated the technical content
03/31/2010	2.01	Editorial	Revised and edited the technical content
04/30/2010	2.02	Editorial	Revised and edited the technical content
06/07/2010	2.03	Editorial	Revised and edited the technical content
06/29/2010	2.04	Editorial	Changed language and formatting in the technical content.
07/23/2010	2.05	Minor	Clarified the meaning of the technical content.
09/27/2010	2.05	No change	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	2.05	No change	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	2.05	No change	No changes to the meaning, language, or formatting of the technical content.
03/18/2011	2.05	No change	No changes to the meaning, language, or formatting of the technical content.
06/10/2011	2.05	No change	No changes to the meaning, language, or formatting of the technical content.

# Table of Contents

<b>1</b>	<b>Introduction .....</b>	<b>11</b>
1.1	Glossary .....	11
1.2	References.....	12
1.2.1	Normative References.....	12
1.2.2	Informative References .....	14
1.3	Structure Overview (Synopsis) .....	14
1.3.1	The InfoPath Form Template Format.....	14
1.3.2	Form Definition File (XSF) .....	15
1.3.3	XML Schema Files (XSD) .....	15
1.3.4	Form View Files (XSLT) .....	16
1.3.5	Print View Files (XSLT) .....	21
1.3.6	Submit Files (XML) .....	21
1.3.7	Template.XML File .....	22
1.3.8	Upgrade.XSL File.....	22
1.3.9	Resource Files.....	22
1.3.10	Unused Files .....	22
1.4	Relationship to Protocols and Other Structures .....	23
1.5	Applicability Statement.....	23
1.6	Versioning and Localization .....	23
1.7	Vendor-Extensible Fields.....	23
<b>2</b>	<b>Structures .....</b>	<b>24</b>
2.1	The InfoPath Form Template Format .....	24
2.1.1	manifest.xsf .....	24
2.1.2	primaryschema.xsd .....	24
2.1.3	view.xsl .....	24
2.1.4	sampledata.xml .....	24
2.1.5	template.xml .....	24
2.1.6	secondaryschema.xsd .....	24
2.1.7	submitdata.xml .....	24
2.1.8	upgrade.xsl .....	25
2.1.9	merge.xsl.....	25
2.1.10	secondaryschema_offline.xml .....	25
2.1.11	Business Object.....	25
2.1.12	script.js.....	25
2.1.13	script.vbs .....	25
2.1.14	importerrors.xml .....	25
2.1.15	irm_template .....	25
2.1.16	Resource Files.....	25
2.2	Form Definition File (XSF) Specification .....	25
2.2.1	xdTitle .....	31
2.2.2	xdViewName.....	32
2.2.3	xdRoleName .....	33
2.2.4	xdYesNo.....	33
2.2.5	xdEnabledDisabled .....	36
2.2.6	xdManualAuto .....	36
2.2.7	xdExpressionLiteral .....	37
2.2.8	xdFileName .....	37
2.2.9	xdScriptLanguage.....	38
2.2.10	xdSolutionVersion.....	38

2.2.11	xdEmptyString .....	39
2.2.12	xdErrorMessage.....	39
2.2.13	xdDesignMode.....	39
2.2.14	xdTrustLevel .....	40
2.2.15	xdSignedDataBlockName .....	40
2.2.16	xdSignedDataBlockMessage .....	41
2.2.17	xdSignatureRelationEnum.....	41
2.2.18	xdHWSname .....	42
2.2.19	xdHWSCaption .....	42
2.2.20	xDocumentClass.....	42
2.2.21	schemaErrorMessages .....	46
2.2.22	override .....	46
2.2.23	applicationParameters .....	47
2.2.24	solutionProperties.....	47
2.2.25	featureRestrictions.....	48
2.2.26	save .....	49
2.2.27	exportToWeb .....	50
2.2.28	exportToExcel .....	50
2.2.29	print .....	50
2.2.30	sendMail.....	51
2.2.31	autoRecovery .....	51
2.2.32	query .....	52
2.2.33	scripts.....	52
2.2.34	script .....	53
2.2.35	dataObjects .....	53
2.2.36	dataObject.....	54
2.2.37	query .....	55
2.2.38	adoAdapter .....	56
2.2.39	webServiceAdapter .....	56
2.2.40	hwsAdapter .....	57
2.2.41	operation.....	58
2.2.42	hwsOperation.....	59
2.2.43	input.....	60
2.2.44	partFragment.....	60
2.2.45	xmlFileAdapter .....	61
2.2.46	sharepointListAdapter .....	62
2.2.47	field .....	63
2.2.48	davAdapter .....	63
2.2.49	folderURL .....	64
2.2.50	fileName.....	65
2.2.51	emailAdapter .....	65
2.2.52	to .....	67
2.2.53	cc .....	68
2.2.54	bcc .....	68
2.2.55	subject.....	69
2.2.56	intro .....	69
2.2.57	attachmentFileName .....	70
2.2.58	submitToHostAdapter.....	70
2.2.59	dataAdapters .....	71
2.2.60	documentSchemas .....	72
2.2.61	documentSchema .....	72
2.2.62	customValidation .....	73
2.2.63	errorCondition .....	73

2.2.64	errorMessage	74
2.2.65	domEventHandlers	75
2.2.66	domEventHandler	76
2.2.67	importParameters	77
2.2.68	importSource	77
2.2.69	listProperties	78
2.2.70	fields	78
2.2.71	field	79
2.2.72	submit	81
2.2.73	submitAction	83
2.2.74	successMessage	83
2.2.75	errorMessage	84
2.2.76	useHttpHandler	84
2.2.77	useScriptHandler	85
2.2.78	useQueryAdapter	85
2.2.79	onLoad	85
2.2.80	save	86
2.2.81	roles	86
2.2.82	role	87
2.2.83	membership	88
2.2.84	getUserNameFromData	88
2.2.85	userName	89
2.2.86	group	89
2.2.87	hwsWorkflow	90
2.2.88	location	90
2.2.89	allowedActions	91
2.2.90	action	91
2.2.91	allowedTasks	92
2.2.92	task	92
2.2.93	fileNew	93
2.2.94	initialXmlDocument	93
2.2.95	customCategory	94
2.2.96	package	94
2.2.97	files	95
2.2.98	file	95
2.2.99	fileProperties	96
2.2.100	property	96
2.2.101	permissions	98
2.2.102	allowedControl	99
2.2.103	externalViews	99
2.2.104	externalView	100
2.2.105	attributeData	101
2.2.106	button	101
2.2.107	chooseFragment	103
2.2.108	editWith	104
2.2.109	unboundControls	106
2.2.110	button	107
2.2.111	editing	108
2.2.112	masterDetail	108
2.2.113	fragmentToInsert	109
2.2.114	mainpane	109
2.2.115	printSettings	110
2.2.116	header	113

2.2.117	footer.....	113
2.2.118	toolbar .....	114
2.2.119	menu .....	114
2.2.120	menuArea .....	115
2.2.121	taskpane .....	116
2.2.122	views .....	117
2.2.123	view.....	118
2.2.124	xmlToEdit .....	119
2.2.125	documentSignatures .....	120
2.2.126	signedDataBlock .....	121
2.2.127	message.....	122
2.2.128	documentVersionUpgrade .....	122
2.2.129	useTransform .....	122
2.2.130	extensions .....	123
2.2.131	extension.....	124
2.2.132	ruleSetAction .....	124
2.2.133	rule.....	125
2.2.134	submitAction .....	126
2.2.135	exitRuleSet .....	127
2.2.136	dialogBoxMessageAction .....	127
2.2.137	dialogBoxExpressionAction .....	127
2.2.138	switchViewAction .....	128
2.2.139	assignmentAction .....	128
2.2.140	queryAction .....	129
2.2.141	openNewDocumentAction .....	129
2.2.142	closeDocumentAction .....	129
2.2.143	ruleSet .....	130
2.2.144	ruleSets.....	130
2.2.145	calculations.....	131
2.2.146	calculatedField.....	132
2.2.147	Form Definition File (XSF) Extension Specification.....	132
2.2.147.1	serverCommandActionType .....	134
2.2.147.2	emailAttachmentType .....	135
2.2.147.3	compatibilityModesType .....	135
2.2.147.4	solutionType.....	135
2.2.147.5	formDescriptionType .....	136
2.2.147.6	formLocaleType .....	136
2.2.147.7	managedCodeType.....	136
2.2.147.8	solutionDefinition .....	137
2.2.147.9	server.....	138
2.2.147.10	toolbar.....	139
2.2.147.11	commands .....	140
2.2.147.12	command .....	140
2.2.147.13	solutionPropertiesExtension .....	141
2.2.147.14	install .....	142
2.2.147.15	wss .....	143
2.2.147.16	contentType .....	143
2.2.147.17	contentTypeTemplate .....	144
2.2.147.18	share .....	144
2.2.147.19	mail.....	145
2.2.147.20	admin .....	145
2.2.147.21	mergedPrintView .....	146
2.2.147.22	includedViews .....	146

2.2.147.23	includedView .....	147
2.2.147.24	offline .....	147
2.2.147.25	listPropertiesExtension .....	148
2.2.147.26	fieldsExtension .....	148
2.2.147.27	fieldExtension .....	149
2.2.147.28	dataConnections .....	149
2.2.147.29	useHttpHandlerExtension .....	150
2.2.147.30	connectoid.....	151
2.2.147.31	davAdapterExtension .....	152
2.2.147.32	adoAdapterExtension .....	152
2.2.147.33	webServiceAdapterExtension .....	153
2.2.147.34	relativeQuery .....	154
2.2.147.35	emailAdapterExtension.....	155
2.2.147.36	xmlFileAdapterExtension .....	155
2.2.147.37	sharepointListAdapterExtension.....	156
2.2.147.38	sendByMail .....	157
2.2.147.39	warnings .....	157
2.2.147.40	warning.....	158
2.2.147.41	viewsExtension .....	158
2.2.147.42	viewExtension .....	159
2.2.147.43	xmlToEditExtension .....	160
2.2.147.44	preview .....	160
2.2.147.45	autoUpdatePrompt .....	161
2.2.147.46	inputScopes.....	161
2.2.147.47	inputScope .....	162
2.2.147.48	words .....	162
2.2.147.49	word .....	163
2.2.147.50	managedCode.....	163
2.2.147.51	submit .....	164
2.2.147.52	submitAction .....	165
2.2.147.53	successMessage .....	166
2.2.147.54	errorMessage.....	166
2.2.147.55	featureRestrictionsExtension .....	166
2.2.147.56	exportToPDFForXPS .....	167
2.3	XML Schema Files (XSD) Specification .....	167
2.3.1	Control Representation .....	167
2.3.1.1	Button Control.....	168
2.3.1.2	Check Box Control .....	168
2.3.1.3	Contact Selector Control .....	169
2.3.1.4	Date Picker Control .....	169
2.3.1.5	Drop-Down List Control .....	170
2.3.1.6	Expression Box Control.....	170
2.3.1.7	File Attachment Control .....	171
2.3.1.8	Hyperlink Control.....	171
2.3.1.9	List Box Control .....	171
2.3.1.10	Option Button Control.....	172
2.3.1.11	Repeating Section Control .....	173
2.3.1.12	Repeating Table Control.....	173
2.3.1.13	Rich Text Box Control .....	174
2.3.1.14	Section Control and Optional Section Control.....	174
2.3.1.15	Table Control.....	175
2.3.1.16	Text Box Control .....	175
2.4	Form View Files (XSLT) Specification .....	175

2.4.1	View Representation .....	176
2.4.1.1	View Syntax.....	177
2.4.1.2	XSL Root Template .....	181
2.4.1.3	XSL Root Template Style Sheets.....	183
2.4.1.4	Control Data Formatting .....	222
2.4.1.5	Button Control.....	223
2.4.1.6	Check Box Control .....	230
2.4.1.7	Contact Selector Control.....	232
2.4.1.8	Date Picker Control .....	236
2.4.1.9	Drop-Down List Control .....	245
2.4.1.10	Expression Box Control .....	255
2.4.1.11	File Attachment Control .....	258
2.4.1.12	Hyperlink Control.....	259
2.4.1.13	List Box Control .....	261
2.4.1.14	Option Button Control.....	266
2.4.1.15	Repeating Section Control .....	267
2.4.1.16	Repeating Table Control.....	270
2.4.1.17	Rich Text Box Control.....	272
2.4.1.18	Section Control and Optional Section Control.....	275
2.4.1.19	Table Control.....	283
2.4.1.20	Text Box Control .....	284
2.4.1.21	Ignored Controls .....	295
2.4.1.22	Invalid Controls .....	295
2.4.1.23	Invalid Constructs .....	296
2.4.2	Control-Specific Attributes .....	296
2.4.2.1	action.....	298
2.4.2.2	allownonmatching .....	299
2.4.2.3	autoAdvance .....	299
2.4.2.4	auxDom .....	299
2.4.2.5	backgroundPicture .....	300
2.4.2.6	binding.....	300
2.4.2.7	bindingProperty .....	301
2.4.2.8	bindingType .....	301
2.4.2.9	boundProp .....	302
2.4.2.10	CtrlId .....	303
2.4.2.11	datafmt .....	304
2.4.2.12	disableEditing .....	309
2.4.2.13	enabledProperty .....	310
2.4.2.14	enabledValue .....	310
2.4.2.15	ghosted .....	311
2.4.2.16	ictID.....	311
2.4.2.17	ictVersion .....	311
2.4.2.18	inline.....	311
2.4.2.19	innerCtrl .....	311
2.4.2.20	inputscope .....	312
2.4.2.21	inputScopeId.....	312
2.4.2.22	layoutText .....	312
2.4.2.23	linkedToMaster .....	312
2.4.2.24	masterID .....	312
2.4.2.25	masterName .....	313
2.4.2.26	num .....	313
2.4.2.27	offValue.....	313
2.4.2.28	onValue .....	314

2.4.2.29	postbackModel.....	314
2.4.2.30	ref .....	315
2.4.2.31	SignatureBlock .....	315
2.4.2.32	SignedSectionDisplaySignatures .....	316
2.4.2.33	SignedSectionName .....	316
2.4.2.34	value.....	317
2.4.2.35	xctname .....	317
2.4.2.36	xmlToEdit .....	318
2.4.3	XSL Function Extensions .....	319
2.4.3.1	msxsl.....	319
2.4.3.1.1	string-compare .....	319
2.4.3.2	xdDate .....	319
2.4.3.2.1	AddDays .....	319
2.4.3.2.2	AddSeconds.....	320
2.4.3.2.3	Now .....	320
2.4.3.2.4	Today .....	320
2.4.3.3	xdEnvironment.....	320
2.4.3.3.1	IsBrowser.....	320
2.4.3.3.2	IsMobile.....	321
2.4.3.4	xdFormatting .....	321
2.4.3.5	xdImage.....	321
2.4.3.6	xdMath.....	321
2.4.3.6.1	Avg .....	321
2.4.3.6.2	Eval.....	321
2.4.3.6.3	Max.....	322
2.4.3.6.4	Min.....	322
2.4.3.6.5	Nz .....	322
2.4.3.7	xdUser .....	322
2.4.3.7.1	get-UserName .....	322
2.4.3.8	xdUtil.....	323
2.4.3.8.1	Match .....	323
2.4.3.9	xdXDocument .....	323
2.4.3.9.1	get-dom.....	323
2.4.3.9.2	getDOM .....	323
2.4.3.9.3	getnamednodeproperty .....	323
2.5	Print View Files (XSLT) Specification.....	324
2.6	Submit Files (XML) Specification .....	324
2.6.1	myFields .....	324
2.6.2	dataFields .....	325
2.7	Template.XML Specification.....	325
2.8	Upgrade.XSL Specification .....	325
2.8.1	MSXSL:Node-Set() .....	325
<b>3</b>	<b>Structure Examples .....</b>	<b>326</b>
3.1	The InfoPath Form Template Format .....	326
3.1.1	Simple Form Template .....	326
3.1.2	Complex Form Template .....	326
3.2	Form Definition File (XSF) Examples.....	327
3.2.1	XSF Extension Examples .....	329
3.3	XML Schema Files (XSD) Examples .....	330
3.4	Form View Files (XSL) Examples.....	331
3.4.1	Control representation .....	331
3.4.1.1	Button Control.....	331

3.4.1.2	Check Box Control .....	332
3.4.1.3	Contact Selector Control.....	333
3.4.1.4	Date Picker Control .....	334
3.4.1.5	Drop-Down List Control .....	335
3.4.1.6	Expression Box Control.....	338
3.4.1.7	File Attachment Control .....	339
3.4.1.8	Hyperlink Control.....	339
3.4.1.9	List Box Control.....	340
3.4.1.10	Option Button Control.....	342
3.4.1.11	Repeating Section Control .....	343
3.4.1.12	Repeating Table Control.....	344
3.4.1.13	Rich Text Box Control.....	347
3.4.1.14	Section Control and Optional Section Control .....	347
3.4.1.15	Table Control.....	350
3.4.1.16	Text Box Control.....	351
3.4.2	Control-Specific Attributes .....	353
3.4.3	XSL Function Extensions .....	360
3.5	Print View Files (XSLT) Examples .....	364
3.6	Submit Files (XML) Examples .....	365
3.7	Template.XML Examples .....	366
3.8	Upgrade.XSL Examples.....	366
3.8.1	Upgrade.XSL Example.....	367
3.8.2	MSXSL:Node-Set() Example .....	369
<b>4</b>	<b>Security Considerations.....</b>	<b>370</b>
4.1	The InfoPath Form Template Format .....	370
4.2	Template.XML Specification.....	370
<b>5</b>	<b>Appendix A: Full XML Schemas .....</b>	<b>371</b>
5.1	The InfoPath XSF XSD .....	371
5.2	The InfoPath XSF2 XSD .....	399
<b>6</b>	<b>Appendix B: Product Behavior .....</b>	<b>409</b>
<b>7</b>	<b>Change Tracking.....</b>	<b>410</b>
<b>8</b>	<b>Index .....</b>	<b>411</b>

# 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 (total amount, date filed, and so on.), 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, and the form file used to store the data. Details on how form files are associated with a form template are specified in [\[MS-IPFFX\]](#).

## 1.1 Glossary

The following terms are defined in [\[MS-GLOS\]](#):

### **GUID**

The following terms are defined in [\[MS-OFCGLOS\]](#):

**ActiveX Data Objects (ADO)**  
**ASP.NET control**  
**assembly**  
**attachment**  
**browser-compatible form template**  
**browser-enabled form template**  
**built-in control**  
**business object**  
**cabinet (.cab) file**  
**column**  
**connection string**  
**control**  
**data adapter**  
**data connection**  
**data connection library**  
**data source**  
**digital signature**  
**empty string**  
**Extended Backus-Naur Form (EBNF)**  
**file**  
**form**  
**form definition (.xsf) file**  
**form file**  
**form library**  
**form security level**  
**form server**  
**form template**  
**form template (.xsn) file**  
**form view**  
**friendly name**  
**group**

HTTP method  
left-to-right  
list  
list view  
lookup field  
main data connection  
main data source  
mobile device  
offline  
postback  
print view  
query  
repeating group  
right-to-left  
rule  
secondary data connection  
secondary data source  
site  
site collection  
SOAP action  
SQL statement  
Structured Query Language (SQL)  
submit  
symbol file  
Uniform Resource Locator (URL)  
Uniform Resource Name (URN)  
Universal Data Connection (.udc, .udcx) file  
user name  
Web Distributed Authoring and Versioning Protocol (WebDAV)  
Web service  
XML element  
XML fragment  
XML node  
XML schema  
XML schema document  
XPath expression  
XSL Transformation (XSLT)

The following terms are specific to this document:

**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.

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

## 1.2 References

### 1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact [dochelp@microsoft.com](mailto:dochelp@microsoft.com). We will assist you in finding the relevant information. Please check the archive site,

<http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624>, as an additional source.

[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-10646] International Organization for Standardization, "Information Technology - Universal Multiple-Octet Coded Character Set (UCS)", ISO/IEC 10646:2003, December 2003, <http://www.iso.ch/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=39921&ICS1>

[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/global/bb896001.aspx>

[MS-IPFFX] Microsoft Corporation, "[InfoPath Form File Format Specification](#)"

[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 Specification](#)".

[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.ietf.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-xmlsig-core-20020212/>

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

[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>

[MS-GLOS] Microsoft Corporation, "[Windows Protocols Master Glossary](#)".

[MS-OFCGLOS] Microsoft Corporation, "[Microsoft Office Master Glossary](#)".

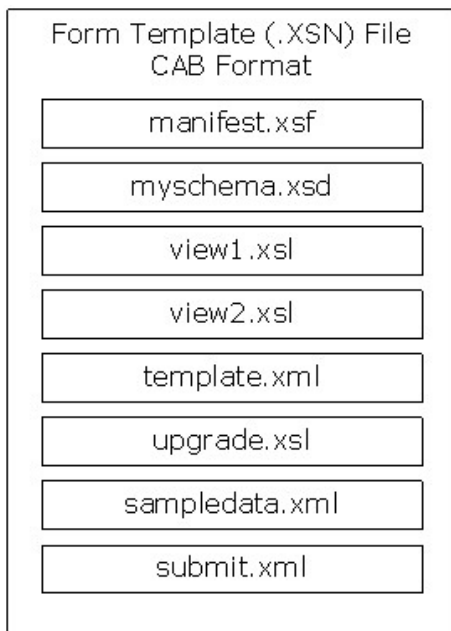
## 1.3 Structure Overview (Synopsis)

### 1.3.1 The InfoPath Form Template Format

A form template (.xsn) file is a CAB file, as specified in [\[MC-MCF\]](#), which 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 which 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](#).

### 1.3.2 Form Definition File (XSF)

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

1. The **list (1)** of files that comprise the form template (.xsn) file and the relationships between them.
2. Properties of the form template such as deployment information and user interface customizations.
3. Properties of certain **controls** within the form template. For example, a control can have associated rules (1) which 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](#).

### 1.3.3 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 (1)** 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 (1).

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](#).

### 1.3.4 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, what actions to take when a particular field's value changes.

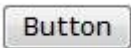

A form view is represented by an **XSL Transformation (XSLT)** file, which uses the .XSL file extension. There are one or more XSLT files in the form template:



- 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.

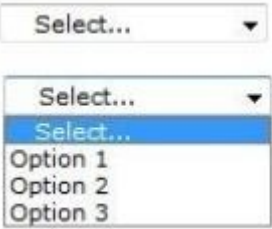

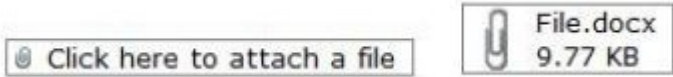

The default XSLT file is specified in the form definition (.xsf) file.

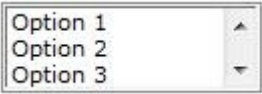

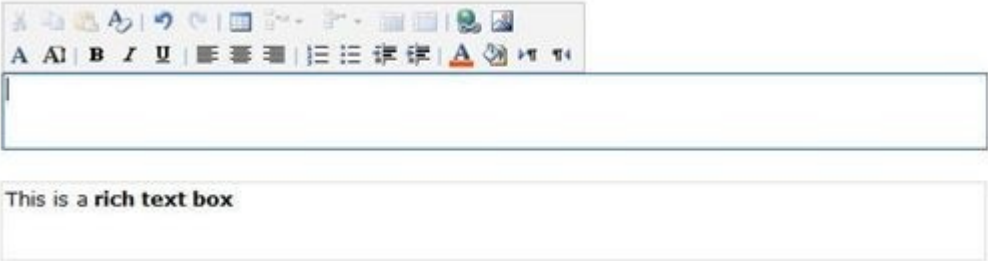
The contents of a XSLT file are used to transform the data in a form file into HTML, so the form can be rendered and edited in a Web browser. The resulting HTML visualization consists of HTML used to lay out and represent the controls and informative text. The HTML for a control will also contain properties 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	<p>A control used to execute an action. It is not tied to a field.</p> <p>The following figure illustrates a typical representation of a button control:</p>  <p><b>Figure 2: A button control</b></p> <p>The visualization and properties of the control are specified in section <a href="#">2.4.1.5</a>. The relationship to the XML schema of the form template is specified in section <a href="#">2.3.1.1</a>. Behaviors affecting the control are specified in section <a href="#">2.2.144</a> and <a href="#">2.2.62</a>.</p>
Check Box	<p>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.</p> <p>The following figure illustrates a typical representation of a check box control:</p> 

Control	Description and sample visual representation
	<p><b>Figure 3: A set of check box controls</b></p> <p>The visualization and properties of the control are specified in section <a href="#">2.4.1.6</a>. The relationship to the XML schema of the form template is specified in section <a href="#">2.3.1.2</a>. Behaviors affecting the control are specified in section <a href="#">2.2.144</a> and <a href="#">2.2.62</a>.</p>
Contact Selector	<p>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:</p>  <p><b>Figure 4: A contact selector control</b></p> <p>The visualization and properties of the control are specified in section <a href="#">2.4.1.7</a>. The relationship to the XML schema of the form template is specified in section <a href="#">2.3.1.3</a>. Behaviors affecting the control are specified in section <a href="#">2.2.144</a> and <a href="#">2.2.62</a>.</p>
Date Picker	<p>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:</p>  <p><b>Figure 5: A date picker control</b></p> <p>The visualization and properties of the control are specified in section <a href="#">2.4.1.8</a>. The relationship to the XML schema of the form template is specified in section <a href="#">2.3.1.4</a>. Behaviors affecting the control are specified in section <a href="#">2.2.144</a> and <a href="#">2.2.62</a>.</p>
Drop-Down List Box	<p>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:</p>

Control	Description and sample visual representation
	 <p><b>Figure 6: A drop-down list box control</b></p> <p>The visualization and properties of the control are specified in section <a href="#">2.4.1.9</a>. The relationship to the XML schema of the form template is specified in section <a href="#">2.3.1.5</a>. Behaviors affecting the control are specified in section <a href="#">2.2.144</a> and <a href="#">2.2.62</a>.</p>
Expression Box	<p>A read-only text control used to display information.</p> <p>The following figure illustrates a typical representation of an expression box control:</p>  <p><b>Figure 7: An expression box control</b></p> <p>The visualization and properties of the control are specified in section <a href="#">2.4.1.10</a>. The relationship to the XML schema of the form template is specified in section <a href="#">2.3.1.6</a>. Behaviors affecting the control are specified in section <a href="#">2.2.144</a> and <a href="#">2.2.62</a>.</p>
File Attachment	<p>A control that allows users to attach files to their form files. Each file attachment control permits one file to be attached.</p> <p>The following figure illustrates a typical representation of a file attachment control:</p>  <p><b>Figure 8: A file attachment control</b></p> <p>The visualization and properties of the control are specified in section <a href="#">2.4.1.11</a>. The relationship to the XML schema of the form template is specified in section <a href="#">2.3.1.7</a>. Behaviors affecting the control are specified in section <a href="#">2.2.144</a> and <a href="#">2.2.62</a>.</p>
Hyperlink	<p>A control that can be used to link to a URL (Uniform Resource Locator).</p> <p>The following figure illustrates a typical representation of a hyperlink:</p>  <p><b>Figure 9: A hyperlink control</b></p> <p>The visualization and properties of the control are specified in section <a href="#">2.4.1.12</a>. The relationship to the XML schema of the form template is specified in section <a href="#">2.3.1.8</a>. Behaviors affecting the control are specified in section <a href="#">2.2.144</a> and <a href="#">2.2.62</a>.</p>

Control	Description and sample visual representation
List Box	<p>A control that presents users with a list (1) of choices in a box from which users select the appropriate item.</p> <p>The following figure illustrates a typical representation of a list box control:</p>  <p><b>Figure 10: A list box control</b></p> <p>The visualization and properties of the control are specified in section <a href="#">2.4.1.13</a>. The relationship to the XML schema of the form template is specified in section <a href="#">2.3.1.9</a>. Behaviors affecting the control are specified in section <a href="#">2.2.144</a> and <a href="#">2.2.62</a>.</p>
Option Button	<p>A control that lets users select from a set of mutually exclusive choices. When one option button in a <b>group (1)</b> is selected, the other option buttons are cleared.</p> <p>The following figure illustrates a typical representation of an option button control:</p>  <p><b>Figure 11: A set of option button controls</b></p> <p>The visualization and properties of the control are specified in section <a href="#">2.4.1.14</a>. The relationship to the XML schema of the form template is specified in section <a href="#">2.3.1.10</a>. Behaviors affecting the control are specified in section <a href="#">2.2.144</a> and <a href="#">2.2.62</a>.</p>
Rich Text Box	<p>A text input control that can contain formatted text, including bold and italic text, and a variety of fonts, font sizes, and font colors.</p> <p>The following figure illustrates a typical representation of a rich text box control:</p>  <p><b>Figure 12: A rich text box control, the top image shows it in the editing state</b></p> <p>The visualization and properties of the control are specified in section <a href="#">2.4.1.17</a>. The relationship to the XML schema of the form template is specified in section <a href="#">2.3.1.13</a>. Behaviors affecting the control are specified in section <a href="#">2.2.144</a> and <a href="#">2.2.62</a>.</p>
Table	<p>A table used to lay out the form. It is not tied to a field.</p>

Control	Description and sample visual representation
	<p>The following figure illustrates a typical representation of a table control:</p>  <p><b>Figure 13: A table</b></p> <p>The visualization and properties of the control are specified in section <a href="#">2.4.1.19</a>. The relationship to the XML schema of the form template is specified in section <a href="#">2.3.1.15</a>. Behaviors affecting the control are specified in section <a href="#">2.2.144</a> and <a href="#">2.2.62</a>.</p>
Text Box	<p>A text input control that can contain any unformatted text. Text box controls cannot contain formatted text.</p> <p>The following figure illustrates a typical representation of a text box control:</p>  <p><b>Figure 14: A text box control</b></p> <p>The visualization and properties of the control are specified in section <a href="#">2.4.1.20</a>. The relationship to the XML schema of the form template is specified in section <a href="#">2.3.1.16</a>. Behaviors affecting the control are specified in section <a href="#">2.2.144</a> and <a href="#">2.2.62</a>.</p>
Repeating Table	<p>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.</p> <p>The following figure illustrates a typical representation of a repeating table control:</p>  <p><b>Figure 15: A repeating table control with three text box controls per row</b></p> <p>The visualization and properties of the control are specified in section <a href="#">2.4.1.16</a>. The relationship to the XML schema of the form template is specified in section <a href="#">2.3.1.12</a>. Behaviors affecting the control are specified in section <a href="#">2.2.144</a> and <a href="#">2.2.62</a>.</p>
Section, Optional Section	<p>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.</p> <p>The following figure illustrates a typical representation of a section control:</p>  <p><b>Figure 16: A section control containing two text box controls</b></p>

Control	Description and sample visual representation
	The visualization and properties of the control are specified in section <a href="#">2.4.1.18</a> . The relationship to the XML schema of the form template is specified in section <a href="#">2.3.1.14</a> . Behaviors affecting the control are specified in section <a href="#">2.2.144</a> and <a href="#">2.2.62</a> .
Repeating Section	<p>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.</p> <p>The following figure illustrates a typical representation of a repeating section control:</p>  <p><b>Figure 17: A repeating section control</b></p> <p>The visualization and properties of the control are specified in section <a href="#">2.4.1.15</a>. The relationship to the XML schema of the form template is specified in section <a href="#">2.3.1.11</a>. Behaviors affecting the control are specified in section <a href="#">2.2.144</a> and <a href="#">2.2.62</a>.</p>

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](#).

### 1.3.5 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, using dark text on a white background, or suppressing 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 will be 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](#).

### 1.3.6 Submit Files (XML)

**Submit** files are XML files, as specified 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](#).

### 1.3.7 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, 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](#).

### 1.3.8 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](#).

### 1.3.9 Resource Files

The following files are present for the form templates.

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

The list (1) of files in the form template is specified in sections [2.1](#) and [2.2.97](#). Examples are provided in section [3.1](#).

### 1.3.10 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 (1) so that they can be accessed **offline**.
- The merge.xsl file contains an XSL Transformation (XSLT), as specified 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 specified 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.

- The importerrors.xml file may contain errors resulting from importing files.

The list (1) 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, specified in [\[MC-MCF\]](#).

All XSL Transformation (XSLT) files contained in a form template (.xsn) file are XSLTs files, as specified in [\[W3C-XSLT\]](#).

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

Template.xml is a form file, as specified 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, 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 lists 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 which have different hexadecimal values than the following: 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 (1) 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 MAY contain one or more *submitdata.xml* files if there are data connections (1) 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.xml

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

### 2.1.9 merge.xml

A form template (.xsn) file MAY contain merge.xml. merge.xml 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 which is 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 as 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 which 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 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 Form Definition File (XSF) Extension Specification, section [2.2.147](#).

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

Element	Specified in Section
<a href="#">action</a>	<a href="#">2.2.90</a>
<a href="#">adoAdapter</a>	<a href="#">2.2.38</a>
<a href="#">allowedActions</a>	<a href="#">2.2.89</a>
<a href="#">allowedControl</a>	<a href="#">2.2.102</a>
<a href="#">allowedTasks</a>	<a href="#">2.2.91</a>

Element	Specified in Section
<a href="#">applicationParameters</a>	<a href="#">2.2.23</a>
<a href="#">assignmentAction</a>	<a href="#">2.2.139</a>
<a href="#">attachmentFileName</a>	<a href="#">2.2.57</a>
<a href="#">attributeData</a>	<a href="#">2.2.105</a>
<a href="#">autoRecovery</a>	<a href="#">2.2.31</a>
<a href="#">bcc</a>	<a href="#">2.2.54</a>
<a href="#">button</a>	<a href="#">2.2.106</a>
<a href="#">button</a>	<a href="#">2.2.110</a>
<a href="#">calculatedField</a>	<a href="#">2.2.146</a>
<a href="#">calculations</a>	<a href="#">2.2.145</a>
<a href="#">cc</a>	<a href="#">2.2.53</a>
<a href="#">chooseFragment</a>	<a href="#">2.2.107</a>
<a href="#">closeDocumentAction</a>	<a href="#">2.2.142</a>
<a href="#">customCategory</a>	<a href="#">2.2.95</a>
<a href="#">customValidation</a>	<a href="#">2.2.62</a>
<a href="#">dataAdapters</a>	<a href="#">2.2.59</a>
<a href="#">dataObject</a>	<a href="#">2.2.36</a>
<a href="#">dataObjects</a>	<a href="#">2.2.35</a>
<a href="#">davAdapter</a>	<a href="#">2.2.48</a>
<a href="#">dialogBoxExpressionAction</a>	<a href="#">2.2.137</a>
<a href="#">dialogBoxMessageAction</a>	<a href="#">2.2.136</a>
<a href="#">documentSchema</a>	<a href="#">2.2.61</a>
<a href="#">documentSchemas</a>	<a href="#">2.2.60</a>
<a href="#">documentSignatures</a>	<a href="#">2.2.125</a>
<a href="#">documentVersionUpgrade</a>	<a href="#">2.2.128</a>
<a href="#">domEventHandler</a>	<a href="#">2.2.66</a>
<a href="#">domEventHandlers</a>	<a href="#">2.2.65</a>
<a href="#">editing</a>	<a href="#">2.2.111</a>
<a href="#">editWith</a>	<a href="#">2.2.108</a>
<a href="#">emailAdapter</a>	<a href="#">2.2.51</a>

Element	Specified in Section
<a href="#">errorCondition</a>	<a href="#">2.2.63</a>
<a href="#">errorMessage</a>	<a href="#">2.2.64</a>
<a href="#">errorMessage</a>	<a href="#">2.2.75</a>
<a href="#">exitRuleSet</a>	<a href="#">2.2.135</a>
<a href="#">exportToExcel</a>	<a href="#">2.2.28</a>
<a href="#">exportToWeb</a>	<a href="#">2.2.27</a>
<a href="#">extension</a>	<a href="#">2.2.131</a>
<a href="#">extensions</a>	<a href="#">2.2.130</a>
<a href="#">externalView</a>	<a href="#">2.2.104</a>
<a href="#">externalViews</a>	<a href="#">2.2.103</a>
<a href="#">featureRestrictions</a>	<a href="#">2.2.25</a>
<a href="#">field</a>	<a href="#">2.2.47</a>
<a href="#">field</a>	<a href="#">2.2.71</a>
<a href="#">fields</a>	<a href="#">2.2.70</a>
<a href="#">file</a>	<a href="#">2.2.98</a>
<a href="#">fileName</a>	<a href="#">2.2.50</a>
<a href="#">fileNew</a>	<a href="#">2.2.93</a>
<a href="#">fileProperties</a>	<a href="#">2.2.99</a>
<a href="#">files</a>	<a href="#">2.2.97</a>
<a href="#">folderURL</a>	<a href="#">2.2.49</a>
<a href="#">footer</a>	<a href="#">2.2.117</a>
<a href="#">fragmentToInsert</a>	<a href="#">2.2.113</a>
<a href="#">getUserNameFromData</a>	<a href="#">2.2.84</a>
<a href="#">group</a>	<a href="#">2.2.86</a>
<a href="#">header</a>	<a href="#">2.2.116</a>
<a href="#">hwsAdapter</a>	<a href="#">2.2.40</a>
<a href="#">hwsOperation</a>	<a href="#">2.2.42</a>
<a href="#">hwsWorkflow</a>	<a href="#">2.2.87</a>
<a href="#">importParameters</a>	<a href="#">2.2.67</a>
<a href="#">importSource</a>	<a href="#">2.2.68</a>

Element	Specified in Section
<a href="#">initialXmlDocument</a>	<a href="#">2.2.94</a>
<a href="#">input</a>	<a href="#">2.2.43</a>
<a href="#">intro</a>	<a href="#">2.2.56</a>
<a href="#">listProperties</a>	<a href="#">2.2.69</a>
<a href="#">location</a>	<a href="#">2.2.88</a>
<a href="#">mainpane</a>	<a href="#">2.2.114</a>
<a href="#">masterDetail</a>	<a href="#">2.2.112</a>
<a href="#">membership</a>	<a href="#">2.2.83</a>
<a href="#">menu</a>	<a href="#">2.2.119</a>
<a href="#">menuArea</a>	<a href="#">2.2.120</a>
<a href="#">message</a>	<a href="#">2.2.127</a>
<a href="#">onLoad</a>	<a href="#">2.2.79</a>
<a href="#">openNewDocumentAction</a>	<a href="#">2.2.141</a>
<a href="#">operation</a>	<a href="#">2.2.41</a>
<a href="#">override</a>	<a href="#">2.2.22</a>
<a href="#">package</a>	<a href="#">2.2.96</a>
<a href="#">partFragment</a>	<a href="#">2.2.44</a>
<a href="#">permissions</a>	<a href="#">2.2.101</a>
<a href="#">print</a>	<a href="#">2.2.29</a>
<a href="#">printSettings</a>	<a href="#">2.2.115</a>
<a href="#">property</a>	<a href="#">2.2.100</a>
<a href="#">query</a>	<a href="#">2.2.32</a>
<a href="#">query</a>	<a href="#">2.2.37</a>
<a href="#">queryAction</a>	<a href="#">2.2.140</a>
<a href="#">role</a>	<a href="#">2.2.82</a>
<a href="#">roles</a>	<a href="#">2.2.81</a>
<a href="#">rule</a>	<a href="#">2.2.133</a>
<a href="#">ruleSet</a>	<a href="#">2.2.143</a>
<a href="#">ruleSetAction</a>	<a href="#">2.2.132</a>
<a href="#">ruleSets</a>	<a href="#">2.2.144</a>

Element	Specified in Section
<a href="#">save</a>	<a href="#">2.2.26</a>
<a href="#">save</a>	<a href="#">2.2.80</a>
<a href="#">schemaErrorMessages</a>	<a href="#">2.2.21</a>
<a href="#">script</a>	<a href="#">2.2.34</a>
<a href="#">scripts</a>	<a href="#">2.2.33</a>
<a href="#">sendMail</a>	<a href="#">2.2.30</a>
<a href="#">sharepointListAdapter</a>	<a href="#">2.2.46</a>
<a href="#">signedDataBlock</a>	<a href="#">2.2.126</a>
<a href="#">solutionProperties</a>	<a href="#">2.2.24</a>
<a href="#">subject</a>	<a href="#">2.2.55</a>
<a href="#">submit</a>	<a href="#">2.2.72</a>
<a href="#">submitAction</a>	<a href="#">2.2.73</a>
<a href="#">submitAction</a>	<a href="#">2.2.134</a>
<a href="#">submitToHostAdapter</a>	<a href="#">2.2.58</a>
<a href="#">successMessage</a>	<a href="#">2.2.74</a>
<a href="#">switchViewAction</a>	<a href="#">2.2.138</a>
<a href="#">task</a>	<a href="#">2.2.92</a>
<a href="#">taskpane</a>	<a href="#">2.2.121</a>
<a href="#">to</a>	<a href="#">2.2.52</a>
<a href="#">toolbar</a>	<a href="#">2.2.118</a>
<a href="#">unboundControls</a>	<a href="#">2.2.109</a>
<a href="#">useHttpHandler</a>	<a href="#">2.2.76</a>
<a href="#">useQueryAdapter</a>	<a href="#">2.2.78</a>
<a href="#">userName</a>	<a href="#">2.2.85</a>
<a href="#">useScriptHandler</a>	<a href="#">2.2.77</a>
<a href="#">useTransform</a>	<a href="#">2.2.129</a>
<a href="#">view</a>	<a href="#">2.2.123</a>
<a href="#">views</a>	<a href="#">2.2.122</a>
<a href="#">webServiceAdapter</a>	<a href="#">2.2.39</a>
<a href="#">xDocumentClass</a>	<a href="#">2.2.20</a>

Element	Specified in Section
<a href="#">xmlFileAdapter</a>	<a href="#">2.2.45</a>
<a href="#">xmlToEdit</a>	<a href="#">2.2.124</a>

### 2.2.1 xdTitle

This simple type specifies restrictions for a title string.

Referenced By
<a href="#">adoAdapter.xsfschema@name</a>
<a href="#">adoAdapterExtension.xsf2@ref</a>
<a href="#">adoAdapterExtension.xsf2@submitAdapterName</a>
<a href="#">button.xsfschema@caption</a>
<a href="#">button.xsfschema@tooltip</a>
<a href="#">command.xsf2@caption</a>
<a href="#">customCategory.xsfschema@name</a>
<a href="#">dataObject.xsfschema@name</a>
<a href="#">davAdapter.xsfschema@name</a>
<a href="#">davAdapterExtension.xsf2@ref</a>
<a href="#">editWith.xsfschema@caption</a>
<a href="#">emailAdapter.xsfschema@name</a>
<a href="#">emailAdapterExtension.xsf2@ref</a>
<a href="#">field.xsfschema@columnName</a>
<a href="#">field.xsfschema@name</a>
<a href="#">hwsAdapter.xsfschema@name</a>
<a href="#">initialXmlDocument.xsfschema@caption</a>
<a href="#">inputScope.xsf2@caption</a>
<a href="#">menu.xsfschema@caption</a>
<a href="#">sharepointListAdapter.xsfschema@name</a>
<a href="#">sharepointListAdapterExtension.xsf2@ref</a>
<a href="#">submitAction.rule.xsfschema@adapter</a>
<a href="#">submitAction.submit.xsf2@adapter</a>
<a href="#">submitAction.submit.xsfschema@adapter</a>

Referenced By
<a href="#">submitToHostAdapter.xsfschema@name</a>
<a href="#">toolbar.xsfschema@caption</a>
<a href="#">toolbar.xsfschema@name</a>
<a href="#">viewExtension.xsf2@ref</a>
<a href="#">webServiceAdapter.xsfschema@name</a>
<a href="#">webServiceAdapterExtension.xsf2@ref</a>
<a href="#">xmlFileAdapter.xsfschema@name</a>
<a href="#">xmlFileAdapterExtension.xsf2@ref</a>
<a href="#">xmlToEditExtension.xsf2@ref</a>

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

This simple type specifies restrictions for specifying the name of a form view.

Referenced By
<a href="#">externalView.xsfschema@name</a>
<a href="#">includedView.xsf2@name</a>
<a href="#">role.xsfschema@name</a>
<a href="#">switchViewAction.xsfschema@view</a>
<a href="#">view.xsfschema@caption</a>
<a href="#">view.xsfschema@name</a>

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}\/\\#&"><])(([\p{Zl}\p{Zp}\p{C}\/\\#&"><]) * ([^\p{Z}\p{C}\/\\#&"><]))?" />
    </xsd:restriction>
</xsd:simpleType>

```

### 2.2.3 xdRoleName

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

Referenced By
<a href="#">role.xsfschema@name</a>

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

This 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
<a href="#">action.xsfschema@canInitiateWorkflow</a>
<a href="#">adoAdapter.xsfschema@queryAllowed</a>
<a href="#">adoAdapter.xsfschema@submitAllowed</a>
<a href="#">autoUpdatePrompt.xsf2@showPrompt</a>
<a href="#">calculations.xsfschema@treatBlankValueAsZero</a>
<a href="#">closeDocumentAction.xsfschema@promptToSaveChanges</a>
<a href="#">contentTypeTemplate.xsf2@browserEnable</a>
<a href="#">dataObject.xsfschema@initOnLoad</a>
<a href="#">davAdapter.xsfschema@overwriteAllowed</a>
<a href="#">davAdapter.xsfschema@queryAllowed</a>
<a href="#">davAdapter.xsfschema@submitAllowed</a>
<a href="#">documentSchema.xsfschema@rootSchema</a>
<a href="#">editWith.xsfschema@autoComplete</a>

Referenced By
<a href="#">editWith.xsfschema@proofing</a>
<a href="#">emailAdapter.xsfschema@queryAllowed</a>
<a href="#">emailAdapter.xsfschema@submitAllowed</a>
<a href="#">field.sharepointListAdapter.xsfschema@isLookup</a>
<a href="#">field.xsfschema@required</a>
<a href="#">field.xsfschema@viewable</a>
<a href="#">fieldExtension.xsf2@readWrite</a>
<a href="#">hwsAdapter.xsfschema@queryAllowed</a>
<a href="#">hwsAdapter.xsfschema@submitAllowed</a>
<a href="#">hwsWorkflow.xsfschema@taskpaneVisible</a>
<a href="#">importParameters.xsfschema@enabled</a>
<a href="#">importParameters.xsfschema@useScriptHandler</a>
<a href="#">managedCode.xsf2@enabled</a>
<a href="#">mergedPrintView.xsf2@isCustomizable</a>
<a href="#">mergedPrintView.xsf2@isDefault</a>
<a href="#">offline.xsf2@cacheQueries</a>
<a href="#">offline.xsf2@openIfQueryFails</a>
<a href="#">partFragment.xsfschema@sendAsString</a>
<a href="#">printSettings.xsfschema@collate</a>
<a href="#">roles.xsfschema@hideStatusBarDisplay</a>
<a href="#">rule.xsfschema@isEnabled</a>
<a href="#">scripts.xsfschema@enforceScriptTimeout</a>
<a href="#">sendByMail.xsf2@disableEmailForms</a>
<a href="#">server.xsf2@isMobileEnabled</a>
<a href="#">server.xsf2@isPreSubmitPostBackEnabled</a>
<a href="#">sharepointListAdapter.xsfschema@queryAllowed</a>
<a href="#">sharepointListAdapter.xsfschema@submitAllowed</a>
<a href="#">sharepointListAdapterExtension.xsf2@queryThisFormOnly</a>
<a href="#">solutionDefinition.xsf2@allowClientOnlyCode</a>
<a href="#">solutionDefinition.xsf2@verifyOnServer</a>

Referenced By
<a href="#">solutionProperties.xsfschema@allowCustomization</a>
<a href="#">solutionProperties.xsfschema@automaticallyCreateNodes</a>
<a href="#">submit.xsf2@disableMenuItem</a>
<a href="#">submit.xsf2@showSignatureReminder</a>
<a href="#">submit.xsf2@showStatusDialog</a>
<a href="#">submit.xsfschema@disableMenuItem</a>
<a href="#">submit.xsfschema@showSignatureReminder</a>
<a href="#">submit.xsfschema@showStatusDialog</a>
<a href="#">submitToHostAdapter.xsfschema@queryAllowed</a>
<a href="#">submitToHostAdapter.xsfschema@submitAllowed</a>
<a href="#">toolbar.xsf2@enabledBottom</a>
<a href="#">toolbar.xsf2@enabledTop</a>
<a href="#">view.xsfschema@showMenuItem</a>
<a href="#">viewExtension.xsf2@clientOnly</a>
<a href="#">viewExtension.xsf2@readOnly</a>
<a href="#">warning.xsf2@hidden</a>
<a href="#">webServiceAdapter.xsfschema@queryAllowed</a>
<a href="#">webServiceAdapter.xsfschema@submitAllowed</a>
<a href="#">webServiceAdapter.xsfschema@useDataSet</a>
<a href="#">webServiceAdapterExtension.xsf2@trackDataSetChanges</a>
<a href="#">wss.xsf2@browserEnable</a>
<a href="#">xDocumentClass.xsfschema@dataFormSolution</a>
<a href="#">xDocumentClass.xsfschema@requireFullTrust</a>
<a href="#">xmlToEditExtension.xsf2@allowLinkedImages</a>
<a href="#">xmlToEditExtension.xsf2@excludeEmbeddedImages</a>

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>
```

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

## 2.2.5 xdEnabledDisabled

This 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
<a href="#">autoRecovery.xsfschema@feature</a>
<a href="#">exportToExcel.xsfschema@ui</a>
<a href="#">exportToPDForXPS.xsf2@ui</a>
<a href="#">exportToWeb.xsfschema@ui</a>
<a href="#">print.xsfschema@ui</a>
<a href="#">save.featureRestrictions.xsfschema@ui</a>
<a href="#">sendMail.xsfschema@ui</a>

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

This 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
<a href="#">importSource.xsfschema@authoringOfTransform</a>
<a href="#">xDocumentClass.xsfschema@trustSetting</a>

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

This 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
<a href="#">attachmentFileName.emailAdapter.xsfschem@valueType</a>
<a href="#">bcc.emailAdapter.xsfschema@valueType</a>
<a href="#">cc.emailAdapter.xsfschema@valueType</a>
<a href="#">fileName.davAdapter.xsfschema@valueType</a>
<a href="#">subject.emailAdapter.xsfschema@valueType</a>
<a href="#">to.emailAdapter.xsfschema@valueType</a>

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

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

Referenced By
<a href="#">file.xsfschema@name</a>
<a href="#">importSource.xsfschema@schema</a>
<a href="#">importSource.xsfschema@transform</a>
<a href="#">initialXmlDocument.xsfschema@href</a>
<a href="#">mainpane.xsfschema@transform</a>

Referenced By
<a href="#">script.xsfschema@src</a>
<a href="#">solutionProperties.xsfschema@lastOpenView</a>
<a href="#">useTransform.xsfschema@transform</a>

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

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

The **language** attribute of the [scripts](#) element MUST NOT be present. The **scriptLanguage** attribute of the [solutionProperties](#) element attribute MUST be ignored.

Referenced By
<a href="#">scripts.xsfschema@language</a>
<a href="#">solutionProperties.xsfschema@scriptLanguage</a>

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

This simple type specifies restrictions for specifying the version of the form template.

Referenced By
<a href="#">solutionProperties.xsfschema@lastVersionNeedingTransform</a>
<a href="#">useTransform.xsfschema@maxVersionToUpgrade</a>
<a href="#">useTransform.xsfschema@minVersionToUpgrade</a>

Referenced By
<a href="#">xDocumentClass.xsfschema@solutionFormatVersion</a>
<a href="#">xDocumentClass.xsfschema@solutionVersion</a>

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

This simple type specifies restrictions for specifying an **empty string (1)**.

Referenced By
<a href="#">useTransform.xsfschema@transform</a>

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

This simple type specifies restrictions for specifying an error message.

Referenced By
<a href="#">errorMessage.xsfschema</a>

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

This 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
<a href="#">externalView.xsfschema@designMode</a>
<a href="#">view.xsfschema@designMode</a>

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

This 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
<a href="#">xDocumentClass.xsfschema@trustLevel</a>

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

This simple type specifies restrictions for specifying the name of a signed data block.

Referenced By
<a href="#">signedDataBlock.xsfschema@name</a>

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

This simple type specifies restrictions for specifying the confirmation message that is displayed when a **digital signature (1)** is applied to the form or section of the form.

Referenced By
<a href="#">message.signedDataBlock.xsfschema</a>

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

This 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
<a href="#">signedDataBlock.xsfschema@mode</a>

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

This simple type specifies restrictions for an attribute that MUST NOT be present.

Referenced By
<a href="#">action.xsfschema@name</a>
<a href="#">task.xsfschema@name</a>

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="[^-^\.^\\^\\[^\]]^\\|^\\+^?^\\*^@^\\{^\\}\\^\\ (^\\) ^>^<^=^;^,]*"/>
  </xsd:restriction>
</xsd:simpleType>
```

### 2.2.19 xdHWSCaption

This simple type specifies restrictions for an attribute that MUST NOT be present.

Referenced By
<a href="#">action.xsfschema@caption</a>
<a href="#">task.xsfschema@caption</a>

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

This 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 (for example, loading, upgrading, and so on.) of all forms generated from the form template. This element MUST be the root element of the form definition (.xsf) file.

Child Elements
<a href="#">applicationParameters</a>
<a href="#">calculations</a>
<a href="#">customValidation</a>

Child Elements
<a href="#">dataAdapters</a>
<a href="#">dataObjects</a>
<a href="#">documentSchemas</a>
<a href="#">documentSignatures</a>
<a href="#">documentVersionUpgrade</a>
<a href="#">domEventHandlers</a>
<a href="#">extensions</a>
<a href="#">externalViews</a>
<a href="#">featureRestrictions</a>
<a href="#">fileNew</a>
<a href="#">hwsWorkflow</a>
<a href="#">importParameters</a>
<a href="#">listProperties</a>
<a href="#">onLoad</a>
<a href="#">package</a>
<a href="#">permissions</a>
<a href="#">query</a>
<a href="#">roles</a>
<a href="#">ruleSets</a>
<a href="#">save</a>
<a href="#">schemaErrorMessages</a>
<a href="#">scripts</a>
<a href="#">submit</a>
<a href="#">taskpane</a>
<a href="#">views</a>

#### Attributes:

**author** : This attribute specifies the name of the author of the form template. If this attribute is not present, its value MUST be interpreted as an empty string (1).

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

**name** : This attribute specifies an identifier for the form template in the form of a **URN**. This attribute MUST be specified.

**productVersion** : This attribute specifies 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** : This attribute 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 (1) MUST be loaded with an elevated form security level. An elevated form security level allows cross-domain data connections (1) and full access to business objects. If this attribute is not present, its value MUST be interpreted as "no".

**solutionFormatVersion** : This attribute specifies 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** : This attribute specifies 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** : This attribute specifies 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:all>
  </xsd:complexType>
</xsd:element>
```

```

    <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

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

Parent Elements
<a href="#">xDocumentClass</a>

Child Elements
<a href="#">override</a>

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

### 2.2.22 override

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

Parent Elements
<a href="#">schemaErrorMessages</a>

Child Elements
<a href="#">errorMessage</a>

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

This element MUST be ignored.

Parent Elements
<a href="#">xDocumentClass</a>

Child Elements
<a href="#">solutionProperties</a>

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

This element MUST be ignored.

Parent Elements
<a href="#">applicationParameters</a>

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

This element specifies one or more of the following features that are restricted when editing the form (1):

- 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
<a href="#">xDocumentClass</a>

Child Elements
<a href="#">autoRecovery</a>
<a href="#">exportToExcel</a>
<a href="#">exportToWeb</a>

Child Elements
<a href="#">print</a>
<a href="#">save</a>
<a href="#">sendMail</a>

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

This element specifies whether the UI elements of the form (1) 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
<a href="#">featureRestrictions</a>

Attributes:

**ui** : This attribute 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

This element MUST be ignored.

Parent Elements
<a href="#">featureRestrictions</a>

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

This element MUST be ignored.

Parent Elements
<a href="#">featureRestrictions</a>

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

This element specifies whether the UI elements of the form (1) and keyboard shortcuts for printing the form (1) are disabled. Restricting printing through this element MUST NOT disable printing the form (1) through use of form code.

Parent Elements
<a href="#">featureRestrictions</a>

Attributes:

**ui** : This attribute 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="xsd:boolean" use="required"/>
  </xsd:complexType>
</xsd:element>
```

**2.2.30 sendMail**

This element MUST be ignored.

Parent Elements
<a href="#">featureRestrictions</a>

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="xsd:boolean" use="required"/>
  </xsd:complexType>
</xsd:element>
```

**2.2.31 autoRecovery**

This element MUST be ignored.

Parent Elements
<a href="#">featureRestrictions</a>

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="xsd:boolean" use="required"/>
  </xsd:complexType>
</xsd:element>
```

### 2.2.32 query

This element specifies the main data connection **data adapter** that queries a **data source (2)** for data to populate the **main data source**. The main data connection MUST specify a data connection (1) 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, 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
<a href="#">xDocumentClass</a>

Child Elements
<a href="#">adoAdapter</a>
<a href="#">queryAction</a>
<a href="#">sharepointListAdapter</a>
<a href="#">webServiceAdapter</a>
<a href="#">xmlFileAdapter</a>

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

This element MUST NOT be present.

Parent Elements
<a href="#">xDocumentClass</a>

Child Elements
<a href="#">script</a>

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

This element MUST NOT be present.

Parent Elements
<a href="#">scripts</a>

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

This 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
<a href="#">xDocumentClass</a>

Child Elements
<a href="#">dataObject</a>

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

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

Parent Elements
<a href="#">dataObjects</a>

Child Elements
<a href="#">query</a>

Attributes:

**initOnLoad** : This attribute specifies whether the secondary data source **MUST** be queried when the form (1) is loaded. If this attribute is not present, its value **MUST** be interpreted as "no".

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

**schema** : This attribute specifies 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

This element specifies a data adapter that queries a secondary data source.

Parent Elements
<a href="#">dataObject</a>

Child Elements
<a href="#">adoAdapter</a>
<a href="#">sharepointListAdapter</a>
<a href="#">webServiceAdapter</a>
<a href="#">xmlFileAdapter</a>

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

This 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 (2).

Parent Elements
<a href="#">dataAdapters</a>
<a href="#">query</a>

Attributes:

**commandText** : This attribute specifies the **SQL statement** that is used for querying or submitting data to a database.

**connectionString** : This attribute specifies the ActiveX Data Objects (ADO) **connection string** that is used to connect to a database. The specified value **MUST NOT** specify a data connection (1) to a data source (2) file with the extensions ".mdb", ".mde", or ".accdb".

**name** : Specifies 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 by as an empty string (1).

**queryAllowed** : This attribute 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="xsd:string" 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="xsd:boolean" use="optional"/>
    <xsd:attribute name="submitAllowed" type="xsd:boolean" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

## 2.2.39 webServiceAdapter

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

When a form (1) 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 (Extensible Markup Language) template file, as specified by section [2.6](#), which is populated by data extracted from the form file.

Parent Elements
-----------------

Parent Elements
<a href="#">dataAdapters</a>
<a href="#">query</a>
<a href="#">submit</a>

Child Elements
<a href="#">operation</a>

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. If this attribute is not present, its value MUST be interpreted as an empty string (1).

**queryAllowed** : This attribute 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** : This attribute 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** : This attribute 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** : This attribute specifies the URL (Uniform Resource Locator) of the Web service. If this attribute is not present, its value MUST be interpreted as an empty string (1).

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

This element MUST NOT be present.

Parent Elements
-----------------

Parent Elements
<a href="#">dataAdapters</a>

Child Elements
<a href="#">hwsOperation</a>

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

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

Parent Elements
<a href="#">webServiceAdapter</a>

Child Elements
<a href="#">input</a>

Attributes:

**name** : This attribute specifies the name of the Web service method.

**serviceUrl** : This attribute specifies the URL (Uniform Resource Locator) of the Web service to which the request is sent.

**soapAction** : This attribute specifies 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

This element MUST NOT be present.

Parent Elements
<a href="#">hwsAdapter</a>

Child Elements
<a href="#">input</a>

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

This element specifies a SOAP message that is submitted to a Web service.

Parent Elements
<a href="#">hwsOperation</a>
<a href="#">operation</a>

Child Elements
<a href="#">partFragment</a>

Attributes:

**source** : This attribute specifies the name of the file containing the XML (Extensible Markup Language) 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

This 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
<a href="#">input</a>

Attributes:

**dataObject** : This attribute **MUST** be ignored.

**filter** : This attribute specifies an XPath expression that **MUST** evaluate to an XML (Extensible Markup Language) 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 (1).

**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 (Extensible Markup Language) 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="xsd:boolean" 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

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

Parent Elements
<a href="#">dataAdapters</a>
<a href="#">query</a>

Attributes:

**fileUrl** : This attribute specifies either the URL (Uniform Resource Locator) of an XML (Extensible Markup Language) file that is not contained in the form template or the name of an XML (Extensible Markup Language) file that is contained in the form template.

**name** : This attribute specifies 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 (1).

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="xsd:string" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

```

        <xsd:attribute name="fileUrl" type="xsd:anyURI" use="required"/>
    </xsd:complexType>
</xsd:element>

```

## 2.2.46 sharepointListAdapter

This element specifies the properties of a data adapter that **MUST** be created to query a protocol server list (1). The protocol server list (1) **MUST** be used as a secondary data source, and the protocol server list data adapter **MUST NOT** support submitting the form file.

Parent Elements
<a href="#">dataAdapters</a>
<a href="#">query</a>

Child Elements
<a href="#">field</a>

Attributes:

**infopathGroup** : This attribute specifies the name of the parent XML (Extensible Markup Language) 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** : This attributes 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 the protocol server list (1) for data. If this attribute is not present, its value **MUST** be interpreted as "yes".

**sharepointGuid** : This attribute specifies the **GUID** of the protocol server list (1).

**siteUrl** : This attribute specifies the URL (Uniform Resource Locator) of the parent protocol server **site (2)**.

**submitAllowed** : This attribute 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="xsd:boolean" use="optional"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

```

```

    </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

This 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 (1). Each protocol server list field returned from a query **MUST** be specified by an instance of this element.

Parent Elements
<a href="#">sharepointListAdapter</a>

Attributes:

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

**sharepointName** : This attribute specifies protocol server list field name that corresponds to the name of the field in the form view as 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

This element specifies the properties of a data adapter that **MUST** be created to submit a form file to a **WebDAV** server. The WebDAV data adapter **MUST NOT** support querying a WebDAV server.

Parent Elements
<a href="#">dataAdapters</a>
<a href="#">submit</a>

Child Elements
<a href="#">fileName</a>
<a href="#">folderURL</a>

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.

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

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

**submitAllowed** : This attribute 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

This element specifies the URL (Uniform Resource Locator) of a WebDAV server or protocol server to which the form file MUST be submitted.

Parent Elements
<a href="#">davAdapter</a>

Attributes:

**value** : This attribute specifies the server URL (Uniform Resource Locator). 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

This 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
<a href="#">davAdapter</a>

Attributes:

**value** : This attribute specifies 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** : This attribute 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="xs:xdExpressionLiteral" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

### 2.2.51 emailAdapter

This element specifies the information needed to submit the form (1) 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
<a href="#">dataAdapters</a>

Parent Elements
<a href="#">submit</a>

Child Elements
<a href="#">attachmentFileName</a>
<a href="#">bcc</a>
<a href="#">cc</a>
<a href="#">intro</a>
<a href="#">subject</a>
<a href="#">to</a>

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

```

        <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

This 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
<a href="#">emailAdapter</a>

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 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

This element specifies the carbon-copied (CC) recipients for the e-mail message that is generated when the form file is submitted using the e-mail data adapter.

Parent Elements
<a href="#">emailAdapter</a>

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 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

This element specifies the blind carbon-copied (BCC) recipients for the e-mail message that is generated when the form file is submitted using the e-mail data adapter.

Parent Elements
<a href="#">emailAdapter</a>

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 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="xsd:expressionLiteral" use="optional"/>
  </xsd:complexType>
</xsd:element>

```

## 2.2.55 subject

This 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
<a href="#">emailAdapter</a>

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="xsd:expressionLiteral" use="optional"/>
  </xsd:complexType>
</xsd:element>

```

## 2.2.56 intro

This 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
<a href="#">emailAdapter</a>

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

This 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
<a href="#">emailAdapter</a>

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 ("expression") or a string literal ("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="xsd:string" use="optional"/>
  </xsd:complexType>
</xsd:element>

```

## 2.2.58 submitToHostAdapter

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

Parent Elements
<a href="#">dataAdapters</a>
<a href="#">submit</a>

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**

This element specifies the secondary data connection data adapters that submit the form file to a data source (2).

Parent Elements
<a href="#">xDocumentClass</a>

Child Elements
<a href="#">adoAdapter</a>
<a href="#">davAdapter</a>
<a href="#">emailAdapter</a>
<a href="#">hwsAdapter</a>
<a href="#">sharepointListAdapter</a>
<a href="#">submitToHostAdapter</a>
<a href="#">webServiceAdapter</a>
<a href="#">xmlFileAdapter</a>

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

This 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 [\[XMLSCHEMA1\]](#), then 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 [\[XMLSCHEMA1\]](#), then a documentSchema element MUST NOT exist for the included XML schema.

Parent Elements
<a href="#">xDocumentClass</a>

Child Elements
<a href="#">documentSchema</a>

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

This 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
<a href="#">documentSchemas</a>

Attributes:

**location** : This attribute specifies the XML schema location as either only the name of the XML schema document or 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="xsd:boolean"/>
  </xsd:complexType>
</xsd:element>
```

2.2.62 customValidation

This element specifies a rule-based custom validation which is enforced in addition to the XML schema validation.

Parent Elements
<a href="#">xDocumentClass</a>

Child Elements
<a href="#">errorCondition</a>

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

2.2.63 errorCondition

This element specifies a custom validation for a set of XML nodes in the form file.

Parent Elements
<a href="#">customValidation</a>

Child Elements
<a href="#">errorMessage</a>

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 (1).

**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 (1) 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

This 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
<a href="#">errorCondition</a>
<a href="#">override</a>

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", then 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="xsd:errorMessage">
        <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

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

Parent Elements
<a href="#">xDocumentClass</a>

Child Elements
<a href="#">domEventHandler</a>

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="xsd: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

This 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
<a href="#">domEventHandlers</a>

Child Elements
<a href="#">ruleSetAction</a>

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 (1).

**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

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

Parent Elements
<a href="#">xDocumentClass</a>

Child Elements
<a href="#">importSource</a>

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

This 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 XSL Transformations (XSLT) during the merge operation.

Parent Elements
<a href="#">importParameters</a>

Attributes:

**authoringOfTransform** : This attribute specifies whether the XSL Transformations (XSLT) 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

This 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
<a href="#">xDocumentClass</a>

Child Elements
<a href="#">fields</a>

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

This 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
<a href="#">listProperties</a>

Child Elements
<a href="#">field</a>

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

This 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
<a href="#">fields</a>

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 (1). If this attribute is present, it **MUST** be set to one of the following values:

- average – The use of this value specifies that 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 use of this value specifies that the aggregate value **MUST** be determined by calculating the number of XML nodes.
- first – The use of this value specifies that the aggregate value **MUST** be determined by returning the first XML node value in the collection.
- last – The use of this value specifies that the aggregate value **MUST** be determined by returning the last XML node value in the collection.
- max – The use of this value specifies that 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 use of this value specifies that the aggregate value **MUST** be determined by concatenating all XML node values in the collection separated by newline characters (Unicode #x000D and #x000A).

- **min** – The use of this value specifies that 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 use of this value specifies that 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 use of this value specifies that 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 (2)** in the **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="xsd:xdTitle" use="required"/>
    <xsd:attribute name="columnName" type="xsd:xdTitle" use="required"/>
    <xsd:attribute name="required" type="xsd:xdYesNo" use="optional"/>
    <xsd:attribute name="viewable" type="xsd: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>
```

```

        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
</xsd:element>

```

## 2.2.72 submit

This element specifies the necessary information for configuring the submit operation for the form (1). This includes information regarding the method to use, UI elements which 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 will be submitted using a data adapter if the target for the submit operation is a data source (2) 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 will be submitted by an associated collection of **rules (1)** which execute associated actions as specified by the [rulesetAction](#) element.
- Script Handler: The form file will be submitted by associated form code as specified by the [useScriptHandler](#) element.
- HTTP Handler: The form file will be submitted using the **HTTP method** as specified by the [useHttpHandler](#) element.

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

Parent Elements
<a href="#">xDocumentClass</a>

Child Elements
<a href="#">davAdapter</a>
<a href="#">emailAdapter</a>
<a href="#">errorMessage</a>
<a href="#">ruleSetAction</a>
<a href="#">submitAction</a>
<a href="#">submitToHostAdapter</a>
<a href="#">successMessage</a>
<a href="#">useHttpHandler</a>

Child Elements
<a href="#">useQueryAdapter</a>
<a href="#">useScriptHandler</a>
<a href="#">webServiceAdapter</a>

Attributes:

**caption** : This attribute specifies the name of the submit button. A corresponding button **MUST** appear on the form view toolbar when the form (1) 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 – form (1) closes.

keepOpen – form state does not change.

openNew – form (1) resets itself to the state of a new form (1).

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

```

    <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

This element MUST NOT be present.

Parent Elements
<a href="#">submit</a>

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

This element specifies the string used to notify the user that the form (1) was submitted successfully.

Parent Elements
-----------------

Parent Elements
<a href="#">submit</a>

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

This element specifies the string used to notify the user that the form (1) was not submitted successfully.

Parent Elements
<a href="#">submit</a>

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

This element specifies that the form (1) MUST be submitted to the specified URL (Uniform Resource Locator) using the specified HTTP method.

Parent Elements
<a href="#">submit</a>

Attributes:

**href** : This attribute specifies the URL (Uniform Resource Locator) 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" [\[HTML\]](#).

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>
```

```

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

```

### 2.2.77 useScriptHandler

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

Parent Elements
<a href="#">documentVersionUpgrade</a>
<a href="#">save</a>
<a href="#">submit</a>

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

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

### 2.2.78 useQueryAdapter

This element **MUST NOT** be present.

Parent Elements
<a href="#">submit</a>

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

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

### 2.2.79 onLoad

This element specifies a set of rules (1) that is called when the form (1) is loaded.

Parent Elements
<a href="#">xDocumentClass</a>

Child Elements
<a href="#">ruleSetAction</a>

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

This element MUST NOT be present.

Parent Elements
<a href="#">xDocumentClass</a>

Child Elements
<a href="#">useScriptHandler</a>

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

This element MUST NOT be present.

Parent Elements
<a href="#">xDocumentClass</a>

Child Elements
<a href="#">membership</a>
<a href="#">role</a>

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

This element MUST NOT be present.

Parent Elements
<a href="#">roles</a>

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

This element MUST NOT be present.

Parent Elements
<a href="#">roles</a>

Child Elements
<a href="#">getUserNameFromData</a>
<a href="#">group</a>
<a href="#">userName</a>

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

This element MUST NOT be present.

Parent Elements
<a href="#">membership</a>

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**

This element **MUST NOT** be present.

Parent Elements
<a href="#">membership</a>

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**

This element **MUST NOT** be present.

Parent Elements
<a href="#">membership</a>

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

This element MUST NOT be present.

Parent Elements
<a href="#">xDocumentClass</a>

Child Elements
<a href="#">allowedActions</a>
<a href="#">allowedTasks</a>
<a href="#">location</a>

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

This element MUST NOT be present.

Parent Elements
<a href="#">hwsWorkflow</a>

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**

This element MUST NOT be present.

Parent Elements
<a href="#">hwsWorkflow</a>

Child Elements
<a href="#">action</a>

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**

This element MUST NOT be present.

Parent Elements
<a href="#">allowedActions</a>

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**

This element MUST NOT be present.

Parent Elements
<a href="#">hwsWorkflow</a>

Child Elements
<a href="#">task</a>

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**

This element MUST NOT be present.

Parent Elements
<a href="#">allowedTasks</a>

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**

This element specifies the name and location of the XML (Extensible Markup Language) template file that contains default values for a new form (1) based on the form template.

Parent Elements
<a href="#">xDocumentClass</a>

Child Elements
<a href="#">initialXmlDocument</a>

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**

This element specifies a reference to the template.xml file used for the creation of a new form (1).

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

Parent Elements
<a href="#">fileNew</a>

Child Elements
<a href="#">customCategory</a>

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

This element specifies the form template category, which is used to group together form templates.

Parent Elements
<a href="#">initialXmlDocument</a>

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

This element specifies the collection of all files in the form template.

Parent Elements
<a href="#">xDocumentClass</a>

Child Elements
<a href="#">files</a>

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

This element specifies the collection of all files in the form template, as specified by section [2.1](#), and their properties.

Parent Elements
<a href="#">package</a>

Child Elements
<a href="#">file</a>

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

This 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
<a href="#">files</a>

Child Elements
<a href="#">fileProperties</a>

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

This element specifies a collection of properties of a file in the form template

Parent Elements
<a href="#">file</a>

Child Elements
<a href="#">property</a>

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

This 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
<a href="#">fileProperties</a>

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 (see **rootAssembly**). 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" attribute.

**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 by section [2.4](#), MUST have a file property with this name.

The **value** attribute value MUST be set to the 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 by 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

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

Parent Elements
<a href="#">xDocumentClass</a>

Child Elements
<a href="#">allowedControl</a>

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

This element MUST be ignored.

Parent Elements
<a href="#">permissions</a>

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

This element MUST be ignored.

Parent Elements
<a href="#">xDocumentClass</a>

Child Elements
<a href="#">externalView</a>

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

This element **MUST** be ignored.

Parent Elements
<a href="#">externalViews</a>

Child Elements
<a href="#">mainpane</a>

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

This 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.

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

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

Parent Elements
<a href="#">menu</a>
<a href="#">menuArea</a>
<a href="#">toolbar</a>

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 by **Repeating Section Control** (section [2.3.1.11](#)), or a repeating table control, as specified by **Repeating Table Control** (section [2.4.1.16](#)).
- xOptional – An optional section control, as specified by **Section Control and Optional Section Control** (section [2.4.1.18](#)).
- xFileAttachment – A file attachment control, as specified by **File Attachment Control** (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** [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** (section [2.2.124](#)) element. 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="xsd:string"/>
    <xsd:attribute name="icon" type="xsd:string"/>
    <xsd:attribute name="tooltip" type="xsd:string"/>
    <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:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
</xsd:element>
```

```

        <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

This element specifies an **XML fragment**. An XML fragment is an XML (Extensible Markup Language) subtree that is intended to represent a unit of data. It is typically used for data insertion and replacement operations.

Parent Elements
<a href="#">fragmentToInsert</a>

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 (1).

**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:complexType>
</xsd:element>

```

```

        <xsd:attribute name="innerFragment" type="xsd:string" use="optional"/>
    </xsd:complexType>
</xsd:element>

```

## 2.2.108 editWith

This element specifies an instance of a control that edits data in the form file.

Parent Elements
<a href="#">xmlToEdit</a>

Child Elements
<a href="#">fragmentToInsert</a>
<a href="#">masterDetail</a>

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 (Extensible Markup Language) data to be used in the control. If this attribute is not present, its value MUST be interpreted as an empty string.

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

```

<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

This element specifies a collection of buttons in the form view.

Parent Elements
<a href="#">view</a>

Child Elements
<a href="#">button</a>

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

This element specifies a button that MAY have an associated event handler or [ruleSetAction](#).

Parent Elements
<a href="#">unboundControls</a>

Child Elements
<a href="#">ruleSetAction</a>

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

```

</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

This element specifies additional information about controls used in the form view to edit the form file.

Parent Elements
<a href="#">view</a>

Child Elements
<a href="#">xmlToEdit</a>

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

This element MUST NOT be present.

Parent Elements
<a href="#">editWith</a>

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**

This element specifies alternate versions of default XML (Extensible Markup Language) data that is inserted into an associated control.

Parent Elements
<a href="#">editWith</a>

Child Elements
<a href="#">chooseFragment</a>

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**

This element specifies the XSL Transformation (XSLT) file, as specified by the [XSL specification](#), included in the form template and used to represent a form view.

Parent Elements
<a href="#">externalView</a>
<a href="#">view</a>

Attributes:

**transform** : This attribute specifies the name of the XSLT file that is used to transform the form (1). 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

This element MUST be ignored.

Parent Elements
<a href="#">mergedPrintView</a>
<a href="#">view</a>

Child Elements
<a href="#">footer</a>
<a href="#">header</a>

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

This element MUST be ignored.

Parent Elements
<a href="#">printSettings</a>

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

This element MUST be ignored.

Parent Elements
<a href="#">printSettings</a>

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>

```

```

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

```

### 2.2.118 toolbar

This element MUST be ignored.

Parent Elements
<a href="#">view</a>

Child Elements
<a href="#">button</a>
<a href="#">menu</a>

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

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

Parent Elements
menu
<a href="#">menuArea</a>
<a href="#">toolbar</a>
<a href="#">view</a>

Child Elements
<a href="#">button</a>
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="xsd:UIItem" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="caption" type="xsd:Title" use="required"/>
  </xsd:complexType>
</xsd:element>
```

### 2.2.120 menuArea

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

Parent Elements
<a href="#">view</a>

Child Elements
<a href="#">button</a>
<a href="#">menu</a>

Attributes:

**name** : This attribute **MUST** be set to "msoStructuralEditingContextMenu".

msoEditMenu – Refers to the Edit menu, attribute **MUST NOT** have this value.

msoFileMenu – Refers to the File menu, attribute **MUST NOT** have this value.

msoHelpMenu – Refers to the Help menu, attribute **MUST NOT** have this value.

msoInsertMenu – Refers to the Insert menu, attribute **MUST NOT** have this value.

msoFormatMenu – Refers to the Format menu, attribute **MUST NOT** have this value.

msoStructuralEditingContextMenu – refers to the context menu for structural controls.

- Repeating Section Control (Section [2.4.1.15](#))

- Repeating Table Control (Section [2.4.1.16](#))
- Section Control and Optional Section Control (Section [2.4.1.18](#))

msoTableMenu – Refers to the Table menu, attribute MUST NOT have this value.

msoToolsMenu – Refers to the Tools menu, attribute MUST NOT have this value.

msoViewMenu – Refers to the View menu, attribute MUST NOT have this value.

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

This element MUST be ignored.

Parent Elements
<a href="#">xDocumentClass</a>

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

```

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

```

## 2.2.122 views

This 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 XSL Transformation (XSLT) files, as specified by the [XSL specification](#), 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 is set to "no".
- The **designMode** attribute of the [view](#) element is set "normal".

Parent Elements
<a href="#">xDocumentClass</a>

Child Elements
<a href="#">view</a>

Attributes:

**default** : This attribute specifies the name of the form view which 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>
</element>

```

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

### 2.2.123 view

This 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 (1) and how they are rendered using HTML. A form view is represented by an XSL Transformation (XSLT) file, as specified by the [XSL specification](#).

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

Parent Elements
<a href="#">views</a>

Child Elements
<a href="#">editing</a>
<a href="#">mainpane</a>
<a href="#">menu</a>
<a href="#">menuArea</a>
<a href="#">printSettings</a>
<a href="#">toolbar</a>
<a href="#">unboundControls</a>

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 (1).

**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

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

Parent Elements
<a href="#">editing</a>

Child Elements
<a href="#">editWith</a>

Attributes:

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

**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 (1).

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

This element specifies the digital signatures (1), as specified by [MS-IPFFX](#), that are used to sign the form file according to [XMLDSig](#).

Parent Elements
<a href="#">xDocumentClass</a>

Child Elements
<a href="#">signedDataBlock</a>

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

This element specifies a set of XML nodes in the form file that **MUST** be signed by a digital signature (1).

Parent Elements
<a href="#">documentSignatures</a>

Child Elements
<a href="#">message</a>

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 (1) signs all previous digital signatures.

cosign – The digital signature (1) is treated independently of all previous digital signatures (1).

single – The signed data block **MUST NOT** be signed by more than one digital signature (1).

**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

This element specifies the confirmation message that **MUST** be displayed before a digital signature (1) is applied to the form (1) or section of the form (1).

Parent Elements
<a href="#">signedDataBlock</a>

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

This element specifies the process by which forms (1) created based on an older version of the form template are upgraded to the latest version of the form template.

Parent Elements
<a href="#">xDocumentClass</a>

Child Elements
<a href="#">useScriptHandler</a>
<a href="#">useTransform</a>

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

This element specifies the XSLT file, as defined by section [2.8](#), and the restrictions that **MUST** be used to upgrade the form (1).

Parent Elements
<a href="#">documentVersionUpgrade</a>

Attributes:

**maxVersionToUpgrade** : This attribute specifies the inclusive value for the maximum form template version, specified by the **solutionVersion** attribute of the [xDocumentClass](#) element, 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**

This element specifies the enabled extensions in the form definition (.xsf) file. Each enabled extension MUST conform to the XML schema as defined by the [XSF extension specification](#).

Parent Elements
<a href="#">xDocumentClass</a>

Child Elements
<a href="#">extension</a>

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

This element specifies a container for an XML schema extension.

Parent Elements
<a href="#">extensions</a>

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

This element specifies the rule set, as defined by the [ruleSet](#) element that MUST be called by a form or form file event.

Parent Elements
<a href="#">button</a>
<a href="#">domEventHandler</a>
<a href="#">onLoad</a>
<a href="#">submit</a>

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

This element specifies a rule (1), which is composed of the rule definition and the event by which the rule (1) is called. The rule definition is defined by the **rule** and [ruleSet](#) elements. The event is defined by the [button](#), [domEventHandler](#), [onLoad](#), and [submit](#) elements and their associated [ruleSetAction](#) element.

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 (1) 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 (1) are grouped together as a rule set, as specified by the [ruleSet](#) element, containing one or more rules (1). 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.

Parent Elements
<a href="#">ruleSet</a>

Child Elements
<a href="#">assignmentAction</a>
<a href="#">closeDocumentAction</a>
<a href="#">dialogBoxExpressionAction</a>
<a href="#">dialogBoxMessageAction</a>
<a href="#">exitRuleSet</a>
<a href="#">openNewDocumentAction</a>
<a href="#">queryAction</a>
<a href="#">submitAction</a>
<a href="#">switchViewAction</a>

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

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

Parent Elements
<a href="#">rule</a>

Attributes:

**adapter** : This attribute specifies the name of the corresponding data adapter that will be 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="xsd:xdTitle" use="required"/>
</xsd:complexType>
</xsd:element>

```

### 2.2.135 exitRuleSet

This element specifies that rule processing **MUST** stop for the entire rule set.

Parent Elements
<a href="#">rule</a>

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

This element **MUST** be ignored.

Parent Elements
<a href="#">rule</a>

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

This element **MUST** be ignored.

Parent Elements
<a href="#">rule</a>

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

This element specifies the form view that **MUST** be shown when called by a form event.

Parent Elements
<a href="#">rule</a>

Attributes:

**view** : This attribute specifies the name of the form view that will be shown. The specified name **MUST** match an existing **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="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

This element specifies an action that **MUST** set the value of a field.

Parent Elements
<a href="#">rule</a>

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

This element specifies a data adapter that **MUST** query its data source (2) when called by a form action.

Parent Elements
<a href="#">query</a>
<a href="#">rule</a>

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

This element **MUST NOT** be present.

Parent Elements
<a href="#">rule</a>

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

This element specifies an action to close the form (1).

Parent Elements
<a href="#">rule</a>

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

This element specifies a set of one or more rules (1) for the form (1).

Parent Elements
<a href="#">ruleSets</a>

Child Elements
<a href="#">rule</a>

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

This element specifies the rule sets for the form (1).

Parent Elements
<a href="#">xDocumentClass</a>

Child Elements
----------------

Child Elements
<a href="#">ruleSet</a>

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

This element specifies definitions for the calculations performed in the form (1) and how blank values are handled.

Parent Elements
<a href="#">xDocumentClass</a>

Child Elements
<a href="#">calculatedField</a>

Attributes:

**treatBlankValueAsZero** : This attribute specifies whether an empty string (1) is equivalent to the integer 0. 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

This element specifies an individual calculation, when the calculation is to be performed, and where there the result will be stored.

Parent Elements
<a href="#">calculations</a>

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 as 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
<a href="#">compatibilityModesType</a>	<a href="#">2.2.147.3</a>
<a href="#">emailAttachmentType</a>	<a href="#">2.2.147.2</a>
<a href="#">formDescriptionType</a>	<a href="#">2.2.147.5</a>
<a href="#">formLocaleType</a>	<a href="#">2.2.147.6</a>

Type	Specified in Section
<a href="#">managedCodeType</a>	<a href="#">2.2.147.7</a>
<a href="#">serverCommandActionType</a>	<a href="#">2.2.147.1</a>
<a href="#">solutionType</a>	<a href="#">2.2.147.4</a>
Element	Specified in Section
<a href="#">admin</a>	<a href="#">2.2.147.20</a>
<a href="#">adoAdapterExtension</a>	<a href="#">2.2.147.32</a>
<a href="#">autoUpdatePrompt</a>	<a href="#">2.2.147.45</a>
<a href="#">command</a>	<a href="#">2.2.147.12</a>
<a href="#">commands</a>	<a href="#">2.2.147.11</a>
<a href="#">connectoid</a>	<a href="#">2.2.147.30</a>
<a href="#">contentType</a>	<a href="#">2.2.147.16</a>
<a href="#">contentTypeTemplate</a>	<a href="#">2.2.147.17</a>
<a href="#">dataConnections</a>	<a href="#">2.2.147.28</a>
<a href="#">davAdapterExtension</a>	<a href="#">2.2.147.31</a>
<a href="#">emailAdapterExtension</a>	<a href="#">2.2.147.35</a>
<a href="#">errorMessage</a>	<a href="#">2.2.147.54</a>
<a href="#">exportToPDFForXPS</a>	<a href="#">2.2.147.56</a>
<a href="#">featureRestrictionsExtension</a>	<a href="#">2.2.147.55</a>
<a href="#">fieldExtension</a>	<a href="#">2.2.147.27</a>
<a href="#">fieldsExtension</a>	<a href="#">2.2.147.26</a>
<a href="#">includedView</a>	<a href="#">2.2.147.23</a>
<a href="#">includedViews</a>	<a href="#">2.2.147.22</a>
<a href="#">inputScope</a>	<a href="#">2.2.147.47</a>
<a href="#">inputScopes</a>	<a href="#">2.2.147.46</a>
<a href="#">install</a>	<a href="#">2.2.147.14</a>
<a href="#">listPropertiesExtension</a>	<a href="#">2.2.147.25</a>
<a href="#">mail</a>	<a href="#">2.2.147.19</a>
<a href="#">managedCode</a>	<a href="#">2.2.147.50</a>
<a href="#">mergedPrintView</a>	<a href="#">2.2.147.21</a>
<a href="#">offline</a>	<a href="#">2.2.147.24</a>

Type	Specified in Section
<a href="#">preview</a>	<a href="#">2.2.147.44</a>
<a href="#">relativeQuery</a>	<a href="#">2.2.147.34</a>
<a href="#">sendByMail</a>	<a href="#">2.2.147.38</a>
<a href="#">server</a>	<a href="#">2.2.147.9</a>
<a href="#">share</a>	<a href="#">2.2.147.18</a>
<a href="#">sharepointListAdapterExtension</a>	<a href="#">2.2.147.37</a>
<a href="#">solutionDefinition</a>	<a href="#">2.2.147.8</a>
<a href="#">solutionPropertiesExtension</a>	<a href="#">2.2.147.13</a>
<a href="#">submit</a>	<a href="#">2.2.147.51</a>
<a href="#">submitAction</a>	<a href="#">2.2.147.52</a>
<a href="#">successMessage</a>	<a href="#">2.2.147.53</a>
<a href="#">toolbar</a>	<a href="#">2.2.147.10</a>
<a href="#">useHttpHandlerExtension</a>	<a href="#">2.2.147.29</a>
<a href="#">viewExtension</a>	<a href="#">2.2.147.42</a>
<a href="#">viewsExtension</a>	<a href="#">2.2.147.41</a>
<a href="#">warning</a>	<a href="#">2.2.147.40</a>
<a href="#">warnings</a>	<a href="#">2.2.147.39</a>
<a href="#">webServiceAdapterExtension</a>	<a href="#">2.2.147.33</a>
<a href="#">word</a>	<a href="#">2.2.147.49</a>
<a href="#">words</a>	<a href="#">2.2.147.48</a>
<a href="#">wss</a>	<a href="#">2.2.147.15</a>
<a href="#">xmlFileAdapterExtension</a>	<a href="#">2.2.147.36</a>
<a href="#">xmlToEditExtension</a>	<a href="#">2.2.147.43</a>

### 2.2.147.1 serverCommandActionType

This simple type specifies restrictions for specifying a form action on the form toolbar.

Referenced By
<a href="#">command.xsf2@action</a>

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

This simple type specifies restrictions for specifying the file format of an attached form (1) or form template when it is sent in e-mail.

Referenced By
<a href="#">emailAdapterExtension.xsf2@emailAttachmentType</a>
<a href="#">sendByMail.xsf2@emailAttachmentType</a>

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

This simple type specifies restrictions for specifying the compatibility mode for the form template.

Referenced By
<a href="#">solutionDefinition.xsf2@runtimeCompatibility</a>

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

This simple type specifies restrictions for an attribute that, if present, MUST be ignored.

Referenced By
<a href="#">solutionDefinition.xsf2@solutionType</a>

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

This simple type specifies restrictions for specifying the form template description.

Referenced By
<a href="#">solutionDefinition.xsf2@description</a>

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

This simple type specifies restrictions for specifying the locale of the form template.

Referenced By
<a href="#">server.xsf2@formLocale</a>

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

This simple type specifies restrictions for specifying the business objects programming language used in the form template.

Referenced By
<a href="#">managedCode.xsf2@language</a>

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**

This 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 by the [XSF specification](#) or [XSF extension specification](#). 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
<a href="#">autoUpdatePrompt</a>
<a href="#">dataConnections</a>
<a href="#">featureRestrictionsExtension</a>
<a href="#">inputScopes</a>
<a href="#">listPropertiesExtension</a>
<a href="#">managedCode</a>
<a href="#">mergedPrintView</a>
<a href="#">offline</a>
<a href="#">preview</a>
<a href="#">sendByMail</a>
<a href="#">server</a>
<a href="#">solutionPropertiesExtension</a>
<a href="#">submit</a>
<a href="#">viewsExtension</a>
<a href="#">warnings</a>

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 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: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>
```

## 2.2.147.9 server

This element specifies display and functional properties for the form template.

Parent Elements
<a href="#">solutionDefinition</a>

Child Elements
----------------

Child Elements
<a href="#">toolbar</a>

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 (1) to be loaded in a mobile Web browser.

**isPreSubmitPostBackEnabled** : This attribute specifies whether the Web browser MUST **postback** the form (1) 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: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

This element specifies information about the toolbar that is displayed when a form (1) is loaded.

Parent Elements
<a href="#">server</a>

Child Elements
<a href="#">commands</a>

Attributes:

**enabledBottom** : This attribute specifies whether the toolbar MUST be displayed at the bottom of the form (1).

**enabledTop** : **enabledBottom** : 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**

This element contains commands that are displayed on visible toolbars when a form (1) is loaded.

Parent Elements
<a href="#">toolbar</a>

Child Elements
<a href="#">command</a>

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**

This element specifies a command that **MUST** be displayed on the toolbar when a form (1) is opened.

Parent Elements
<a href="#">commands</a>

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

### 2.2.147.13 solutionPropertiesExtension

This element MUST be ignored.

Parent Elements
<a href="#">solutionDefinition</a>

Child Elements
<a href="#">admin</a>
<a href="#">contentType</a>
<a href="#">contentTypeTemplate</a>
<a href="#">install</a>
<a href="#">mail</a>
<a href="#">share</a>
<a href="#">wss</a>

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

This element MUST be ignored.

Parent Elements
<a href="#">solutionPropertiesExtension</a>

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

```

    <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

This element MUST be ignored.

Parent Elements
<a href="#">solutionPropertiesExtension</a>

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="xs:boolean" use="optional"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>

```

### 2.2.147.16 contentType

This element MUST be ignored.

Parent Elements
<a href="#">solutionPropertiesExtension</a>

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

This element MUST be ignored.

Parent Elements
<a href="#">solutionPropertiesExtension</a>

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="xs:boolean" use="optional"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>

```

## 2.2.147.18 share

This element MUST be ignored.

Parent Elements
<a href="#">solutionPropertiesExtension</a>

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

This element MUST be ignored.

Parent Elements
<a href="#">solutionPropertiesExtension</a>

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

This element MUST be ignored.

Parent Elements
<a href="#">solutionPropertiesExtension</a>

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**

This element MUST be ignored.

Parent Elements
<a href="#">solutionDefinition</a>

Child Elements
<a href="#">includedViews</a>
<a href="#">printSettings</a>

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**

This element MUST be ignored.

Parent Elements
<a href="#">mergedPrintView</a>

Child Elements
<a href="#">includedView</a>

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

This element MUST be ignored.

Parent Elements
<a href="#">includedViews</a>

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

This element MUST be ignored.

Parent Elements
<a href="#">solutionDefinition</a>

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="xsd:boolean" default="no" use="optional"/>
    <xsd:attribute name="cacheQueries" type="xsd:boolean" 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**

This element MUST be ignored.

Parent Elements
<a href="#">solutionDefinition</a>

Child Elements
<a href="#">fieldsExtension</a>

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="xsd2:fieldsExtension" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

**2.2.147.26 fieldsExtension**

This element MUST be ignored.

Parent Elements
<a href="#">listPropertiesExtension</a>

Child Elements
<a href="#">fieldExtension</a>

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

This element **MUST** be ignored.

Parent Elements
<a href="#">fieldsExtension</a>

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

This element contains elements that specify extensions to data adapter connection settings.

Parent Elements
<a href="#">solutionDefinition</a>

Child Elements
<a href="#">adoAdapterExtension</a>
<a href="#">davAdapterExtension</a>
<a href="#">emailAdapterExtension</a>
<a href="#">sharepointListAdapterExtension</a>
<a href="#">useHttpHandlerExtension</a>
<a href="#">webServiceAdapterExtension</a>
<a href="#">xmlFileAdapterExtension</a>

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

This element specifies extended information for [useHttpHandler](#).

Parent Elements
<a href="#">dataConnections</a>

Child Elements
<a href="#">connectoid</a>

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

This 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 Universal Data Connection (.udc, .udcx) 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, to access specific data sources (2) 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
<a href="#">adoAdapterExtension</a>
<a href="#">davAdapterExtension</a>
<a href="#">sharepointListAdapterExtension</a>
<a href="#">useHttpHandlerExtension</a>
<a href="#">webServiceAdapterExtension</a>
<a href="#">xmlFileAdapterExtension</a>

Attributes:

**connectionLinkType** : This attribute specifies the location context of the Universal Data Connection (.udc, .udcx) 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 (Uniform Resource Locator) to the root of the site collection where the original Universal Data Connection (.udc, .udcx) file is located. This value MUST be identical for all **connectoid** elements in the form template.

**source** : This attribute specifies a URL (Uniform Resource Locator) that specifies the relative path from the **siteCollection** attribute to the Universal Data Connection (.udc, .udcx) 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**

This element specifies the extended information for [davAdapter](#).

Parent Elements
<a href="#">dataConnections</a>

Child Elements
<a href="#">connectoid</a>

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**

This element specifies extended information for [adoAdapter](#).

Parent Elements
<a href="#">dataConnections</a>

Child Elements
<a href="#">connectoid</a>

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

This element specifies extended information for [webServiceAdapter](#). This element MUST NOT have both a [connectoid](#) child element and a [relativeQuery](#) child element present.

Parent Elements
<a href="#">dataConnections</a>

Child Elements
<a href="#">connectoid</a>
<a href="#">relativeQuery</a>

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

This element specifies a substring of the specified Web service URL (Uniform Resource Locator) that is replaced at run time by a different substring to create a new Web service URL (Uniform Resource Locator). This element is used when hosting a form (1) 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 (Uniform Resource Locator) substring MUST be replaced by the value specified by an implementation-specific **ASP.NET control** hosting the form (1).

Parent Elements
<a href="#">webServiceAdapterExtension</a>

Attributes:

**replace** : This attribute specifies the substring of the Web service URL (Uniform Resource Locator) that will be replaced at run time. The specified URL (Uniform Resource Locator) MUST be an absolute path and MUST NOT be a local or UNC path. The specified URL (Uniform Resource Locator) MUST match the beginning of the value specified by the **serviceUrl** attribute of the [operation](#) element.

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

This element specifies the method of submitting the form file using the e-mail data adapter.

Parent Elements
<a href="#">dataConnections</a>

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 (Extensible Markup Language) file.

xmlXsn – This value specifies that the form file and form template (.xsn) file are both attached as two separate attachments to the e-mail.

**ref** : This attribute specifies the associated [emailAdapter](#) element 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:emailAttachmentType"
      use="required"/>
  </xsd:complexType>
</xsd:element>
```

## 2.2.147.36 xmlFileAdapterExtension

This element specifies extended information for [xmlFileAdapter](#).

Parent Elements
<a href="#">dataConnections</a>

Child Elements
<a href="#">connectoid</a>

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

This element specifies extended information for [sharepointListAdapter](#).

Parent Elements
<a href="#">dataConnections</a>

Child Elements
<a href="#">connectoid</a>

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 (1) for values applicable only to the current form (1).

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

```

</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

This 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
<a href="#">solutionDefinition</a>

Attributes:

**disableEmailForms** : This attribute specifies the MIME type of the attached form file or form template.

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.

If this attribute is not present, its value MUST be interpreted as "no".

**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:emailAttachmentType"
      use="optional"/>
    <xsd:attribute name="disableEmailForms" type="xsf:xdYesNo" use="optional"/>
  </xsd:complexType>
</xsd:element>

```

### 2.2.147.39 warnings

This element MUST be ignored.

Parent Elements
<a href="#">solutionDefinition</a>

Child Elements
<a href="#">warning</a>

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

This element MUST be ignored.

Parent Elements
<a href="#">warnings</a>

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

his element contains extended information for the form views in this form template.

Parent Elements
<a href="#">solutionDefinition</a>

Child Elements
<a href="#">viewExtension</a>

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

This element specifies extended information for [view](#).

Parent Elements
<a href="#">viewsExtension</a>

Child Elements
<a href="#">xmlToEditExtension</a>

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

```

        <xsd:anyAttribute processContents="skip"/>
    </xsd:complexType>
</xsd:element>

```

## 2.2.147.43 xmlToEditExtension

This element specifies extended information for [xmlToEdit](#).

Parent Elements
<a href="#">viewExtension</a>

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 the 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

This element MUST be ignored.

Parent Elements
<a href="#">solutionDefinition</a>

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**

This element MUST be ignored.

Parent Elements
<a href="#">solutionDefinition</a>

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="xs:boolean" use="optional"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

**2.2.147.46 inputScopes**

This element MUST be ignored.

Parent Elements
<a href="#">solutionDefinition</a>

Child Elements
<a href="#">inputScope</a>

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

This element MUST be ignored.

Parent Elements
<a href="#">inputScopes</a>

Child Elements
<a href="#">words</a>

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

This element MUST be ignored.

Parent Elements
<a href="#">inputScope</a>

Child Elements
<a href="#">word</a>

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

This element **MUST** be ignored.

Parent Elements
<a href="#">words</a>

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

This 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
-----------------

Parent Elements
<a href="#">solutionDefinition</a>

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: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

This element MUST be ignored.

Parent Elements
<a href="#">solutionDefinition</a>

Child Elements
<a href="#">errorMessage</a>
<a href="#">submitAction</a>
<a href="#">successMessage</a>

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

This element MUST be ignored.

Parent Elements
<a href="#">submit</a>

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

This element MUST be ignored.

Parent Elements
<a href="#">submit</a>

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

This element MUST be ignored.

Parent Elements
<a href="#">submit</a>

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

This element MUST be ignored.

Parent Elements
<a href="#">solutionDefinition</a>

Child Elements
<a href="#">exportToPDForXPS</a>

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 exportToPDFForXPS

This element MUST be ignored.

Parent Elements
<a href="#">featureRestrictionsExtension</a>

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="exportToPDFForXPS">
  <xsd:complexType>
    <xsd:attribute name="ui" type="xsd:boolean" 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 of the form definition (.xsf) file in section [2.2.61](#).

### 2.3.1 Control Representation

Each control that is bound to a field or group (1) has a particular representation in the 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 <a href="#">2.3.1.1</a> )	Specifies the representation of a button control in the XML schema document.
Check Box Control (section <a href="#">2.3.1.2</a> )	Specifies the representation of a check box control in the XML schema document.
Contact Selector Control (section <a href="#">2.3.1.3</a> )	Specifies the representation of a contact selector control in the XML schema document.
Date Picker Control (section <a href="#">2.3.1.4</a> )	Specifies the representation of a date picker control in the XML schema document.
Drop-Down List Control (section <a href="#">2.3.1.5</a> )	Specifies the representation of a drop-down control in the XML schema document.
Expression Box Control (section <a href="#">2.3.1.6</a> )	Specifies the representation of an expression box control in

Section	Description
	the XML schema document.
File Attachment Control (section <a href="#">2.3.1.7</a> )	Specifies the representation of a file attachment control in the XML schema document.
Hyperlink Control (section <a href="#">2.3.1.8</a> )	Specifies the representation of a hyperlink control in the XML schema document.
List Box Control (section <a href="#">2.3.1.9</a> )	Specifies the representation of a list box control in the XML schema document.
Option Button Control (section <a href="#">2.3.1.10</a> )	Specifies the representation of an option button control in the XML schema document.
Repeating Section Control (section <a href="#">2.3.1.11</a> )	Specifies the representation of a repeating section control in the XML schema document.
Repeating Table Control (section <a href="#">2.3.1.12</a> )	Specifies the representation of a repeating table control in the XML schema document.
Rich Text Box Control (section <a href="#">2.3.1.13</a> )	Specifies the representation of a rich text box control in the XML schema document.
Section Control and Optional Section Control section ( <a href="#">2.3.1.14</a> )	Specifies the representation of a section control in the XML schema document.
Table Control (section <a href="#">2.3.1.15</a> )	Specifies the representation of a table control in the XML schema document.
Text Box Control (section <a href="#">2.3.1.16</a> )	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 [\[XMLSCHEMA1\]](#) representation.

### 2.3.1.2 Check Box Control

A check box control that is not required to contain data, for example which is bound to an **XML element**, named "field1" with data type set to "boolean" for which no constraining facets [\[XMLSCHEMA1\]](#) have been set MUST have the following XSD [\[XMLSCHEMA1\]](#) definition:

```
<xsd:element name="field1" nillable="true" type="xsd:boolean"/>
```

A check box control that is required to contain data, for example which is bound to an XML element, named "field1" with data type set to "boolean" for which no constraining facets [\[XMLSCHEMA1\]](#) have been set MUST have the following XSD [\[XMLSCHEMA1\]](#) definition:

```
<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 [\[XMLSCHEMA1\]](#) for which any valid constraining facets [\[XMLSCHEMA1\]](#) 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 (1), for example named "group1", MUST have a complex type XSD schema definition, as specified in [\[XMLSCHEMA1\]](#), as shown in the following example:

```
<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 which is bound to an XML element, named "field1" with data type set to "date" for which no constraining facets [\[XMLSCHEMA1\]](#) have been set MUST have an XSD [\[XMLSCHEMA1\]](#) schema definition:

```
<xsd:element name="field1" nillable="true" type="xsd:date"/>
```

A date picker control that is required to contain data, for example which is bound to an XML element, named "field1" with data type set to "date" for which no constraining facets [\[XMLSCHEMA1\]](#) have been set MUST have the following XSD [\[XMLSCHEMA1\]](#) schema definition:

```
<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 [\[XMLSCHEMA1\]](#) for which any valid constraining facets [\[XMLSCHEMA1\]](#) 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 which is bound to an XML element, named "field1" with data type set to "string" for which no constraining facets [\[XMLSCHEMA1\]](#) have been set MUST have the following XSD [\[XMLSCHEMA1\]](#) definition:

```
<xsd:element name="field1" type="xsd:string"/>
```

A drop-down list control that is required to contain data, for example which is bound to an XML element, named "field1" with data type set to "string" for which a **xsd:minLength** constraining facet [\[XMLSCHEMA1\]](#) has been set MUST have the following XSD [\[XMLSCHEMA1\]](#) schema definition:

```
<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 [\[XMLSCHEMA1\]](#) for which any valid constraining facets [\[XMLSCHEMA1\]](#) 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 which 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 (1).

### 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 [\[XMLSCHEMA1\]](#) have been set MUST have the following XSD [\[XMLSCHEMA1\]](#) definition:

```
<xsd:element name="field1" nillable="true" type="xsd:base64Binary"/>
```

A file attachment control that is required to contain data, for example which is bound to an XML element, named "field1", with data type set to "xsd:base64Binary" for which a **xsd:minLength** constraining facet [\[XMLSCHEMA1\]](#) has been set MUST have the following XSD [\[XMLSCHEMA1\]](#) definition:

```
<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 one of the following XSD data types [\[XMLSCHEMA1\]](#) for which any valid constraining facets [\[XMLSCHEMA1\]](#) MAY also be set:

```
base64Binary
```

### 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 to 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 (1).

### 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 [\[XMLSCHEMA1\]](#) have been set MUST have the following XSD [\[XMLSCHEMA1\]](#) definition:

```
<xsd:element name="field1" type="xsd:string"/>
```

A list box control that is required to contain data, for example which is bound to an XML element, named "field1", with data type set to "string" for which an **xsd:minLength** constraining facet [\[XMLSCHEMA1\]](#) has been set MUST have the following XSD [\[XMLSCHEMA1\]](#) definition:

```
<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 [\[XMLSCHEMA1\]](#) for which any valid constraining facets [\[XMLSCHEMA1\]](#) 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 which is bound to an XML element, named "field1", with data type set to "string" for which no constraining facets [\[XMLSCHEMA1\]](#) have been set MUST have the following XSD [\[XMLSCHEMA1\]](#) definition:

```
<xsd:element name="field1" type="xsd:string"/>
```

An option button control that is required to contain data, for example which is bound to an XML element, named "field1", with data type set to "string" for which a minLength constraining facet [\[XMLSCHEMA1\]](#) has been set MUST have the following XSD [\[XMLSCHEMA1\]](#) definition:

```
<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 [\[XMLSCHEMA1\]](#) for which any valid constraining facets [\[XMLSCHEMA1\]](#) 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, MUST have the following complex type XSD [\[XMLSCHEMA1\]](#) definition:

```
<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 [\[XMLSCHEMA1\]](#) have been set MUST have the following XSD [\[XMLSCHEMA1\]](#) definition:

```
<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", "field3" with data types set to "string" for which no constraining facets [\[XMLSCHEMA1\]](#) have been set MUST have the following XSD [\[XMLSCHEMA1\]](#) definition:

```
<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", MUST have the following complex type XSD [\[XMLSCHEMA1\]](#) definition:

```

<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 (1), for example named "group1" and containing no other bound controls, MUST have the following complex type XSD [\[XMLSCHEMA1\]](#) definition:

```

<xsd:element name="group1">
    <xsd:complexType>
        <xsd:sequence/>
    </xsd:complexType>
</xsd:element>

```

A section control or optional section control bound to a group (1), 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 [\[XMLSCHEMA1\]](#) have been set MUST have the following XSD [\[XMLSCHEMA1\]](#) definition:

```

<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 [\[XMLSCHEMA1\]](#) representation.

### 2.3.1.16 Text Box Control

A text box control that is not required to contain data, for example which is bound to an XML element, named "field1", with data type set to "string" for which no constraining facets [\[XMLSCHEMA1\]](#) have been set MUST have the following XSD [\[XMLSCHEMA1\]](#) definition:

```
<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 [\[XMLSCHEMA1\]](#) has been set MUST have the following XSD [\[XMLSCHEMA1\]](#) definition:

```
<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 [\[XMLSCHEMA1\]](#) for which any valid constraining facets [\[XMLSCHEMA1\]](#) 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:

- **Section [2.4.1 View Representation](#).** It specifies a valid XSLT transformation of the form data into HTML.
- **Section [2.4.2 Control-specific Attributes](#).** It specifies the HTML representation of certain control properties and behaviors.

- **Section 3.4.3 XSL Function Extensions.** It specifies extensions to XSLT used in transforming the form data to HTML.

## 2.4.1 View Representation

Each control MUST have the representation as 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 <a href="#">2.4.1.1</a> )	Specifies the syntax used to represent the XSLT file
XSL Root Template (section <a href="#">2.4.1.2</a> )	Specifies the starting element that contains the XSLT file
XSL Root Template Style Sheets (section <a href="#">2.4.1.3</a> )	Specifies the representation for the style sheet
Control Data Formatting (section <a href="#">2.4.1.4</a> )	Specifies the representation of data formatting for multiple controls
Button Control (section <a href="#">2.4.1.5</a> )	Specifies the representation of a button control
Check Box Control (section <a href="#">2.3.1.2</a> )	Specifies the representation of a check box control
Contact Selector Control (section <a href="#">2.4.1.7</a> )	Specifies the representation of a contact selector control
Date Picker Control (section <a href="#">2.4.1.8</a> )	Specifies the representation of a date picker control
Drop-Down List Control (section <a href="#">2.4.1.9</a> )	Specifies the representation of a drop-down list control
Expression Box Control (section <a href="#">3.4.1.6</a> )	Specifies the representation of an expression box control
File Attachment Control (section <a href="#">2.4.1.11</a> )	Specifies the representation of a file attachment control
Hyperlink Control (section <a href="#">2.4.1.12</a> )	Specifies the representation of a hyperlink control
List Box Control (section <a href="#">2.4.1.13</a> )	Specifies the representation of a list box control
Option Button Control (section <a href="#">2.4.1.14</a> )	Specifies the representation of an option button control
Repeating Section Control (section <a href="#">2.4.1.15</a> )	Specifies the representation of a repeating section control
Repeating Table Control (section <a href="#">2.4.1.16</a> )	Specifies the representation of a table control
Rich Text Box Control (section <a href="#">2.3.1.13</a> )	Specifies the representation of a rich text box control
Section Control and Optional Section Control (section <a href="#">2.4.1.18</a> )	Specifies the representation of a section control
Table Control (section <a href="#">2.4.1.19</a> )	Specifies the representation of a table control

Section	Description
Text Box Control (section <a href="#">2.4.1.20</a> )	Specifies the representation of a text box control
Ignored Controls (section <a href="#">2.4.1.21</a> )	Specifies a list (1) of controls that are ignored
Invalid Controls (section <a href="#">2.4.1.22</a> )	Specifies a list (1) of controls that are invalid
Invalid Constructs (section <a href="#">2.4.1.23</a> )	Specifies a list (1) of invalid constructs for the XSLT

### 2.4.1.1 View Syntax

The formal grammar of view XSL file is given in this specification using an extended Backus-Naur Form (**Extended Backus-Naur Form (EBNF)**) notation. The EBNF notation is used instead of ABNF or XML schema to enhance the clarity of the constructs used in the XSLT transformation.

#### Notation

Each rule (1) in the grammar defines one symbol, in the form (1).

**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 (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 (1).

**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 which does not match any production specified earlier MUST be interpreted as a literal. Additionally any construct which 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 specified 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"/>
```

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 the [postbackModel](#) section of this document.

**CONTROL\_ID**::= (**LATIN\_CHARACTER**) (**LATIN\_CHARACTER**|**ISO\_646\_DIGIT**|\_)\*

The value of **CONTROL\_ID** MUST be a valid value for type "xsd:xdTitle".

**TEMPLATE\_MODE\_ID**::= \_(**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 which contains at least one char.

**XML\_NAMESPACE**::= values MUST be as specified in [\[XML Namespaces\]](#)

**INPUT\_SCOPE\_ID**::= **ANY\_STRING**

**INPUT\_SCOPE\_NAME**::= **ANY\_STRING**

**INPUT\_SCOPE ::=**

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 the [inputScopeId](#), [inputScope](#) and [allowNonMatching](#) sections of this document.

**ALIGN** ::= left | right

**VALIGN** ::= 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](#) entry in the XSF file.

**TAB\_INDEX**<sub><4></sub> ::= -1 | 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 by [\[HTML\]](#), 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** ::= DISPLAY: none as specified by [\[CSS-LEVEL1\]](#), section 5.6.1.

**STYLE\_MARGIN** ::= value pairs MUST be as specified by [\[CSS-LEVEL1\]](#), sections 5.5.1, 5.5.2, 5.5.3, 5.5.4, 5.5.5.

**STYLE\_PADDING** ::= value pairs MUST be as specified by [\[CSS-LEVEL1\]](#), 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 by [\[CSS-LEVEL1\]](#), section 5.4.3.

**STYLE\_BACKGROUND\_COLOR** ::= value pairs MUST be as specified by [\[CSS-LEVEL1\]](#), section 5.3.2.

**STYLE\_BORDER** ::= value pairs MUST be as specified by [\[CSS-LEVEL1\]](#), 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 by [\[CSS-LEVEL2\]](#), section 8.5.3.

**STYLE\_BORDER\_COLLAPSE::=** value pairs MUST be as specified by [\[CSS-LEVEL2\]](#), section 17.6.

**STYLE\_FONT::=** value pairs MUST be as specified by [\[CSS-LEVEL1\]](#), section 5.2.

**STYLE\_FONT\_STYLE::=** value pairs MUST be as specified by [\[CSS-LEVEL1\]](#), section 5.2.3.

**STYLE\_COLOR::=** value pairs MUST be as specified by [\[CSS-LEVEL1\]](#), section 5.3.1.

**STYLE\_FONT\_WEIGHT::=** value pairs MUST be as specified by [\[CSS-LEVEL1\]](#), section 5.2.5.

**STYLE\_TEXT\_ALIGN::=** value pairs MUST be as specified by [\[CSS-LEVEL1\]](#), section 5.4.6

**STYLE\_WRAP::=** WHITE-SPACE: normal | WHITE-SPACE: nowrap; WORD-WRAP: normal

**STYLE\_OVERFLOW::=** value pairs MUST be as specified by [\[CSS-LEVEL2\]](#), 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](#).

**LEAF\_XPATH::=** MUST be an extended location XPath, as specified in [\[XPATH\]](#), which MUST NOT contain XPath predicates ([\[XPATH\]](#), section 2.4 Predicates) and MUST use only the child and attribute XPath axes ([\[XPATH\]](#), section 2.2 Axes). The generated node-set MUST have only 1 element. The XPath MUST be a relative or absolute XPath. To alter the context of the XPath evaluation, the [xdXDocument:GetDOM](#) function MUST be used before the first XPath step.

**GROUP\_XPATH::=** MUST be an extended location XPath which MUST NOT contain XPath predicates, as specified in [\[XPATH\]](#), section 2.4 Predicates, and MUST use only the child XPath axes, specified in [\[XPATH\]](#), section 2.2 Axes. There are no restrictions for the number of elements in the generated node-set. The XPath MUST be a relative or absolute XPath. To alter the context of the XPath evaluation, the [xdXDocument:GetDOM](#) function MUST be used before the first XPath step.

**RELATIVE\_REPEATING\_GROUP\_XPATH::=** MUST be a location XPath which MUST NOT contain XPath predicates, as specified in [\[XPATH\]](#), section 2.4 Predicates, and MUST use only the child XPath axes, as specified in [\[XPATH\]](#), section 2.2 Axes. There are no restrictions for the number of elements in the generated node-set. The XPath MUST be a relative XPath.

**RELATIVE\_GROUP\_XPATH::=** MUST be an extended location XPath which MUST NOT contain XPath predicates ([\[XPATH\]](#), section 2.4 Predicates) and MUST use only the child XPath axes ([\[XPATH\]](#), section 2.2 Axes). There are no restrictions for the number of elements in the generated node-set. The XPath MUST be a relative XPath.

**RELATIVE\_LEAF\_XPATH::=** MUST be a location XPath which MUST NOT contain XPath predicates ([\[XPATH\]](#), section 2.4 Predicates) and MUST use only the child and attribute XPath axes ([\[XPATH\]](#), section 2.2 Axes). The XPath MUST be a relative XPath.

**BOOLEAN\_XPATH\_EXPRESSION**::= MUST be an XPath expression which yields an object which MUST be a Boolean basic type. Boolean is specified in [\[XPATH\]](#), section 3.4. To extend the syntax of XPath expression the XSL Function Extensions specified in Section 2.4.3 MUST be used. The XPath MUST be less than 100 in depth. It MUST NOT use position and last functions specified in [\[XPATH\]](#) section 4.1. It MUST NOT use XPath Predicates. XPath Predicates are specified in [\[XPATH\]](#), section 2.4.

**STRING\_XPATH\_EXPRESSION**::= MUST be an XPath expression which yields an object which 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 MUST be less than 100 in depth. It MUST NOT use position and last functions specified in [\[XPATH\]](#) section 4.1. It MUST NOT use XPath Predicates. XPath Predicates are 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 Extended Backus-Naur Form (EBNF) rules (1) 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/SignatureProperties")?

(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:xsdf="http://schemas.microsoft.com/office/infopath/2003/solutionDefinition")?
(xmlns:xsdf2="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, then it MUST conform to one of the control productions specified for controls in sections 2.4.1.5 to 2.4.1.20. If the fragment contains an XSL element with the syntax of **SECTION\_CALL**, then it MUST be located only in the locations where a **<DIV/>** element ([\[HTML\]](#), 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 [File](#).

### 2.4.1.3 XSL Root Template Style Sheets

The following rules (1) 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: #efff6
}
.primaryVeryDark {
COLOR: #efff6; BACKGROUND-COLOR: #000000
}
.primaryDark {
COLOR: white; BACKGROUND-COLOR: #585867
}
.primaryMedium {
COLOR: black; BACKGROUND-COLOR: #dadae1
}
.primaryLight {
COLOR: black; BACKGROUND-COLOR: #efff6
}
.accentDark {
COLOR: white; BACKGROUND-COLOR: #585867
}
.accentLight {
COLOR: black; BACKGROUND-COLOR: #efff6
}
```

```

</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 (1) that MUST be used for formatting data in controls.

**DATA\_FMT\_LOCALE\_VAL** ::= MUST be a locale identifier value, as specified in [\[MS-LCID\]](#).

**DATA\_FMT\_LOCALE** ::= locale:**DATA\_FMT\_LOCALE\_VAL**

**DATA\_FMT\_NUM\_DIGITS** ::= [0-9] | auto See numDigits in [datafmt](#) section.

**DATA\_FMT\_GROUPING** ::= -1 | [0-9] | 32 See grouping in datafmt section.

**DATA\_FMT\_DECIMAL\_SEP** ::= . | , | space\_char See decimalSep in datafmt section.

**DATA\_FMT\_THOUSAND\_SEP** ::= . | , | space\_char See thousandSep in datafmt section.

**DATA\_FMT\_NEG\_ORDER** ::= See negativeOrder in datafmt section.

**DATA\_FMT\_POS\_ORDER** ::= See positiveOrder in datafmt section.

**DATA\_FMT\_CUR\_LOCALE** ::= currencyLocale:**DATA\_FMT\_LOCALE\_VAL**

**DATA\_FMT\_DATE\_FORMAT\_CUSTOM** ::= See dateFormat in datafmt section.

**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 datafmt section.

**DATA\_FMT\_EN\_STR** ::= 0 | 1 See englishStringOnly in datafmt section.

**DATA\_FMT\_TIME\_FORMAT\_CUSTOM** ::= See timeFormat in datafmt section.

**DATA\_FMT\_TIME\_FORMAT** ::= Short Time | Long Time | none |  
**DATA\_FMT\_TIME\_FORMAT\_CUSTOM**

**DATA\_FMT\_NOSECONDS** ::= 0 | 1 See noSeconds in datafmt section.

**DATA\_FMT\_CAT\_STRING**::= &quot;string&quot;; &quot;plainMutiline&quot;;

**DATA\_FMT\_CAT\_PERCENTAGE** ::= &quot;percentage&quot;;&quot;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)&quot;;

**DATA\_FMT\_CAT\_NUMBER** ::= &quot;number&quot;;&quot;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)&quot;;

**DATA\_FMT\_CAT\_DATETIME** ::= &quot;datetime&quot;;&quot;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?)&quot;;

**DATA\_FMT\_CAT\_DATE** ::= &quot;date&quot;;&quot;semicolon delimited list of  
(DATA\_FMT\_LOCALE?, DATA\_FMT\_DATE\_FORMAT, DATA\_FMT\_ALT\_CAL?,  
DATA\_FMT\_EN\_STR?)&quot;;

**DATA\_FMT\_CAT\_TIME** ::= &quot;time&quot;;&quot;semicolon delimited list of  
(DATA\_FMT\_LOCALE?, DATA\_FMT\_TIME\_FORMAT, DATA\_FMT\_NOSECONDS?)&quot;;

**DATA\_FMT\_CTRL\_DATE\_PICKER** ::= DATA\_FMT\_CAT\_DATE | DATA\_FMT\_CAT\_DATETIME

**DATA\_FMT\_CTRL\_EXPBOX** ::= DATA\_FMT\_CAT\_TIME | DATA\_FMT\_CAT\_DATE |  
DATA\_FMT\_CAT\_DATETIME | DATA\_FMT\_CAT\_NUMBER | 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 will execute actions, rules (1), or custom code when clicked.

Symbol	Description
BUTTON_RULES_AND_CUSTOM_CODE	The button executes rules (1) and custom code when

Symbol	Description
	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 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 ::= BUTTON\_RULES\_AND\_CUSTOM\_CODE |  
 BUTTON\_RULES\_AND\_CUSTOM\_CODE\_WITH\_CONDITIONAL\_FORMATTING |**

**BUTTON\_RULES\_AND\_CUSTOM\_CODE\_WITH\_DYNAMIC\_DISPLAY\_NAME\_AND\_CONDITIONAL\_FORMATTING |**

**BUTTON\_UPDATE\_FORM\_WITH\_CONDITIONAL\_FORMATTING |**

**BUTTON\_UPDATE\_FORM\_WITH\_DYNAMIC\_DISPLAY\_NAME\_AND\_CONDITIONAL\_FORMATTING | BUTTON\_ACTION |**

**BUTTON\_ACTION\_WITH\_CONDITIONAL\_FORMATTING |**

**BUTTON\_ACTION\_WITH\_DYNAMIC\_DISPLAY\_NAME\_AND\_CONDITIONAL\_FORMATTING**

**BUTTON\_ACTION\_TYPE** ::= 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\_BEGIN<sub>1</sub>**

<xsl:attribute name="value">

<xsl:value-of select="STRING\_XPATH\_EXPRESSION"/>

</xsl:attribute>

#### **CHECK\_FOR\_GETDOM\_END<sub>1</sub>**

</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\_BEGIN<sub>1</sub>**

<xsl:attribute name="value">

<xsl:value-of select="STRING\_XPATH\_EXPRESSION"/>

</xsl:attribute>

#### **CHECK\_FOR\_GETDOM\_END<sub>1</sub>**

</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\_BEGIN<sub>1</sub>**

<xsl:attribute name="value">

<xsl:value-of select="STRING\_XPATH\_EXPRESSION"/>

</xsl:attribute>

**CHECK\_FOR\_GETDOM\_END<sub>1</sub>**

</input>

Control-specific attributes used by the button control.

Attributes
<a href="#">xd:action</a>
<a href="#">xd:auxDom</a>
<a href="#">xd:CtrlId</a>
<a href="#">xd:postbackModel</a>
<a href="#">xd:xctname</a>

XSL function extensions used by the button control.

Extensions
<a href="#">xdEnvironment:IsBrowser</a>

2.4.1.6 Check Box Control

A check box control is a bi-state leaf control which has a value when it is checked, and a different value when it is not checked.

Symbol	Description
SIMPLE_CHECK_BOX	A control which 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 (commonly a check mark) contained in the white square.
CHECK_BOX_WITH_CONDITIONAL_FORMATTING	Similar to 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.

**CHECK\_BOX ::= SIMPLE\_CHECK\_BOX |  
CHECK\_BOX\_WITH\_CONDITIOANL\_FORMATTING  
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\_BEGIN<sub>1</sub>**

```
<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\_END<sub>1</sub>**

```
</input>
```

**(ANY\_STRING<sub>2</sub>)?**

**CHECK\_BOX\_ONVALUE::=** xd:onValue="(ISO\_DIGIT+)|(&quot;ANY\_STRING&quot;)"

**CHECK\_BOX\_OFFVALUE::=** xd:offValue="(ISO\_DIGIT+)|(&quot;ANY\_STRING&quot;)"

**CHECK\_BOX\_SINGLE\_VALUE::=** CHECK\_BOX\_OFFVALUE | CHECK\_BOX\_ONVALUE

**CHECK\_BOX\_BOTH\_VALUES::=** CHECK\_BOX\_OFFVALUE 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:

Attributes
<a href="#">xd:binding</a>
<a href="#">xd:boundProp</a>
<a href="#">xd:CtrlId</a>

Attributes
<a href="#">xd:offValue</a>
<a href="#">xd:onValue</a>
<a href="#">xd:postbackModel</a>
<a href="#">xd:Value</a>
<a href="#">xd:xctname</a>

### 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 (1).

**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="SiteUriXPath" VALUE="/Context/@siteUri"/>
```

```

<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.

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 will open the address book.
DisplayNameXPath	This parameter specifies the <b>LEAF_XPATH</b> containing the display names.
ObjectIdXPath	This parameter specifies the <b>LEAF_XPATH</b> containing the object IDs.
ObjectTypeXPath	This parameter specifies the <b>LEAF_XPATH</b> containing the object types.
SiteUrlXPath	This parameter specifies <b>LEAF_XPATH</b> in the secondary data source (2) containing the server URL, whose user information list (1) this control is querying. This parameter <b>MUST</b> be ignored by the form server.
SiteUrlDataSource	This parameter specifies the name of the secondary data source (2) that contains the server URL, whose user information list (1) this control is querying. This parameter <b>MUST</b> 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 <b>left-to-right</b> or <b>right-to-left</b> .

**FONT ::= ANY\_STRING** without a comma

The FONT\_ITALIC symbol specifies if the text will be shown italic or not. Following are the possible values and explanations.

Value	Description
0	not italic
1	Italic

**FONT\_ITALIC ::= 0 | 1**

**FONT\_SIZE** ::= all integers and all real numbers, ending in .5, between (inclusive) 1 and 2000

The FONT\_STRIKETHROUGH symbol specifies if the text will be shown with a strike through line or not. Following are the possible values and explanations.

Value	Description
0	no strike through
1	strike through

**FONT\_STRIKETHROUGH** ::= 0 | 1

The FONT\_UNDERLINE symbol specifies if the text shown will be underlined or not. Following are the possible values and explanations.

Value	Description
0	not underlined
1	Underlined

**FONT\_UNDERLINE** ::= 0 | 1

The FONT\_WEIGHT symbol specifies if the text shown will be bold or not. Following are the possible values and explanations.

Value	Description
400	not bold
700	Bold

**FONT\_WEIGHT** ::= 400 | 700

**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 an 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 (inclusive) 0 and 999

The CONTACT\_SELECTOR\_DIRECTION symbol specifies if the control is rendered left-to-right or right-to-left. Following are the possible values and explanations.

Value	Description
0	Use the orientation of the form (1).
1	Left to right.
2	Right to-left.

**CONTACT\_SELECTOR\_DIRECTION** ::= [0-2]

**CONTACT\_SELECTOR\_PERSON\_XPATH** ::= **RELATIVE\_RECREATING\_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\_RECREATING\_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\_RECREATING\_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.

Attributes
<a href="#">xd:binding</a>
<a href="#">xd:bindingProperty</a>
<a href="#">xd:bindingType</a>
<a href="#">xd:boundProp</a>
<a href="#">xd:CtrlId</a>
<a href="#">xd:disableEditing</a>
<a href="#">xd:xctname</a>

XSL function extensions 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.

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 which 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, although allows for conditional formatting. Conditional text formatting attributes such as bold, italics, and color can be applied to the displayed date. Conditional disabling will disable both the text box, and the date picker's calendar button. Likewise, conditionally hiding the control will hide both the display box as well as the calendar button.
DATE_PICKER_WITH_DATA_FORMATTING	Similar to SIMPLE_DATE_PICKER, although the data is formatted from its natural XML value.
DATE_PICKER_WITH_DATA_FORMATTING_AND_CONDITIONAL_FORMATTING	Similar to DATE_PICKER_WITH_CONDITIONAL_FORMATTING, although the data is formatted from its natural XML value.
DATE_PICKER_WITH_PLACEHOLDER_TEXT	Similar to SIMPLE_DATE_PICKER, with the addition of text which 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.

**DATE\_PICKER ::= SIMPLE\_DATE\_PICKER |**  
**DATE\_PICKER\_WITH\_CONDITIONAL\_FORMATTING |**  
**DATE\_PICKER\_WITH\_DATA\_FORMATTING |**  
**DATE\_PICKER\_WITH\_DATA\_FORMATTING\_AND\_CONDITIONAL\_FORMATTING |**  
**DATE\_PICKER\_WITH\_PLACEHOLDER\_TEXT |**  
**DATE\_PICKER\_WITH\_CONDITIONAL\_FORMATTING\_AND\_PLACEHOLDER\_TEXT |**  
**DATE\_PICKER\_WITH\_DATA\_FORMATTING\_AND\_PLACEHOLDER\_TEXT |**  
**DATE\_PICKER\_WITH\_DATA\_FORMATTING\_AND\_CONDITIONAL\_FORMATTING\_AND\_PLACEHOLDER\_TEXT**

**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\_BEGIN<sub>1</sub>**

```
<xsl:value-of select="LEAF_XPATH1" />
```

**CHECK\_FOR\_GETDOM\_END<sub>1</sub>**

```
</span>
```

```
<button class="DATE_PICKER_BUTTON_CLASS_NAME" (title="ANY_STRING1")?
xd:xctname="DTPicker_DTButton" xd:innerCtrl="_DTButton" (tabIndex="-1")?>
```

```

```

```
</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\_BEGIN<sub>1</sub>**

```
(<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: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")?>



</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")?>



</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_FORMATTI
NG)?</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")?>

</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")?>



</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_DText"
xd:binding="LEAF_XPATH1" (tabIndex="TAB_INDEX")? xd:innerCtrl="_DText"
(INPUT_SCOPE)?>

CHECK_FOR_GETDOM_BEGIN1

(<xsl:attribute name="style">

<xsl:choose>

(<xsl:when
test="BOOLEAN_XPATH_EXPRESSIONx">(DATE_PICKER_STYLE_CONDITIONAL_FORMATTI
NG)?</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\_END<sub>1</sub>

```
</span>
<button class="DATE_PICKER_BUTTON_CLASS_NAME" (title="ANY_STRING1")?
xd:xctname="DTPicker_DTButton" xd:innerCtrl="_DTButton" (tabIndex="-1")?>

</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:dateFormat="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\_BEGIN<sub>1</sub>

```
(<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(LEAF_XPATH,
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")?>



</button>

</div>

DATE_PICKER_WITH_DATA_FORMATTING_AND_CONDITIONAL_FORMATTING_AND_PLAC
EHOLDER_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_FORMATTI
NG)?</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\_END<sub>1</sub>**

```

</span>

<button class="DATE_PICKER_BUTTON_CLASS_NAME" (title="ANY_STRING1")?
xd:xctname="DTPicker_DTButton" xd:innerCtrl="_DTButton" (tabIndex="-1")?>



</button>

</div>

```

**DATE\_PICKER\_TEXT\_BOX\_CLASS\_NAME** ::= xdDTText | xdDTTextRTL

**DATE\_PICKER\_BUTTON\_CLASS\_NAME** ::= xdDTButton | 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:

Attributes
<a href="#">xd:allowNonMatching</a>
<a href="#">xd:binding</a>
<a href="#">xd:boundProp</a>
<a href="#">xd:CtrlId</a>
<a href="#">xd:dateFormat</a>
<a href="#">xd:innerCtrl</a>
<a href="#">xd:inputScope</a>
<a href="#">xd:num</a>
<a href="#">xd:postbackModel</a>
<a href="#">xd:xctrname</a>

#### 2.4.1.9 Drop-Down List Control

The dropdown list control enables the user to select a single value from a list (1) of options that can be specified manually by the form template designer, or is populated from a data source (2).

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 DROPDOWN_LIST_BOX, but allows conditional formatting (text formatting and disabling).
DROPDOWN_LIST_BOX_WITH_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 (2) of the form (1).
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_WITH_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

Symbol	Description
MATTING	M_DATA_SOURCE_AND_UNIQUE_DISPLAY_NAMES and DROPDOWN_LIST_BOX_WITH_CONDITIONAL_FORMATTING

**DROPDOWN\_LIST\_BOX ::=SIMPLE\_DROPDOWN\_LIST\_BOX |**

**DROPDOWN\_LIST\_BOX\_WITH\_CONDITIONAL\_FORMATTING |**

**DROPDOWN\_LIST\_BOX\_WITH\_VALUES\_FROM\_DATA\_SOURCE |**

**DROPDOWN\_LIST\_BOX\_WITH\_VALUES\_FROM\_DATA\_SOURCE\_AND\_CONDITIONAL\_FORMATTING |**

**DROPDOWN\_LIST\_BOX\_WITH\_VALUES\_FROM\_DATA\_SOURCE\_AND\_UNIQUE\_DISPLAY\_NAMES |**

**DROPDOWN\_LIST\_BOX\_WITH\_VALUES\_FROM\_DATA\_SOURCE\_AND\_UNIQUE\_DISPLAY\_NAMES\_AND\_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\_BEGIN<sub>1</sub>**

```
<xsl:attribute name="value">
```

```
<xsl:value-of select="LEAF_XPATH1" />
```

```
</xsl:attribute>
```

```
((<option (value="")?>
```

```
<xsl:if test="LEAF_XPATH1=&quot;&quot;">
```

```
<xsl:attribute name="selected">selected</xsl:attribute>
```

```
</xsl:if>
```

**ANY\_STRING<sub>2</sub>**

```
</option> ) |
```

```
(<option value="LEAF_VALUE1">
```

```
<xsl:if test="LEAF_XPATH1=LEAF_VALUE1">
```

```
<xsl:attribute name="selected">selected</xsl:attribute>
```

```
</xsl:if>
```

**ANY\_STRING<sub>x</sub>**

</option>))+

**CHECK\_FOR\_GETDOM\_END<sub>1</sub>**

</select>

**DROPDOWN\_LIST\_BOX\_WITH\_CONDITIONAL\_FORMATTING ::=**

<select class="xdComboBox xdBehavior\_Select" title="**ANY\_STRING<sub>1</sub>**"  
(accessKey="**SINGLE\_CHARACTER**")? (style="**DROPDOWN\_LIST\_BOX\_STYLES**")?  
size="**FONT\_SIZE**" xd:binding="**LEAF\_XPATH<sub>1</sub>**" xd:boundProp="value" xd:xctname="dropdown"  
(xd:postbackModel="**POSTBACKMODEL**")? (tabIndex="**TAB\_INDEX**")?  
xd:CtrlId="**CONTROL\_ID**")>

**CHECK\_FOR\_GETDOM\_BEGIN<sub>1</sub>**

(<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\_XPATH<sub>1</sub>**" />

</xsl:attribute>

((<option (value="")?>

<xsl:if test="**LEAF\_XPATH<sub>1</sub>**=&quot;&quot;">

<xsl:attribute name="selected">selected</xsl:attribute>

</xsl:if>

**ANY\_STRING<sub>2</sub>**

</option>) |

(<option value="**LEAF\_VALUE<sub>1</sub>**">

<xsl:if test="**LEAF\_XPATH<sub>1</sub>**=**LEAF\_VALUE<sub>1</sub>**">

<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\_FOR MATTING :: =**

```

<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\_BEGIN**<sub>1</sub>

```

(<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\_END<sub>1</sub>**

```
</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\_BEGIN<sub>1</sub>**

```
(<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_C
APTION)?</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 Drop Down List Box control:

Attributes
<a href="#">xd:binding</a>
<a href="#">xd:boundProp</a>
<a href="#">xd:CtrlId</a>

Attributes
<a href="#">xd:postbackModel</a>
<a href="#">xd:xctname</a>

XSL function extensions used by the Drop Down List Box control:

<b>xdXDocument:GetDOM</b>
---------------------------

### 2.4.1.10 Expression Box Control

The expression box control is a read-only control which displays the result of an XPath evaluation.

Symbol	Description
SIMPLE_EXPRESSION_BOX	An expression box is a control which displays the value of an XPath. It is constantly disabled.
EXPRESSIONBOX_WITH_CONDITIONAL_FORMATTING	Similar to EXPRESSION_BOX with text formatting and conditional formatting.
EXPRESSIONBOX_WITH_DATA_FORMATTING	Similar to 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

**EXPRESSION\_BOX ::= SIMPLE\_EXPRESSION\_BOX |**

**EXPRESSIONBOX\_WITH\_CONDITIONAL\_FORMATTING |**

**EXPRESSIONBOX\_WITH\_DATA\_FORMATTING |**

**EXPRESSIONBOX\_WITH\_DATA\_FORMATTING\_AND\_CONDITIONAL\_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\_BEGIN<sub>1</sub>**

```
<xsl:value-of select="EXPRESSION_BOX_XPATH1" />
```

**CHECK\_FOR\_GETDOM\_END<sub>1</sub>**

```
</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\_BEGIN<sub>1</sub>**

```
<xsl:attribute name="style">EXPRESSION_BOX_STYLE
```

```
<xsl:choose>
```

```
(<xsl:when
test="BOOLEAN_XPATH_EXPRESSIONx">LEAF_CONTROL_CONDITIONAL_FORMATTING_C
APTION</xsl:when>)+
```

```
</xsl:choose>
```

```
</xsl:attribute>
```

```
<xsl:value-of select="EXPRESSION_BOX_XPATH1" />
```

**CHECK\_FOR\_GETDOM\_END<sub>1</sub>**

```
</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\_BEGIN<sub>1</sub>**

```
(<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\_END<sub>1</sub>**

```
</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\_BEGIN<sub>1</sub>**

```
<xsl:attribute name="style">EXPRESSION_BOX_STYLE
```

```
<xsl:choose>
```

```
(<xsl:when
test="BOOLEAN_XPATH_EXPRESSIONx">LEAF_CONTROL_CONDITIONAL_FORMATTING_CA
PTION</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\_END<sub>1</sub>**

```
</span>
```

**EXPRESSION\_BOX\_XPATH ::= LEAF\_XPATH | 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:

Attributes
<a href="#">xd:binding</a>
<a href="#">xd:CtrlId</a>
<a href="#">xd:datfmt</a>
<a href="#">xd:disableEditing</a>
<a href="#">xd:num</a>
<a href="#">xd:xctname</a>

#### 2.4.1.11 File Attachment Control

The file attachment control enables users to attach a file to a form (1).

**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.

Attributes
<a href="#">xd:binding</a>
<a href="#">xd:boundProp</a>
<a href="#">xd:CtrlId</a>
<a href="#">xd:disableEditing</a>
<a href="#">xd:xctname</a>

XSL function extensions used by the file attachment control.

**xdImage:getImageUrl**

#### 2.4.1.12 Hyperlink Control

The hyperlink control allows the user to create a hyperlink which will navigate the default browser to a specified URL (Uniform Resource Locator).

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 (1).
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.

**HYPERLINK ::= SIMPLE\_HYPERLINK |**

**HYPERLINK\_WITH\_DYNAMIC\_LINK |**

**HYPERLINK\_WITH\_DYNAMIC\_TEXT |**

**HYPERLINK\_WITH\_DYNAMIC\_LINK\_AND\_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\_BEGIN<sub>1</sub>**

```
<xsl:attribute name="href">
```

```
<xsl:value-of select="LEAF_XPATH"/>
```

```
</xsl:attribute>
```

**ANCHOR\_TEXT**

**CHECK\_FOR\_GETDOM\_END<sub>1</sub>**

</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\_BEGIN<sub>1</sub>**

<xsl:value-of select="LEAF\_XPATH"/>

**CHECK\_FOR\_GETDOM\_END<sub>1</sub>**

</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\_BEGIN<sub>1</sub>**

<xsl:attribute name="href">

<xsl:value-of select="LEAF\_XPATH<sub>1</sub>"/>

</xsl:attribute>

<xsl:value-of select="LEAF\_XPATH<sub>2</sub>"/>

**CHECK\_FOR\_GETDOM\_END<sub>1</sub>**

</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:

Attributes
<a href="#">xd:CtrlId</a>
<a href="#">xd:disableEditing</a>

Attributes
<a href="#">xd:xctname</a>

#### 2.4.1.13 List Box Control

The list box control enables the user to select a single value from a list (1) of options that can be specified manually by the form template designer, or is populated from a data source (2).

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 ::= LIST\_BOX\_WITH\_MANUAL\_ENTRIES |**

**LIST\_BOX\_WITH\_LOOKUP\_ENTRIES |**

**LIST\_BOX\_WITH\_UNIQUE\_LOOKUP\_ENTRIES**

**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\_XPATH<sub>1</sub>" xd:boundProp="value" (style="LIST\_BOX\_STYLE")?  
(xd:postbackModel="POSTBACKMODEL")? (accessKey="SINGLE\_CHARACTER")?>

#### CHECK\_FOR\_GETDOM\_BEGIN<sub>1</sub>

#### LIST\_BOX\_CONDITIONAL\_FORMATTING?

```
<xsl:attribute name="value">
<xsl:value-of select="LEAF_XPATH1"/>
</xsl:attribute>
((<option>
<xsl:if test="LEAF_XPATH1=&quot;&quot;;">
<xsl:attribute name="selected">selected</xsl:attribute>
</xsl:if>OPTION_DISPLAY_VALUE?</option>)|
(<option value="OPTION_VALUE1">
<xsl:if test="LEAF_XPATH1=&quot;;OPTION_VALUE1&quot;;">
<xsl:attribute name="selected">selected</xsl:attribute>
</xsl:if>OPTION_DISPLAY_VALUE?</option>))+
```

#### CHECK\_FOR\_GETDOM\_END<sub>1</sub>

</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_XPAT
H1=$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_XPAT
H1=$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:

Attributes
<a href="#">xd:binding</a>
<a href="#">xd:boundProp</a>
<a href="#">xd:CtrlId</a>
<a href="#">xd:postbackModel</a>
<a href="#">xd:xctname</a>

XSL function extensions used by the list box control:

### 2.4.1.14 Option Button Control

An option button is a bi-state control which has a value when selected, and no value when not selected. Option button controls are meant to be used in groups (1), with selection amongst the option buttons in the group (1) being mutually exclusive.

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	Same as SIMPLE_OPTION_BUTTON, but allows the button to be conditionally disabled.

**OPTION\_BUTTON ::= SIMPLE\_OPTION\_BUTTON |**

**OPTION\_BUTTON\_WITH\_CONDITIONAL\_FORMATTING**

**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\_BEGIN<sub>1</sub>**

```
<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\_END<sub>1</sub>**

```
</input>
```

**ANY\_STRING<sub>3</sub>**

**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\_BEGIN<sub>1</sub>**

```
<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\_STRING<sub>3</sub>**

**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:

Attributes
<a href="#">xd:binding</a>
<a href="#">xd:boundProp</a>
<a href="#">xd:CtrlId</a>
<a href="#">xd:onValue</a>
<a href="#">xd:postbackModel</a>
<a href="#">xd:Value</a>
<a href="#">xd:xctname</a>

**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 (1).

Symbol	Description
REPEATING_SECTION_CALL	First part of the repeating section control that indicates where the repeating sections is rendered.
REPEATING_SECTION_BODY_WITHOUT_CONDITIONAL_HIDING	Second part of the repeating section that defines the properties of the repeating section and its content.
REPEATING_SECTION_BODY_WITHOUT_CONDITIONAL_HIDING	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.

**REPEATING\_SECTION** MUST consist of a **REPEATING\_SECTION\_CALL** at the point in the XSL where the control appears (body of 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 |**

**(CHECK\_FOR\_GETDOM\_BEGIN<sub>1</sub>**

```
<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\_END<sub>1</sub>)**

**REPEATING\_SECTION\_BODY ::=**  
**REPEATING\_SECTION\_BODY\_WITHOUT\_CONDITIONAL\_HIDING |**

**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.

Attributes
<a href="#">xd:action</a>
<a href="#">xd:CtrlId</a>
<a href="#">xd:postbackModel</a>
<a href="#">xd:xctname</a>
<a href="#">xd:xmlToEdit</a>

### 2.4.1.16 Repeating Table Control

A repeating table control acts as container for other controls, and can appear multiple times in an instance of a form (1), it has a tabular format.

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	Same as SIMPLE_REPEATING_TABLE, with the ability to conditionally change background color and disable adding or removing rows.
TABLE_ROW_WITH_CONDITIONAL_FORMATTING	Same as TABLE_ROW, although allows for changing background color.

**REPEATING\_TABLE ::= SIMPLE\_REPEATING\_TABLE |**

**REPEATING\_TABLE\_WITH\_CONDITIONAL\_FORMATTING**

**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\_BEGIN**<sub>1</sub>

<xsl:for-each select="**GROUP\_XPATH**">

**REPEATING\_TABLE\_ROWS\_WITH\_CONDITIONALVISIBILITY |**

**TABLE\_ROW\_WITH\_CONDITIONAL\_FORMATTING\***

</xsl:for-each>

**CHECK\_FOR\_GETDOM\_END**<sub>1</sub>

</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\_EXPRESSION<sub>x</sub>">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.

Attributes
<a href="#">Xd:action</a>
<a href="#">xd:CtrlId</a>
<a href="#">xd:postBackModel</a>
<a href="#">xd:xctname</a>
<a href="#">xd:xmlToEdit</a>

### 2.4.1.17 Rich Text Box Control

The rich text box control allows the user to enter rich text (formatted text, tables, hyperlinks, images, and so on.) in the form (1).

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 ::= RICH\_TEXT\_BOX\_PLAIN |  
RICH\_TEXT\_BOX\_WITH\_PLACEHOLDER\_TEXT**

**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\_BEGIN<sub>1</sub>**

**RICH\_TEXT\_BOX\_CONDITIONAL\_FORMATTING?**

```
<xsl:copy-of select="LEAF_XPATH/node()"/>
```

**CHECK\_FOR\_GETDOM\_END<sub>1</sub>**

```
</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\_BEGIN<sub>1</sub>**

**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\_END<sub>1</sub>**

```
</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_S
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.

Attributes
<a href="#">xd:allowNonMatching</a>
<a href="#">xd:binding</a>
<a href="#">xd:CtrlId</a>
<a href="#">xd:disableEditing</a>
<a href="#">xd:ghosted</a>
<a href="#">xd:inputScopeId</a>
<a href="#">xd:postbackModel</a>
<a href="#">xd:xctname</a>

### 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.

Symbol	Description
SIMPLE_SECTION_CALL	First part of the section that indicates where the section control is rendered.
OPTIONAL_SECTION_CALL	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	Second part of the section or optional section that defines the properties of the section and its content.
SIMPLE_SECTION_BODY_WITH_DIGITAL_SIGNATURE	Second part of the section that defines the properties of the section and its content. It also contains a digital signature component that will enable the user to sign it.
SIMPLE_SECTION_BODY_WITH_CONDITIONAL_HIDING	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	Second part of the section or optional section that defines the properties of the section and its content. This part is used when the

Symbol	Description
	section is conditionally hidden. It also contains a digital signature component that will enable the user to sign it.

**SECTION** MUST consist of a **SIMPLE\_SECTION\_CALL** at the point in the XSL where the control appears (body of 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.

**OPTIONAL\_SECTION** MUST consist of a **OPTIONAL\_SECTION\_CALL** at the point in the XSL where the control appears (body of 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\_BEGIN<sub>1</sub>**

```
<xsl:apply-templates select="(GROUP_XPATH/)?RELATIVE_GROUP_XPATH"
mode="TEMPLATE_MODE_ID1"/>
```

**CHECK\_FOR\_GETDOM\_END<sub>1</sub>**

**OPTIONAL\_SECTION\_CALL ::= SIMPLE\_SECTION\_CALL |**

**(CHECK\_FOR\_GETDOM\_BEGIN<sub>1</sub>**

```
<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\_END<sub>1</sub>)**

**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** |  
**SIMPLE\_SECTION\_BODY\_WITH\_DIGITAL\_SIGNATURE** |  
**SIMPLE\_SECTION\_BODY\_WITH\_CONDITIONAL\_HIDING** |  
**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/xd
SignatureProperties:ValidSignedImage">

<td style="display: none;"></td>

<td> </td>

</xsl:when>

<xsl:otherwise>

<td></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/xd
SignatureProperties:InvalidSignedImage">
<td style="display: none;"></td>
<td></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_STRING"
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></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></td>

<td>ANY_STRING</td>

</tr>

</tbody>

</table>

</button>

</xsl:if>

</xsl:if>

</div>
```

Control-specific attributes used by the section control.

Attributes
xd:disableEditing (section <a href="#">2.4.2.12</a> )
xd:postbackModel (section <a href="#">2.4.2.29</a> )
xd:SignatureBlock (section <a href="#">2.4.2.31</a> )
xd:SignedSectionDisplaySignatures (section <a href="#">2.4.2.32</a> )
xd:SignedSectionName (section <a href="#">2.4.2.33</a> )
xd:xctname (section <a href="#">2.4.2.35</a> )
xd:xmlToEdit (section <a href="#">2.4.2.36</a> )

XSL function extensions used by the section control.

Functions
xdImage:getImageUrl (section <a href="#">2.4.3.5</a> )
xdXDocument:GetNamedNodeProperty (section <a href="#">2.4.3.9.3</a> )

### 2.4.1.19 Table Control

A table control is used to ensure that elements of the form (1) are positioned as per the requirements of the form designer.

Symbol	Description
TABLE	Table maps to a standard html layout table as specified by <a href="#">[HTML]</a> section 11.2.1. Each cell can contain arbitrary html or more controls.
TABLE_COLUMN	Maps to the html col element as specified by <a href="#">[HTML]</a> section 11.2.4.
TABLE_ROW	Maps to the html tr element as specified by <a href="#">[HTML]</a> section 11.2.5.
TABLE_CELL	Maps to the html td element as specified by <a href="#">[HTML]</a> 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?)**

Control-specific attribute used by the Table control.

**xd:layoutText**

#### 2.4.1.20 Text Box Control

The text box control allows the user to enter simple text in the form (1).

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

Symbol	Description
	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_PLACEHOLDER_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.

Symbol	Description
	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\_CONTROL ::= SIMPLE\_TEXT\_BOX |**  
**TEXT\_BOX\_WITH\_CONDITIONAL\_FORMATTING | SIMPLE\_TEXT\_BOX\_MULTI\_LINE |**  
**TEXT\_BOX\_MULTI\_LINE\_WITH\_CONDITIONAL\_FORMATTING |**  
**SIMPLE\_TEXT\_BOX\_WITH\_DATA\_FORMATTING |**  
**TEXT\_BOX\_WITH\_CONDITIONAL\_FORMATTING\_AND\_DATA\_FORMATTING |**  
**SIMPLE\_TEXT\_BOX\_WITH\_PLACEHOLDER\_TEXT |**  
**SIMPLE\_TEXT\_BOX\_WITH\_PLACEHOLDER\_TEXT\_AND\_DATA\_FORMATTING |**  
**TEXT\_BOX\_MULTI\_LINE\_PLACEHOLDER\_TEXT |**  
**TEXT\_BOX\_MULTI\_LINE\_CONDITIONAL\_FORMATTING\_AND\_PLACEHOLDER\_TEXT |**  
**TEXT\_BOX\_WITH\_CONDITIONAL\_FORMATTING\_AND\_PLACEHOLDER\_TEXT |**  
**TEXT\_BOX\_WITH\_CONDITIONAL\_FORMATTING\_AND\_PLACEHOLDER\_TEXT\_AND\_DATA\_FORMATTING**

**TEXT\_BOX\_EDITING ::=** contentEditable="true" | xd:disableEditing="yes" |  
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 | xdTextBoxRTL

**SIMPLE\_TEXT\_BOX** ::=

<span class="TEXT\_BOX\_BASE\_CLASS\_NAME" hideFocus="1" title="ANY\_STRING"  
xd:binding="LEAF\_XPATH<sub>1</sub>"

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\_BEGIN<sub>1</sub>

<xsl:value-of select="LEAF\_XPATH<sub>1</sub>" TEXT\_BOX\_OUTPUT\_ESC? />

CHECK\_FOR\_GETDOM\_END<sub>1</sub>

</span>

**TEXT\_BOX\_WITH\_CONDITIONAL\_FORMATTING** ::=

<span class="TEXT\_BOX\_BASE\_CLASS\_NAME" hideFocus="1" title="ANY\_STRING"  
xd:binding="LEAF\_XPATH<sub>1</sub>" (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\_BEGIN<sub>1</sub>

TEXT\_BOX\_CONDITIONAL\_FORMATTING

<xsl:value-of select="LEAF\_XPATH<sub>1</sub>" TEXT\_BOX\_OUTPUT\_ESC? />

CHECK\_FOR\_GETDOM\_END<sub>1</sub>

</span>

**TEXT\_BOX\_CONDITIONAL\_FORMATTING** ::=

**TEXT\_BOX\_CONDITIONAL\_FORMATTING\_ATT**  
(**TEXT\_BOX\_CONDITIONAL\_FORMATTING\_CHOOSE**)? |

**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\_BEGIN<sub>1</sub>**

```
<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\_BEGIN<sub>1</sub>**

**TEXT\_BOX\_CONDITIONAL\_FORMATTING**

**DATA\_FMT\_XSL**

**CHECK\_FOR\_GETDOM\_END<sub>1</sub>**

</span>

**PLACEHOLDER\_TEXT\_XSL\_BASE ::=**

<xsl:when test="not(string(**LEAF\_XPATH<sub>1</sub>**))">

<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\_XPATH<sub>1</sub>**" **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\_XPATH<sub>1</sub>**" (**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\_BEGIN<sub>1</sub>**

**PLACEHOLDER\_TEXT\_XSL**

**CHECK\_FOR\_GETDOM\_END<sub>1</sub>**

</span>

**SIMPLE\_TEXT\_BOX\_WITH\_PLACEHOLDER\_TEXT\_AND\_DATA\_FORMATTING ::=**

<span class="**TEXT\_BOX\_BASE\_CLASS\_NAME**" xdBehavior\_GTFormatting" hideFocus="1"

title="**ANY\_STRING**" xd:binding="**LEAF\_XPATH<sub>1</sub>**" 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.

Attributes
<a href="#">xd:allownonmatching</a>

Attributes
<a href="#">xd:autoAdvance</a>
<a href="#">xd:binding</a>
<a href="#">xd:boundProp</a>
<a href="#">xd:CtrlId</a>
<a href="#">xd:datafmt</a>
<a href="#">xd:disableEditing</a>
<a href="#">xd:ghosted</a>
<a href="#">xd:inputScopeId</a>
<a href="#">xd:num</a>
<a href="#">xd:postbackModel</a>
<a href="#">xd:xctname</a>

#### 2.4.1.21 Ignored Controls

XSL files SHOULD contain valid controls but MAY contain controls which are not recognized by the form server and therefore ignored. The XSL fragment which 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.

Following is a list (1) 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 which 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 a **XmlToEdit** element (section [2.2.124](#)) with a recursive item XPATH.

A **Repeating Section** (section [2.4.1.15](#)) MUST NOT contain the attribute **linkedToMaster**.

A form definition (.xsf) file which 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 which 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 the [View Representation](#) (section View Representation).

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 <a href="#">2.4.2.1</a> )	Specifies the action executed when a control is clicked.
allownonmatching (section <a href="#">2.4.2.2</a> )	Client-only.
autoAdvance (section <a href="#">2.4.2.3</a> )	A Boolean value that specifies whether to move focus to the next control in the form (1) when the text limit is reached on a text box control.
auxDom (section <a href="#">2.4.2.4</a> )	Specifies the <b>name</b> (section <a href="#">2.2.36</a> ) of the secondary data connection associated with the "refresh" <b>action</b> (section <a href="#">2.4.2.1</a> ) of a button control.
backgroundPicture (section <a href="#">2.4.2.5</a> )	Client-only.
binding (section <a href="#">2.4.2.6</a> )	Specifies the XML field from which the control reads and writes data.
bindingProperty (section <a href="#">2.4.2.7</a> )	Specifies the name of the custom control property used by the custom control to read and write data from and to the XML field

Attribute	Description
	associated with the control.
bindingType (section <a href="#">2.4.2.8</a> )	Specifies the format of the data which the custom control reads and writes.
boundProp (section <a href="#">2.4.2.9</a> )	Specifies the name of the XSLT attribute which contains the XML field from which the control reads and writes data.
CtrlId (section <a href="#">2.4.2.10</a> )	Specifies the unique identifier of a control in the form (1).
<a href="#">datafmt</a> (section <a href="#">datafmt</a> )	Specifies the data formatting the control uses to display the data in the associated XML field in the form (1).
disableEditing (section <a href="#">2.4.2.12</a> )	A Boolean value that specifies whether a control is specified as read-only.
enabledProperty (section <a href="#">2.4.2.13</a> )	Specifies the name of the custom control property called to enable or disable the control.
enabledValue (section <a href="#">2.4.2.14</a> )	A Boolean value to use to enable or disable the custom control.
ghosted (section <a href="#">2.4.2.15</a> )	Client-only.
ictID (section <a href="#">2.4.2.16</a> )	Client-only.
ictVersion (section <a href="#">2.4.2.17</a> )	Client-only.
inline (section <a href="#">2.4.2.18</a> )	Client-only.
innerCtrl (section <a href="#">2.4.2.19</a> )	Specifies a type of component inside the date picker control.
inputscope (section <a href="#">2.4.2.20</a> )	Client-only.
inputScopeId (section <a href="#">2.4.2.21</a> )	Client-only.
layoutText (section <a href="#">2.4.2.22</a> )	Client-only.
linkedToMaster (section <a href="#">2.4.2.23</a> )	Client-only.
masterID (section <a href="#">2.4.2.24</a> )	Client-only.
masterName (section <a href="#">2.4.2.25</a> )	Client-only.
num (section <a href="#">2.4.2.26</a> )	Indicates the "xd:num" XSLT attribute contains the XML field from which the control reads and writes data.
offValue (section <a href="#">2.4.2.27</a> )	Specifies the XML field value for a check box control when that check box control is unchecked.
onValue (section <a href="#">2.4.2.28</a> )	Specifies the XML field value for a control when that control is selected.
postbackModel (section <a href="#">2.4.2.29</a> )	Specifies whether to trigger a postback when the XML field value displayed in the control is changed.
ref (section <a href="#">2.4.2.30</a> )	Client-only.
SignatureBlock (section <a href="#">2.4.2.31</a> )	Specifies the name for the signed data block (section <a href="#">2.2.126</a> ) for a control that allows XML digital signatures (1).

Attribute	Description
SignedSectionDisplaySignatures (section <a href="#">2.4.2.32</a> )	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 <a href="#">2.4.2.33</a> )	Specifies the name for the signed data block (section <a href="#">2.2.126</a> ) for a control that allows XML digital signatures (1).
value (section <a href="#">2.4.2.34</a> )	Indicates the "xd:value" XSLT attribute contains the XML field from which the control reads and writes data.
xctname (section <a href="#">2.4.2.35</a> )	Specifies the type of the control.
xmlToEdit (section <a href="#">2.4.2.36</a> )	Specifies the additional editing properties of the control

### 2.4.2.1 action

#### Applicable controls

Button (section [2.4.1.5](#)), repeating section (section [2.4.1.15](#)) and repeating table (section [2.4.1.16](#)) controls MAY contain this attribute in their respective XSLT representations.

All other controls MUST NOT contain this attribute in their respective XSLT representations.

#### Details

For button controls, the value of this attribute MUST be set to 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 (1).

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 (1).

This value MUST be used only within forms (1) in which the main data source is a data connection (1).

**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 (1) has at least one secondary data connection that can query.

**submit:** Calls the action to submit the form (1).

**updateForm:** Calls a postback to the form server to update the form (1).

For repeating section and repeating table controls, the value of this attribute MUST be set to one of the following:

**xCollection::insert:** 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

This 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

##### Applicable Controls

[Text box](#) controls MAY contain this attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their respective XSLT representations.

##### Details

The value of this attribute MUST be set to the following:

**yes:** Move 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

##### Applicable Controls

[Button](#) controls MAY contain this attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their respective XSLT representations.

##### Details

The value of this attribute MUST range from 1 through 255 characters and MUST be equal to the [name](#) of a secondary data connection that exists in the form (1).

If this attribute is unspecified, all of the secondary data sources that can query MUST be refreshed as part of the "refresh" [action](#).

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 backgroundImage

This 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

#### Applicable Controls

The following controls MUST contain this 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 following controls MAY contain this attribute in their XSLT representation:

- Expression box (section [2.4.1.10](#))

All other controls MUST NOT contain this attribute in their respective XSLT representations.

#### Details

The value of this attribute for the following controls MUST be set to a LEAF\_XPATH as specified in **View Syntax** (section [2.4.1.1](#)):

- Check box
- Date picker
- Drop-down list box
- Expression box
- File attachment

- List box
- Option button
- Rich text box
- Text box

For the **contact selector** (Section [3.4.1.3](#)) control the value of this attribute MUST be set to a GROUP\_XPATH as specified in **View Syntax** (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

#### Applicable Controls

Custom controls MAY contain this attribute in their XSLT representation.

[Contact selector](#) controls MUST contain this attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their respective XSLT representations.

#### Details

The value of this attribute MUST be equal to 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

#### Applicable Controls

Custom controls MAY contain this attribute in their XSLT representation.

[Contact selector](#) controls MUST contain this attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their respective XSLT representations.

#### Details

Custom controls use the binding type attribute to determine how to read and write data in and out of the control.

The value of this attribute MUST be set to one of the following:

- **text:** Specifies a text string data format.
- **xmlnode:** Specifies an XML node data format.

- **xmltext:** Specifies 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

### Applicable Controls

The following controls **MUST** contain this attribute in their XSLT representation:

- [Check box](#)
- Custom controls
- [Drop-down list box](#)
- [File attachment](#)
- [List box](#)
- [Option button](#)

The following controls **MAY** contain this attribute in their XSLT representation:

- [Date picker](#)
- [Text box](#)

All other controls **MUST NOT** contain this attribute in their respective XSLT representations.

### Details

The value of this attribute **MUST** be set as specified in the following table:

<b>Value</b>	<b>Description</b>	<b>Controls</b>
src	This value is associated with a client-only feature and <b>MUST</b> be ignored by the form server.	
value	Indicates the <b>value</b> XSLT attribute contains the XML field from which the control reads and writes data.	Drop-down list box List box
xdInkData	This value is associated with a client-only feature and <b>MUST</b> be ignored by the form server.	
xd:inline	Indicates 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 the <b>xd:num</b> XSLT attribute contains the XML field from which the control reads and writes data.	Date picker Text box
xd:value	Indicates the <b>xd:value</b> 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

##### Applicable Controls

The following controls **MUST** contain this 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 box (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.

##### Details

The value of this 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 (1).

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

##### Applicable Controls

The following controls MAY contain this attribute in their XSLT representation:

- [Date picker](#)
- [Expression box](#)
- [Text box](#)

All other controls MUST NOT contain this attribute in their respective XSLT representations.

##### Details

The value of this attribute MUST follow this [structure](#):

"<FormatCategory>"<FormatSpecification>"

##### *FormatCategory*

Specifies the type of formatting applied when the control displays the XML field's data.

The *FormatCategory* string MUST be set to 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*

Specifies 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

FormatCategory	Supported FormatSpecification
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 (1) of one or more specification items. Each specification 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 format specification item names MUST be set to the following and adhere to the specified value requirements:

- **currencyLocale:** Specifies the locale identifier as specified in [\[MS-LCID\]](#).
  - This specification item is used to map the currency symbol string determined by the currency locale.
- **dateFormat:** Specifies the date format pattern to use when displaying date and time values.
  - The value of this specification MUST be set to one of the following:
    - **Short Date:** Specifies a short date format as specified in [\[MC-NLSIP\]](#).
    - **Long Date:** Specifies a long date format as specified in [\[MC-NLSIP\]](#).
    - **Year Month:** Specifies a year month format as specified in [\[MC-NLSIP\]](#).
    - **none:** No date format pattern applied.
    - A date format pattern as specified in [\[ISO-8601\]](#).
- **decimalSep:** Specifies the string to display as the decimal separator in numeric values.
  - The value of this specification item MUST be set to one of the following:
    - '.' (Period)
    - ',' (Comma)
    - ' ' (Space)
    - ' ' (Non-breaking space)
    - '"' (Single Quote)

- '٫' (Arabic Comma)
- Empty (No separator)
- If this specification 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 locale identifier, as specified in [MS-LCID], when displaying date and date time values.
  - The value of this specification item MUST be set to one of the following:
    - **0** (Use locale specified by the locale identifier, as specified in [MS-LCID])
    - **1** (Use English as the locale identifier, as specified in [MS-LCID])
  - If this specification item is unspecified, the behavior MUST be the same as a specification item value of "0".
- **grouping:** Specifies the digit grouping pattern for digits to the left of the decimal.
  - The value of this specification item MUST be set to one of the following:
    - Range from **0 (no grouping) through 9**
    - **32**
  - If this specification item is unspecified, the grouping is determined based on the form server settings.
- **locale:** Specifies the locale identifier, as specified in [MS-LCID].
  - If this specification item is unspecified, the locale is determined based on the form server settings.
- **negativeOrder:** Specifies the format pattern for negative numeric values.
  - If this specification item's value is set to "-1", the negative order is determined by the default pattern associated with the form server's locale identifier, as specified in [MS-LCID].
  - If this specification 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 of this specification item MUST be set to one of the following:
    - **0** (Display seconds for time formatting)
    - **1** (Do not display seconds for time formatting)
  - If this specification item is unspecified, the behavior MUST be the same as a specification item value of "0".
- **numDigits:** Specifies the number of fractional digits to display after the decimal separator.
  - The value of this specification item MUST be set to one of the following:
    - Range from **0 through 9**

- **Auto** (Specifies the general numeric format string as implemented by the form server)
  - If this specification item is unspecified, the number of digits is determined based on the form server settings.
- **plainMultiline:** Specifies the ability for a text box control to display data across multiple lines.
  - The value of this specification item **MUST** be set to an empty string (1).
- **positiveOrder:** Specifies the format pattern for positive currency values.
  - If this specification item is set to "-1", the positive order is determined by the default pattern associated with the form server's locale identifier, as specified in [MS-LCID].
  - If this specification item is unspecified, the positive order is determined based on the form server settings.
- **thousandSep:** Specifies the string that separates groups (1) of digits to the left of the decimal in numerical values.
  - The value of this specification item **MUST** be set to one of the following:
    - '.' (Period)
    - ',' (Comma)
    - ' ' (Space)
    - ' ' (Non-breaking space)
    - '"' (Single Quote)
    - '، ' (Arabic Comma)
    - Empty (No separator)
  - If this specification item is unspecified, the thousand separator is determined based on the form server settings.
- **timeFormat:** Specifies the time format pattern to use when displaying time values.
  - The value of this specification **MUST** be set to one of the following:
    - **Short Time:** Specifies a short time format [\[MC-NLSIP\]](#).
    - **Long Time:** Specifies a long time format [\[MC-NLSIP\]](#).
    - **none:** No time format pattern applied.
    - A time format pattern [\[ISO-8601\]](#).
  - If this specification 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 of this specification item **MUST** be set to one of the following:

- **0** (Use calendar type associated with the locale identifier, as specified in [MS-LCID])
- **1** (Use alternate calendar type for the locale identifier, as specified in [MS-LCID])
- If this specification item is unspecified, the behavior **MUST** be the same as a specification item value of "0".

## Invalid Constructs

### 1. *datetime category*

If the format category is set to "datetime", at least one of the two specification items "timeFormat" and "dateFormat" **MUST** be set to the value "none".

### 2. *Unsupported combinations of locale and date and time format*

The following combinations of locale and date and time format specifications **MUST NOT** be present:

Locale	Date and time format
1028	"M'\u6708'd'\u65E5'"
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"

Locale	Date and time format
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'\'"
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

### Applicable Controls

The following controls MAY contain this 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 (1).

All other controls MUST NOT contain this attribute in their respective XSLT representations.

#### Details

The value of this attribute MUST be set to the following:

- **yes:** Disable 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

#### Applicable Controls

Custom controls MAY contain this attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their respective XSLT representations.

#### Details

The value of this 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

#### Applicable Controls

Custom controls MAY contain this attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their respective XSLT representations.

#### Details

The value of this attribute MUST be set to one of the following:

- **true:** Use "true" to enable the custom control using the "[enabledProperty](#)".
- **false:** Use "false" to 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

This 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

This 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

This 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

This 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

##### Applicable Controls

[Date picker](#) controls MUST contain this attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their respective XSLT representations.

##### Details

This attribute MUST be set to one of the following values, which specify the respective component types:

- **\_DTButton:** Specifies a date picker button.
- **\_DTText:** Specifies 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

This 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

This 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

This 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

This 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

This 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

This 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

##### Applicable Controls

The following controls MAY contain this attribute in their XSLT representation:

- [Date picker](#)
- [Expression box](#)
- [Text box](#)

All other controls MUST NOT contain this attribute in their respective XSLT representations.

##### Details

The value of this attribute MUST be set as specified in the "[boundProp](#)" attribute.

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

##### Applicable Controls

[Check box](#) control

- 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 respective XSLT representations.

##### Details

The value of this attribute MUST be set to a value which is valid for the XML field's data type.

The value of this attribute MUST be set to 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

##### Applicable Controls

[Check box](#) control

- 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 **MAY** contain this attribute in their XSLT representation.

All other controls **MUST NOT** contain this attribute in their respective XSLT representations.

##### Details

The value of this attribute **MUST** be set to a value which is valid for the XML field's data type.

For the check box control, the value of this attribute **MUST** be set to 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

##### Applicable Controls

The following controls **MAY** contain this 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 box (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 respective XSLT representations.

#### Details

The value of this attribute MUST be set to 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

This 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

#### Applicable Controls

**Section and optional section** controls (section [2.4.1.18](#)) MAY contain this attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their respective XSLT representations.

#### Details

The value of this attribute MUST be equal to the **signed data block name** (section [2.2.126](#)) that exists in the form.

The value of this attribute MUST be equal to the value specified for the "SignedSectionName" (section [2.4.2.33](#)) attribute.

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

##### Applicable Controls

**Section and optional section** controls (section [2.4.1.18](#)) MAY contain this attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their respective XSLT representations.

##### Details

The value of this attribute MUST be set to the following:

- **true:** 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

##### Applicable Controls

**Section and optional section** controls (section [2.4.1.18](#)) MAY contain this attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their respective XSLT representations.

##### Details

The value of this attribute MUST be equal to the **signed data block name** (section [2.2.126](#)) that exists in the form (1).

The value of this attribute MUST be equal to the value specified for the "SignatureBlock" (section [2.4.2.31](#)) attribute.

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

### Applicable Controls

The following controls **MUST** contain this 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 respective XSLT representations.

### Details

The value of this attribute **MUST** be set as specified in the "boundProp" (section [2.4.2.9](#)) attribute.

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

### Applicable Controls

All controls **MUST** contain this attribute in their XSLT representation.

### Details

The value of this attribute **MUST** be set to one of the following for **built-in control**:

Control	"xctname" value
Button (section <a href="#">2.4.1.5</a> )	Button
Check box (section <a href="#">2.4.1.6</a> )	CheckBox
Date picker (section <a href="#">2.4.1.8</a> )	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 <a href="#">2.4.1.9</a> )	DropDown
Expression box (section <a href="#">2.4.1.10</a> )	ExpressionBox
File attachment (section <a href="#">2.4.1.11</a> )	FileAttachment
Hyperlink (section <a href="#">2.4.1.12</a> )	Hyperlink
List box (section <a href="#">2.4.1.13</a> )	ListBox
Option button (section <a href="#">2.4.1.14</a> )	OptionButton

Control	"xctname" value
Optional section (section <a href="#">2.4.1.18</a> )	Section
Repeating section (section <a href="#">2.4.1.15</a> )	RepeatingSection
Repeating table (section <a href="#">2.4.1.16</a> )	RepeatingTable
Rich text box (section <a href="#">2.4.1.17</a> )	RichText
Section (section <a href="#">2.4.1.18</a> )	Section
Text box (section <a href="#">2.4.1.20</a> )	PlainText

For custom controls, the value of this attribute **MUST** be set to the control's class identifier, 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

### Applicable Controls

The following controls **MAY** contain this 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 respective XSLT representations.

### Details

The value of this attribute **MUST** be equal to the name of an **xmlToEdit** (section [2.2.124](#)) element that exists in the form (1).

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
<a href="#">msxsl</a>	Specifies a string comparison function.
<a href="#">xdDate</a>	Specifies a set of date- and time-related functions.
<a href="#">xdEnvironment</a>	Specifies a set of functions that are related to the environment in which the form (1) is being filled out.
<a href="#">xdFormatting</a>	Specifies a set of string formatting functions..
<a href="#">xdImage</a>	Specifies a set of image related functions.
<a href="#">xdMath</a>	Specifies a set of mathematical functions.
<a href="#">xdUser</a>	Specifies a set of functions which are related to the user who is filling out the form (1).
<a href="#">xdUtil</a>	Specifies generic helper tools.
<a href="#">xdXDocument</a>	Specifies a set of functions which are related to the data of the form (1) being filled out.

### 2.4.3.1 msxsl

Microsoft XPath Extension Functions specifies a number of functions, one of which is supported by msxsl. Microsoft XPath Extension Functions are specified in [\[MSDN-XPATH\]](#).

#### 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:** Parameter MUST be a string. String is specified in [\[XPATH\]](#), section 3.6.
- **Second String:** 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 0 if they are equivalent strings; it MUST return 1 if second string comes before the first in lexicographical order; and it MUST return -1 if 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:

1. **date:** 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[ISO-8601] format to be a valid parameter. XPath expression is specified in [\[XPATH\]](#).
2. **days:** 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 (1) if both parameters are empty strings (1). 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:** 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:** 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 (1) if both parameters are empty strings (1). 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 (1) 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 (1) 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 (1) 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:** 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:** 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.
- **XSLT Expression:** Parameter MUST be a valid XSLT expression[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:** 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 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 which 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:** 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 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 which 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:** 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 "0".

#### 2.4.3.7 xdUser

xdUser contains a set of functions which are related to the user who is filling out the form (1).

##### 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:

1. **String**: Parameter MUST be a string.
2. **Regular Expression**: Parameter MUST be a valid XML Regular Expression.

The function MUST return TRUE if the input string conforms to the specified regular expression, 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 which are related to the data of the form (1) 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 which 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**: Parameter MUST be a string. It MUST be the name of a secondary data source (2).

The 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**: 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**: Parameter MUST be a string. It specifies the name of the property whose value is to be returned.
- **DefaultValue**: 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, protocol server SHOULD provide a mechanism to store string data.

The 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 [view](#) element in the form definition (.xsf) file. See section [3.5](#) for an example.

## 2.6 Submit Files (XML) Specification

For each [input](#) element inside of a [webServiceAdapter](#) element 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 one per 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 as 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

- 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 [2.6.2](#).
- Have a single child **dataFields** element.

Child Elements
<a href="#">dataFields</a>

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 which 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 WSDL.

Parent Elements
<a href="#">myFields</a>

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 XSL Transformation (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 sets 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 a 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

### 3.1 The InfoPath Form Template Format

#### 3.1.1 Simple Form Template

This example describes a simple form template (.xsn) file.

Simple.xsn

- manifest.xsf
- myschema.xsd
- template.xml
- sampledata.xml
- view1.xsl

The preceding files represent a very simple form (1) 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 (1).
- 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, which 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.

Complex.xsn

manifest.xsf

myschema.xsd

template.xml

sampledata.xml

view1.xsl

view2.xsl

IPTemplate\_bkgd.gif

741C3E77.gif

upgrade.xsl

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 example 3.1.
- This form (1) uses two view.xsl files to represent the form (1): view1.xsl and 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: IPTemplate\_bkgd.gif, 741C3E77.gif, and 70482F6B.gif. These images are used when displaying the form (1).

### 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](#) and [documentSchemas](#) elements 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:files>
  </xsf:package>
</xsf:xDocumentClass>
```

```

</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:view>
</xsf:views>

```

```

</xsf:editing>
<xsf:menuArea name="msoInsertMenu">
  <xsf:menu caption="&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
"CustomerKey","Title","FirstName","MiddleName","
LastName","BirthDate","MaritalStatus","Suffix","
Gender","EmailAddress","AddressLine1","AddressLine2"
from "dbo."DimCustomer as "DimCustomer""
queryAllowed="yes" name="Main connection" submitAllowed="no"></xsf:adoAdapter>
</xsf:query>
</xsf:xDocumentClass>

```

### 3.2.1 XSF Extension Examples

This sample [solutionDefinition](#) element specifies that this is a browser-compatible form template and its contents are verified at <http://www.someserver.com/verificationService>. The contained [offline](#) element specifies that the form (1) is loaded even if contained online queries fail and that the results of any queries are cached locally. The contained [server](#) element specifies:

1. The form template is not compatible with mobile Web browsers
2. The form view is rendered in US English
3. The form (1) 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 is a sample XML schema document (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".

1. my:field1 represents a Text Box control in the XSD which can have a string content.
2. my:group1 contains another group, my:group2. my:group2 is a repeating element. This is used to represent repeating controls, such as Repeating Section Control.

my:field2 represents the control inside the repeating control, which is a Text Box Control.

- my:field3 is a File Attachment Control.

### 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 sample 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 (2).

##### 3.4.1.1 Button Control

The following are XSL examples for [button](#) controls.

The following is an example of 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 will be 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 is an example of 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 will have 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 is an example of a button control that is used to submit the form data. The button display text is statically set to "Submit". This control has two conditional formatting settings. If the value of **my:field1** is equal to 1, the control will be disabled and have a yellow background color. Furthermore, if the value of **my:field2** is equal to "abc", the button display text will be bold and the control will have 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 is an example of a button control that is used to refresh the content of a secondary data source (2). The button display text is statically set to "Refresh". Conditional formatting is set such that if the value of **my:field1** is equal to 1, the control will be disabled and have 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 are XSL examples for [check box](#) controls.

The following is an example of a check box control with the values of "1" if the control is not checked, and "0" if the control is checked. When the user hovers over the control with the cursor, it will display 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='true'">
    <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
  </xsl:if>
</input>

```

The following is an example of a check box control with the values of "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 will be 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='true'">
    <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
  </xsl:if>
</input>

```

### 3.4.1.3 Contact Selector Control

The following is an XSL example for [contact selector](#) controls.

The following is an example of a contact selector control with conditional formatting. Conditional formatting is set such that if the value of **my:field1** is equal to "false", the control will be 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;&#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 are XSL examples for [date picker](#) controls.

The following is an example of a date picker control where **xd:datafmt** is equal to `"&quot;date&quot;;&quot;dateFormat:Short Date&quot;;"`. 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="&quot;date&quot;;&quot;dateFormat:Short Date&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, &quot;date&quot;;, &quot;dateFormat:Short
Date&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">
        
    </button>
</div>

```

The following is an example of a date picker control with conditional formatting. Conditional formatting is set such that if the value of **my:field3** is equal to "1900-01-01", the text for the control will be 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="&quot;date&quot;;&quot;dateFormat:Short Date&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 = &quot;1900-01-01&quot;;">FONT-WEIGHT: bold; TEXT-
DECORATION: line-through</xsl:when>
            </xsl:choose>
        </xsl:attribute>
    </span>
    <button class="xdDTButton" xd:xctname="DTPicker_DTButton" xd:innerCtrl="_DTButton"
tabIndex="-1">
        
    </button>
</div>

```

```

        </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, &quot;date&quot;, &quot;dateFormat:Short
Date;&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">
    
</button>
</div>

```

### 3.4.1.5 Drop-Down List Control

The following are XSL examples for [drop-down list](#) controls.

The following is an example of 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>

```

```

        </option>
    </select>

```

The following is an example of a drop-down list control with values that are dynamically generated from an external data source (2) 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 is an example of 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 is an example of 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:attribute>
    <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>
</select>

```

```

        </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 are XSL examples for [expression box](#) controls.

The following is an example of 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 is an example of an expression box control with conditional formatting and that is displaying the value of **my:field1**. **xd:datafmt** is equal to

"&quot;datetime&quot;;,&quot;dateFormat:Short Date;timeFormat:none;&quot;". 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="&quot;datetime&quot;;,&quot;dateFormat:Short
Date;timeFormat:none;&quot;;" xd:num="">
  <xsl:attribute name="style">
    WIDTH: 145px;
  <xsl:choose>
    <xsl:when test="my:field1 = &quot;1&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,&quot;datetime&quot;;,&quot;dateFormat:Short
Date;timeFormat:none;&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 is an XSL example for [file attachment](#) controls.

```
<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 are XSL examples for [hyperlink](#) controls.

The following is an example of a static hyperlink control.

```
<a href="http://www.contoso.com" xd:disableEditing="yes">http://www.contoso.com</a>
```

The following is an example of 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 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 are XSL examples for [list box](#) controls.

The following is an example of 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='a'">
      <xsl:attribute name="selected">selected</xsl:attribute>
    </xsl:if>A</option>
  <option value="b">
    <xsl:if test="my:field1='b'">
      <xsl:attribute name="selected">selected</xsl:attribute>
    </xsl:if>B</option>
  <option value="c">
    <xsl:if test="my:field1='c'">
      <xsl:attribute name="selected">selected</xsl:attribute>
    </xsl:if>C</option>
</select>

```

The following is an example of a list box control which looks up the selection options from a repeating group (1) within the main data source (2). The control will only display unique selection options. Conditional formatting is set such that if the value of **my:field1** is equal to "a", the control will have 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 = 'a'">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:GetXDocument')">
      <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 is an example of a list box control which looks up the selection options from a repeating group (1) in a secondary data source called "UserNameList". Conditional formatting is set such that if the value of my:field1 is equal to "a", the control will have 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 = 'a'">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('"UserNameList"')/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 are XSL examples for [option button](#) controls.

The following is an example of 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;2&quot;">
            <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
        </xsl:if>
    </input>
    2
</div>
</div>

```

```

☐

```

The following is an example of an option button control with conditional formatting. Conditional formatting is set such that if the value of **my:field3** is equal to "2", the control will be disabled.

```

☐

```

### 3.4.1.11 Repeating Section Control

The following is an XSL example for **repeating section** controls (section [2.4.1.15](#)).

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 equal to "true", the control will be disabled.

#### 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>
</span>

```

## 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>
</xsl:template>
```

### 3.4.1.12 Repeating Table Control

The following are XSL examples for [repeating table](#) controls.

The following is an example of a repeating table control which has three columns (2) containing a text box control inside each column (2). The repeating table also outputs a link with the text "Insert Item" which 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>
        </tr>
      </xsl:for-each>
    </tbody>
  </table>
  <div>
    <strong />
  </div>
</div>
```

```

        </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 is an example of a repeating table control which has three columns (2) containing a text box control inside each column (2). This repeating table also contains a footer. Conditional formatting is set such that if the value of **my:field8** is equal to "2", the control has a different background color. This control also causes a post back 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;lquot;))">
            <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 is an example of a repeating table control which has one column (2) with a text box inside. Conditional formatting is set such that if the value of **my:field1** is equal to "1", the control does not allow the user to insert or delete rows from the table. Note that this conditional formatting is placed outside of the repeating table element.

```

<span>
    <xsl:attribute name="style">
        <xsl:if test="my:field1 = &quot;l&quot; ">msos-xCollection-group8_16-
editing:disabled;</xsl:if>
    </xsl:attribute>

```

```

</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 is an XSL example for [rich text box](#) controls.

The following is an example of a rich text box control with conditional formatting set such that if the value of **my:field2** is equal to "false", the control will be 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 are XSL examples for **section** and **optional section** controls (section [2.4.1.18](#)).

The following is an example of a section control that can be digitally signed. This control contains a text box control.

## Section call

```
<xsl:apply-templates select="my:group1" mode="_1"/>
```

## Section body

```
<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></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;"></td>
```

```
  </td> </xsl:when> <xsl:otherwise>  </td>  <b><xsl:value-of select="xdXDocument:GetNamedNodeProperty(., 'SignedBy', '???')"/></b><span style="margin: 0pt 20pt">View details</span></div>  </td> </xsl:when> <xsl:otherwise>  </td>   | | | | |
```

The following is an example of an optional section control. This control contains a date picker control.

## Optional section call

```
<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>
```

## Optional section body

```
<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=""date";,"dateFormat:Short Date;""
          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,"date";,"dateFormat:Short
                    Date;");" />
              </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">
            
          </button>
        </div>
      </div>
    </div>
  </div>
</xsl:template>
```

### 3.4.1.15 Table Control

The following is an XSL example for [table](#) controls.

The following is an example for table control that is two rows by two columns (2) which has the value "1" in three of the four cells and a button control 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:xcname="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 are XSL examples for [text box](#) controls.

The following is an example of a text box control that is bound to **my:field1**.

```

<div>
  <span class="xdTextBox" hideFocus="1" title="" xd:binding="my:field1" tabIndex="0"
xd:xcname="PlainText" xd:CtrlId="CTRL1" style="WIDTH: 130px">
    <xsl:value-of select="my:field1"/>
  </span>
</div>

```

The following is an example of 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:xcname="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 is an example of a text box control which is bound to **my:field3**. The value of **xd:datafmt** is equal to "&quot;date&quot;;,&quot;locale:1061; dateFormat:d.MM.yyyy&quot;". 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 14<sup>th</sup>, 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="&quot;date&quot;;,&quot;locale:1061; dateFormat:d.MM.yyyy&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,&quot;date&quot;;,&quot;locale:1061;dateFormat:d.
MMMM yyyy'. a.';&quot;)" />
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="my:field3"/>
      </xsl:otherwise>
    </xsl:choose>
  </span>
</div>

```

The following is an example of a text box control which has conditional formatting and is bound to **my:field4**. Conditional formatting is set such that if the value of **my:field4** is equal to "abc", the text in the control will be 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 = &quot;abc&quot;">FONT-WEIGHT: bold</xsl:when>
    </xsl:choose>
    </xsl:attribute>
    <xsl:value-of select="my:field4"/>
  </span>

```

</div>

The following is an example of a text box control which has conditional formatting, data formatting and is bound to **my:field5**. Conditional formatting is set such that if the value of **my:field5** is equal to "def", the text in the control will be underlined. The value of **xd:datafmt** is equal to "&quot;time&quot;;&quot;locale:1033;timeFormat:hh:mm:ss tt&quot;". 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 with a submit action:

```
<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 which 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>
...
```

#### **xd:autoAdvance**

The following XSLT fragment is an example of a text box control with the "autoAdvance" attribute 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>
```

### **xd:auxDom**

The following XSLT fragment is an example of a button control with a "refresh" [action](#):

```
<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"/>
```

### **xd:binding**

The following XSLT is an example of a text box control bound to an XML field:

```
<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>
```

### **xd:bindingProperty**

The following XSLT fragment is an example of a custom control with a "bindingProperty" attribute 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>
```

### **xd:bindingType**

The following XSLT fragment is an example of a custom control with a "bindingType" attribute 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>
```

### **xd:boundProp**

The following XSLT fragment is an example of a text box control with a "boundProp" attribute 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=""currency";,";numDigits:0;negativeOrder:0;positiveOrder:0;currencyLo
cale:1033;"" 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,&quot;currency&quot;;,&quot;;numDigits:0;negativeO
rder:0;positiveOrder:0;currencyLocale:1033;&quot;)" />
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="my:field12"/>
    </xsl:otherwise>
  </xsl:choose>
</span>
```

### **xd:CtrlId**

The following XSLT fragment is an example of a text box control with a "CtrlId" attribute 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>
```

### **xd:datafmt**

The following XSLT fragment is an example of a text box control with currency data formatting specified:

```
<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=""currency&quot;;,&quot;;numDigits:0;negativeOrder:0;positiveOrder:0;currencyLo
cale:1033;&quot;" style="WIDTH: 130px">
  ...
</span>
```

### **xd:disableEditing**

The following XSLT fragment is an example of a hyperlink control with the "disableEditing" attribute 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>
  </a>
</span>
```

```

        </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 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>

```

### **xd:enabledProperty**

The following XSLT fragment is an example of an custom control with the "enabledProperty" attribute 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>

```

### **xd:enabledValue**

The following XSLT fragment is an example of an custom control with the "enabledValue" attribute 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>

```

## xd:innerCtrl

The following XSLT is an example of a date picker control:

```

<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="&quot;date&quot;;&quot;dateFormat:Short Date;&quot;" xd:innerCtrl="_DTText">
    ...
  </span>
  <button class="xdDTButton" xd:xctname="DTPicker_DTButton" xd:innerCtrl="_DTButton"
tabIndex="-1">
    
  </button>
</div>

```

## xd:num

The following XSLT fragment is an example of a text box control with an "xd:num" attribute:

```

<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="&quot;currency&quot;;&quot;;numDigits:0;negativeOrder:0;positiveOrder:0;currencyLoc
ale:1033;&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, &quot;currency&quot;;&quot;;numDigits:0;negativeO
rder:0;positiveOrder:0;currencyLocale:1033;&quot;)" />
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="my:field12"/>
    </xsl:otherwise>
  </xsl:choose>
</span>

```

## xd:offValue

The following XSLT fragment is an example of a check box control with the "offValue" attribute 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
```

### **xd:onValue**

The following XSLT fragment is an example of a check box control with the "onValue" attribute 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, each 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
```

## **xd:postbackModel**

The following XSLT fragment is an example of a text box control with the "postbackModel" attribute 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>
```

## **xd:SignatureBlock**

The following XSLT fragment is an example of a section control with the "SignatureBlock" attribute 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>
```

## **xd:SignedSectionDisplaySignatures**

The following XSLT fragment is an example of a section control with the "SignedSectionDisplaySignature" attribute 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>
```

## **xd:SignedSectionName**

The following XSLT fragment is an example of a section control with the "SignedSectionName" attribute 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>
```

## **xd:value**

The following XSLT fragment is an example of a check box control with an "xd:value" attribute:

```
<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='true'">
    <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
  </xsl:if>
</input>
```

```

        </xsl:if>
    </input> Field 4

```

### xd:xctname

The following XSLT fragment is an example of a text box control:

```

<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:

```

<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>

```

### xd:xmlToEdit

The following XSLT fragment is an example of an optional section control with the "xmlToEdit" attribute 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 usage of a selection of function extensions. XSL function extensions are specified in section [2.4.3](#).

The following sample 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:group1>
</my:myFields>

```

```

        </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>

```

### ▪ 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 will evaluate to true and xsl:when will output FONT-WEIGHT: bold and make the font bold.

### ▪ Conditional formatting with xdDate:AddDays, xdDate:Today

This example is similar to the first example in that it will show 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 will be 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 will add three days to today's date and will output the date three days past today. In the SampleXML1.xml snippet, my:Date is 2008-02-04. If today is 2008-02-01, test will evaluate to true and the style attribute will have the value FONT-WEIGHT: bold; COLOR: #ff0000.

### ▪ 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` will return the time equivalent to now plus the number of seconds indicated in `my:Seconds`. `Xsl:value-of` will output the resulting time.

- **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),
          &quot;\d\d\d\d\d&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. In the sample, this textbox is used for entering 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 will have the style `"COLOR: #ff0000; TEXT-DECORATION: line-through"` and it will make it more visible to the user that the entered zip code is not valid. In the `SampleXML1.xml`, `my:ZipCode` is `98052x` which doesn't match the given regular expression. Therefore test will evaluate to `TRUE` and text box control's style attribute will have the given style attributes.

The following sample 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>.
```

- **Getting data from a secondary data source using `xdXDocument:GetDOM`**

This example uses the `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 (2) for use within the XSL.

The example makes the font of a text box 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="Plain
Text"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-of select="my:field2"/>
</xsl:if>
</span>

```

In the preceding sample 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 (2). Then `xdMath:Avg` is used to calculate the average.

The following sample 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>

```

### ▪ 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, &quot;xdMath:Nz(my:Count) *
xdMath:Nz(my:Value) &quot;))"/>
</xsl:if>
</span>

```

Given is the XSL sample for Expression Box control. The value of the expression box is calculated from the 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`. `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 will calculate this multiplication for every group2 node and will return the result as a node set.

Count	Value
3	4
12	20
4	8

The output will be  $3*4 + 12 * 20 + 4*8 = 284$

### 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. 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".

This 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>
...
```

This 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>
...
```

This excerpt show how "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:views>
```

```

</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 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". The following is a fragment from the WSDL file containing the XML schema for GetRandom Web service 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://webservicesserver/Everett"
targetNamespace="http://webservicesserver/Everett"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:import namespace="http://tempuri.org/twrSchema.xsd"
location="http://webservicesserver/anon/Service1.asmx?schema=typedDataSet" />
  <wsdl:types>
    <s:schema elementFormDefault="qualified"
targetNamespace="http://webservicesserver/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 is how a Submit.xml would look when using the GetRandom Web service method in this example.

```
<dfs:myFields xmlns:dfs="http://schemas.microsoft.com/office/infopath/2003/dataFormSolution"
xmlns:tns="http://webservicesserver/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 Web service 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 sample.

### 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, first textbox which is bound to "my:field1" field will be populated with initial value of "Jean Philippe", and the second text box which is bound to "my:field2" field will be populated with 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: upgrade.xsl and one focused on the MSXSL:node-set() function.

### 3.8.1 Upgrade.XSL Example

The following is an example of a simple upgrade.xsl. The following 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:

1. my:field2 has been removed entirely. That data has been discarded.
2. my:myRepeatingGroup has a new field, my:fieldText, which is empty for all XML nodes under my:myRepeatingGroup.
3. The data for all other fields has been retained and moved to the new data source (2).

### 3.8.2 MSXSL:Node-Set() Example

In the following example, `$var`, is a variable that is a 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 a 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:

```
<?xml version="1.0" encoding="utf-8"?>

<authors><author>Michael Howard</author><author>Michael Kay</author></authors>
```

## 4 Security Considerations

### 4.1 The InfoPath Form Template Format

Because a form template (.xsn) file is a cabinet (.cab) file, as specified in [MC-MCF], any and all security considerations (including, but not limited to, digital signatures (1) affecting cabinet (.cab) files affect form template (.xsn) files.

### 4.2 Template.XML Specification

Because template.xml is a form file, as specified 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 specified 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}/\#\&quot;&gt;&lt;])(([^\p{Zl}\p{Zp}\p{C}/\#\&quot;&gt;&lt;])*)*([^\p{Z}\p{C}/\#\&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;\&#x2D;\&#x2E;,\]*" />
  </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:all>
  </xsd:complexType>
</xsd:element>

```

```

    <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:all>
    </xsd:complexType>
</xsd:element>

```

```

        <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:all>
    </xsd:complexType>
</xsd:element>

```

```

        </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:sequence>
            <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:sequence>
    </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:sequence>
            <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:sequence>
    </xsd:complexType>
</xsd:element>

```

```

        </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_actionontask_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>

```

```

        </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:complexType>
</xsd:element>

```

```

<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: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>

```

```

    </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:complexType>

```

```

        <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 specified 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/extension
  s"
  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: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:complexType>
</xsd:element>

```

```

        <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="xsd:boolean" 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="xsd:boolean" 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: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:complexType>
</xsd:element>

```

```

        <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="xsd:boolean" use="optional" />
<xsd:attribute name="showSignatureReminder" type="xsd:boolean" use="optional" />
<xsd:attribute name="disableMenuItem" type="xsd:boolean" use="optional" />
</xsd:complexType>
</xsd:element>

<xsd:element name="featureRestrictionsExtension">
    <xsd:complexType>
        <xsd:all>
            <xsd:element ref="xsd2:exportToPDFForXPS" minOccurs="0" />
        </xsd:all>
    </xsd:complexType>
</xsd:element>

<xsd:element name="exportToPDFForXPS">
    <xsd:complexType>
        <xsd:attribute name="ui" type="xsd:boolean" 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 Forms Server 2007
- Microsoft® Office InfoPath® 2007
- Microsoft® Office Enterprise 2007
- Microsoft® Office Ultimate 2007
- Microsoft® Office SharePoint® Server 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 Office InfoPath 2003 or Office InfoPath 2003 SP1.

[<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

### A

[action attribute - form view file](#) 298  
[action element - form definition file](#) 91  
[admin element - form definition file extension](#) 145  
[adoAdapter element - form definition file](#) 56  
[adoAdapterExtension element - form definition file extension](#) 152  
[allowedActions element - form definition file](#) 91  
[allowedControl element - form definition file](#) 99  
[allowedTasks element - form definition file](#) 92  
[allownonmatching attribute - form view file](#) 299  
Applicability ([section 1.5](#) 23, [section 1.5](#) 23)  
[applicationParameters element - form definition file](#) 47  
[assignmentAction element - form definition file](#) 128  
[attachmentFileName element - form definition file](#) 70  
[attributeData element - form definition file](#) 101  
[autoAdvance attribute - form view file](#) 299  
[autoRecovery element - form definition file](#) 51  
[autoUpdatePrompt element - form definition file extension](#) 161  
[auxDom attribute - form view file](#) 299

### B

[backgroundPicture attribute - form view file](#) 300  
[bcc element - form definition file](#) 68  
[binding attribute - form view file](#) 300  
[bindingProperty attribute - form view file](#) 301  
[bindingType attribute - form view file](#) 301  
[boundProp attribute - form view file](#) 302  
[business object - form template file component](#) 25  
[button \(with event\) element - form definition file](#) 107  
[Button control - form view file](#) 223  
[Button control - XML schema file](#) 168  
[Button control example](#) 331  
[button element - form definition file](#) 101

### C

[calculatedField element - form definition file](#) 132  
[calculations element - form definition file](#) 131  
[cc element - form definition file](#) 68  
[Change tracking](#) 410  
[Check box control - form view file](#) 230  
[Check box control - XML schema file](#) 168  
[Check box control example](#) 332  
[chooseFragment element - form definition file](#) 103  
[closeDocumentAction element - form definition file](#) 129  
[command element - form definition file extension](#) 140  
[commands element - form definition file extension](#) 140  
[compatibilityModesType simple type - form definition file extension](#) 135

[Complex form template example](#) 326  
[connectoid element - form definition file extension](#) 151  
[Contact selector control - form view file](#) 232  
[Contact selector control - XML schema file](#) 169  
[Contact selector control example](#) 333  
[contentType element - form definition file extension](#) 143  
[contentTypeTemplate element - form definition file extension](#) 144  
[Control data formatting - form view file](#) 222  
[Control representation examples - form view file](#) 331  
[Control-specific attributes example](#) 353  
[CtrlId attribute - form view file](#) 303  
[customCategory element - form definition file](#) 94  
[customValidation element - form definition file](#) 73

### D

[dataAdapters element - form definition file](#) 71  
[dataConnections element - form definition file extension](#) 149  
[dataFields element - submit file](#) 325  
[datafmt attribute - form view file](#) 304  
[dataObject element - form definition file](#) 54  
[dataObjects element - form definition file](#) 53  
[Date picker control - form view file](#) 236  
[Date picker control - XML schema file](#) 169  
[Date picker control example](#) 334  
[davAdapter element - form definition file](#) 63  
[davAdapterExtension element - form definition file extension](#) 152

#### Details

[action attribute - form view file](#) 298  
[action element - form definition file](#) 91  
[admin element - form definition file extension](#) 145  
[adoAdapter element - form definition file](#) 56  
[adoAdapterExtension element - form definition file extension](#) 152  
[allowedActions element - form definition file](#) 91  
[allowedControl element - form definition file](#) 99  
[allowedTasks element - form definition file](#) 92  
[allownonmatching attribute - form view file](#) 299  
[applicationParameters element - form definition file](#) 47  
[assignmentAction element - form definition file](#) 128  
[attachmentFileName element - form definition file](#) 70  
[attributeData element - form definition file](#) 101  
[autoAdvance attribute - form view file](#) 299  
[autoRecovery element - form definition file](#) 51  
[autoUpdatePrompt element - form definition file extension](#) 161  
[auxDom attribute - form view file](#) 299  
[backgroundPicture attribute - form view file](#) 300  
[bcc element - form definition file](#) 68

[binding attribute - form view file](#) 300  
[bindingProperty attribute - form view file](#) 301  
[bindingType attribute - form view file](#) 301  
[boundProp attribute - form view file](#) 302  
[business object - form template file component](#)  
 25  
[button \(with event\) element - form definition file](#)  
 107  
[button control - form view file](#) 223  
[button control - XML schema file](#) 168  
[button element - form definition file](#) 101  
[calculatedField element - form definition file](#) 132  
[calculations element - form definition file](#) 131  
[cc element - form definition file](#) 68  
[check box control - form view file](#) 230  
[check box control - XML schema file](#) 168  
[chooseFragment element - form definition file](#)  
 103  
[closeDocumentAction element - form definition](#)  
[file](#) 129  
[command element - form definition file extension](#)  
 140  
[commands element - form definition file](#)  
[extension](#) 140  
[compatibilityModesType simple type - form](#)  
[definition file extension](#) 135  
[connectoid element - form definition file](#)  
[extension](#) 151  
[contact selector control - form view file](#) 232  
[contact selector control - XML schema file](#) 169  
[contentType element - form definition file](#)  
[extension](#) 143  
[contentTypeTemplate element - form definition](#)  
[file extension](#) 144  
[control data formatting - form view file](#) 222  
[CtrlId attribute - form view file](#) 303  
[customCategory element - form definition file](#) 94  
[customValidation element - form definition file](#) 73  
[dataAdapters element - form definition file](#) 71  
[dataConnections element - form definition file](#)  
[extension](#) 149  
[dataFields element - submit file](#) 325  
[datafmt attribute - form view file](#) 304  
[dataObject element - form definition file](#) 54  
[dataObjects element - form definition file](#) 53  
[date picker control - form view file](#) 236  
[date picker control - XML schema file](#) 169  
[davAdapter element - form definition file](#) 63  
[davAdapterExtension element - form definition](#)  
[file extension](#) 152  
[dialogBoxExpressionAction element - form](#)  
[definition file](#) 127  
[dialogBoxMessageAction element - form definition](#)  
[file](#) 127  
[disableEditing attribute - form view file](#) 309  
[documentSchema element - form definition file](#)  
 72  
[documentSchemas element - form definition file](#)  
 72  
[documentSignatures element - form definition file](#)  
 120  
[documentVersionUpgrade element - form](#)  
[definition file](#) 122  
[domEventHandler element - form definition file](#)  
 76  
[domEventHandlers element - form definition file](#)  
 75  
[drop-down list control - form view file](#) 245  
[drop-down list control - XML schema file](#) 170  
[editing element - form definition file](#) 108  
[editWith element - form definition file](#) 104  
[emailAdapter element - form definition file](#) 65  
[emailAdapterExtension element - form definition](#)  
[file extension](#) 155  
[emailAttachmentType simple type - form](#)  
[definition file extension](#) 135  
[enabledProperty attribute - form view file](#) 310  
[enabledValue attribute - form view file](#) 310  
[errorCondition element - form definition file](#) 73  
[errorMessage element - form definition file](#)  
 (section 2.2.64 74, section 2.2.75 84)  
[errorMessage element - form definition file](#)  
[extension](#) 166  
[exitRuleSet element - form definition file](#) 127  
[exportToExcel element - form definition file](#) 50  
[exportToPDFForXPS element - form definition file](#)  
[extension](#) 167  
[exportToWeb element - form definition file](#) 50  
[expression box control - form view file](#) 255  
[expression box control - XML schema file](#) 170  
[extension element - form definition file](#) 124  
[extensions element - form definition file](#) 123  
[externalView element - form definition file](#) 100  
[externalViews element - form definition file](#) 99  
[featureRestrictions element - form definition file](#)  
 48  
[featureRestrictionsExtension element - form](#)  
[definition file extension](#) 166  
[field \(list view\) element - form definition file](#) 79  
[field element - form definition file](#) 63  
[fieldExtension element - form definition file](#)  
[extension](#) 149  
[fields \(list view\) element - form definition file](#) 78  
[fieldsExtension element - form definition file](#)  
[extension](#) 148  
[file attachment control - form view file](#) 258  
[file attachment control - XML schema file](#) 171  
[file element - form definition file](#) 95  
[fileName element - form definition file](#) 65  
[fileNew element - form definition file](#) 93  
[fileProperties element - form definition file](#) 96  
[files element - form definition file](#) 95  
[folderURL element - form definition file](#) 64  
[footer element - form definition file](#) 113  
[form definition file \(XSF\) extension specification](#)  
 132  
[form definition file structure](#) 25  
[form template file structure](#) 24  
[form view file - control-specific attributes](#) 296  
[form view file - view representation](#) 176  
[form view file - view syntax](#) 177  
[form view file - XSL function extensions](#) 319

[form view file \(XSLT\) structure](#) 175  
[formDescriptionType simple type - form definition file extension](#) 136  
[formLocaleType simple type - form definition file extension](#) 136  
[fragmentToInsert element - form definition file](#) 109  
[getUserNameFromData element - form definition file](#) 88  
[ghosted attribute - form view file](#) 311  
[group element - form definition file](#) 89  
[header element - form definition file](#) 113  
[hwsAdapter element - form definition file](#) 57  
[hwsOperation element - form definition file](#) 59  
[hwsWorkflow element - form definition file](#) 90  
[hyperlink control - form view file](#) 259  
[hyperlink control - XML schema file](#) 171  
[ictID attribute - form view file](#) 311  
[ictVersion attribute - form view file](#) 311  
[ignored controls - form view file](#) 295  
[importerrors.xml - form template file component](#) 25  
[importParameters element - form definition file](#) 77  
[importSource element - form definition file](#) 77  
[includedView element - form definition file extension](#) 147  
[includedViews element - form definition file extension](#) 146  
[initialXmlDocument element - form definition file](#) 93  
[inline attribute - form view file](#) 311  
[innerCtrl attribute - form view file](#) 311  
[input element - form definition file](#) 60  
[inputscope attribute - form view file](#) 312  
[inputScope element - form definition file extension](#) 162  
[inputScopeId attribute - form view file](#) 312  
[inputScopes element - form definition file extension](#) 161  
[install element - form definition file extension](#) 142  
[intro element - form definition file](#) 69  
[invalid constructs - form view file](#) 296  
[invalid controls - form view file](#) 295  
[irm template - form template file component](#) 25  
[layoutText attribute - form view file](#) 312  
[linkedToMaster attribute - form view file](#) 312  
[list box control - form view file](#) 261  
[list box control - XML schema file](#) 171  
[listProperties element - form definition file](#) 78  
[listPropertiesExtension element - form definition file extension](#) 148  
[location element - form definition file](#) 90  
[mail element - form definition file extension](#) 145  
[mainpane element - form definition file](#) 109  
[managedCode element - form definition file extension](#) 163  
[managedCodeType simple type - form definition file extension](#) 136  
[manifest.xsf - form template file component](#) 24  
[masterDetail element - form definition file](#) 108  
[masterID attribute - form view file](#) 312  
[masterName attribute - form view file](#) 313  
[membership element - form definition file](#) 88  
[menu element - form definition file](#) 114  
[menuArea element - form definition file](#) 115  
[merge.xml - form template file component](#) 25  
[mergedPrintView element - form definition file extension](#) 146  
[message element - form definition file](#) 122  
[msxsl function extension - form view file](#) 319  
[MSXSL Node-Set\(\) function - upgrade.XSL file](#) 325  
[myFields element - submit file](#) 324  
[num attribute - form view file](#) 313  
[offline element - form definition file extension](#) 147  
[offValue attribute - form view file](#) 313  
[onLoad element - form definition file](#) 85  
[onValue attribute - form view file](#) 314  
[openNewDocumentAction element - form definition file](#) 129  
[operation element - form definition file](#) 58  
[option button control - form view file](#) 266  
[option button control - XML schema file](#) 172  
[override element - form definition file](#) 46  
[package element - form definition file](#) 94  
[partFragment element - form definition file](#) 60  
[permissions element - form definition file](#) 98  
[postbackModel attribute - form view file](#) 314  
[preview element - form definition file extension](#) 160  
[primaryschema.xsd - form template file component](#) 24  
[print element - form definition file](#) 50  
[print view file - structure](#) 324  
[printSettings element - form definition file](#) 110  
[property element - form definition file](#) 96  
[query \(secondary\) element - form definition file](#) 55  
[query element - form definition file](#) 52  
[queryAction element - form definition file](#) 129  
[ref attribute - form view file](#) 315  
[relativeQuery element - form definition file extension](#) 154  
[repeating section control - form view file](#) 267  
[repeating section control - XML schema file](#) 173  
[repeating table control - form view file](#) 270  
[repeating table control - XML schema file](#) 173  
[resource files - form template file component](#) 25  
[rich text box control - form view file](#) 272  
[rich text box control - XML schema file](#) 174  
[role element - form definition file](#) 87  
[roles element - form definition file](#) 86  
[rule element - form definition file](#) 125  
[ruleSet element - form definition file](#) 130  
[ruleSetAction element - form definition file](#) 124  
[ruleSets element - form definition file](#) 130  
[sampledata.xml - form template file component](#) 24  
[save \(disabled\) element - form definition file](#) 86  
[save element - form definition file](#) 49

[schemaErrorMessages element - form definition file](#) 46  
[script element - form definition file](#) 53  
[script.js - form template file component](#) 25  
[script.vbs - form template file component](#) 25  
[scripts element - form definition file](#) 52  
[secondaryschema.xsd - form template file component](#) 24  
[secondaryschema\\_offline.xml - form template file component](#) 25  
[section/optional section control - form view file](#) 275  
[section/optional section control - XML schema file](#) 174  
[sendByMail element - form definition file extension](#) 157  
[sendMail element - form definition file](#) 51  
[server element - form definition file extension](#) 138  
[serverCommandActionType simple type - form definition file extension](#) 134  
[share element - form definition file extension](#) 144  
[sharepointListAdapter element - form definition file](#) 62  
[sharepointListAdapterExtension element - form definition file extension](#) 156  
[SignatureBlock attribute - form view file](#) 315  
[signedDataBlock element - form definition file](#) 121  
[SignedSectionDisplaySignatures attribute - form view file](#) 316  
[SignedSectionName attribute - form view file](#) 316  
[solutionDefinition element - form definition file extension](#) 137  
[solutionProperties element - form definition file](#) 47  
[solutionPropertiesExtension element - form definition file extension](#) 141  
[solutionType simple type - form definition file extension](#) 135  
[subject element - form definition file](#) 69  
[submit element - form definition file](#) 81  
[submit element - form definition file extension](#) 164  
[submit file - structure](#) 324  
[submitAction element - form definition file \(section 2.2.73 83, section 2.2.134 126\)](#)  
[submitAction element - form definition file extension](#) 165  
[submitdata.xml - form template file component](#) 24  
[submitToHostAdapter element - form definition file](#) 70  
[successMessage element - form definition file](#) 83  
[successMessage element - form definition file extension](#) 166  
[switchViewAction element - form definition file](#) 128  
[table control - form view file](#) 283  
[table control - XML schema file](#) 175  
[task element - form definition file](#) 92  
[taskpane element - form definition file](#) 116  
[template.xml - form template file component](#) 24  
[template.XML file - structure](#) 325  
[text box control - form view file](#) 284  
[text box control - XML schema file](#) 175  
[to element - form definition file](#) 67  
[toolbar element - form definition file](#) 114  
[toolbar element - form definition file extension](#) 139  
[unboundControls element - form definition file](#) 106  
[upgrade.xsl - form template file component](#) 25  
[upgrade.XSL file - structure](#) 325  
[useHttpHandler element - form definition file](#) 84  
[useHttpHandlerExtension element - form definition file extension](#) 150  
[useQueryAdapter element - form definition file](#) 85  
[userName element - form definition file](#) 89  
[useScriptHandler element - form definition file](#) 85  
[useTransform element - form definition file](#) 122  
[value attribute - form view file](#) 317  
[view element - form definition file](#) 118  
[view.xsl - form template file component](#) 24  
[viewExtension element - form definition file extension](#) 159  
[views element - form definition file](#) 117  
[viewsExtension element - form definition file extension](#) 158  
[warning element - form definition file extension](#) 158  
[warnings element - form definition file extension](#) 157  
[webServiceAdapter element - form definition file](#) 56  
[webServiceAdapterExtension element - form definition file extension](#) 153  
[word element - form definition file extension](#) 163  
[words element - form definition file extension](#) 162  
[wss element - form definition file extension](#) 143  
[xctname attribute - form view file](#) 317  
[xdDate function extension - form view file](#) 319  
[xdDesignMode simple type - form definition file](#) 39  
[xdEmptyString simple type - form definition file](#) 39  
[xdEnabledDisabled simple type - form definition file](#) 36  
[xdEnvironment function extension - form view file](#) 320  
[xdErrorMessage simple type - form definition file](#) 39  
[xdExpressionLiteral simple type - form definition file](#) 37  
[xdFileName simple type - form definition file](#) 37  
[xdFormatting function extension - form view file](#) 321  
[xdHWSCaption simple type - form definition file](#) 42  
[xdHWSname simple type - form definition file](#) 42  
[xdImage function extension - form view file](#) 321

[xdManualAuto simple type - form definition file](#) 36  
[xdMath function extension - form view file](#) 321  
[xDocumentClass element - form definition file](#) 42  
[xdRoleName simple type - form definition file](#) 33  
[xdScriptLanguage simple type - form definition file](#) 38  
[xdSignatureRelationEnum simple type - form definition file](#) 41  
[xdSignedDataBlockMessage simple type - form definition file](#) 41  
[xdSignedDataBlockName simple type - form definition file](#) 40  
[xdSolutionVersion simple type - form definition file](#) 38  
[xdTitle simple type - form definition file](#) 31  
[xdTrustLevel simple type - form definition file](#) 40  
[xdUser function extension - form view file](#) 322  
[xdUtil function extension - form view file](#) 323  
[xdViewName simple type - form definition file](#) 32  
[xdXDocument function extension - form view file](#) 323  
[xdYesNo simple type - form definition file](#) 33  
[XML schema file - control representation](#) 167  
[XML schema file structure](#) 167  
[xmlFileAdapter element - form definition file](#) 61  
[xmlFileAdapterExtension element - form definition file extension](#) 155  
[xmlToEdit attribute - form view file](#) 318  
[xmlToEdit element - form definition file](#) 119  
[xmlToEditExtension element - form definition file extension](#) 160  
[XSL root template element - form view file](#) 181  
[XSL root template style sheets - form view file](#) 183  
[dialogBoxExpressionAction element - form definition file](#) 127  
[dialogBoxMessageAction element - form definition file](#) 127  
[disableEditing attribute - form view file](#) 309  
[documentSchema element - form definition file](#) 72  
[documentSchemas element - form definition file](#) 72  
[documentSignatures element - form definition file](#) 120  
[documentVersionUpgrade element - form definition file](#) 122  
[domEventHandler element - form definition file](#) 76  
[domEventHandlers element - form definition file](#) 75  
[Drop-down list control - form view file](#) 245  
[Drop-down list control - XML schema file](#) 170  
[Drop-down list control example](#) 335

## E

[editing element - form definition file](#) 108  
[editWith element - form definition file](#) 104  
[emailAdapter element - form definition file](#) 65  
[emailAdapterExtension element - form definition file extension](#) 155  
[emailAttachmentType simple type - form definition file extension](#) 135  
[enabledProperty attribute - form view file](#) 310

[enabledValue attribute - form view file](#) 310  
[errorCondition element - form definition file](#) 73  
[errorMessage element - form definition file \(section 2.2.64 74, section 2.2.75 84\)](#)  
[errorMessage element - form definition file extension](#) 166

## Examples

[button control](#) 331  
[check box control](#) 332  
[complex form template](#) 326  
[contact selector control](#) 333  
[Control representation - form view file](#) 331  
[control-specific attributes](#) 353  
[date picker control](#) 334  
[drop-down list control](#) 335  
[expression box control](#) 338  
[file attachment control](#) 339  
[form definition file](#) 327  
[form view file](#) 331  
[hyperlink control](#) 339  
[list box control](#) 340  
[MSXSL Node-Set\(\)](#) 369  
[option button control](#) 342  
[overview](#) 326  
[print view file](#) 364  
[repeating section control](#) 343  
[repeating table control](#) 344  
[rich text box control](#) 347  
[section/optional section control](#) 347  
[simple form template](#) 326  
[submit file](#) 365  
[table control](#) 350  
[Template.XML](#) 366  
[text box control](#) 351  
[Upgrade.XSL](#) 367  
[XML schema file](#) 330  
[XSF extension](#) 329  
[XSL function extensions](#) 360  
[exitRuleSet element - form definition file](#) 127  
[exportToExcel element - form definition file](#) 50  
[exportToPDForXPS element - form definition file extension](#) 167  
[exportToWeb element - form definition file](#) 50  
[Expression box control - form view file](#) 255  
[Expression box control - XML schema file](#) 170  
[Expression box control example](#) 338  
[extension element - form definition file](#) 124  
[extensions element - form definition file](#) 123  
[externalView element - form definition file](#) 100  
[externalViews element - form definition file](#) 99

## F

[featureRestrictions element - form definition file](#) 48  
[featureRestrictionsExtension element - form definition file extension](#) 166  
[field \(list view\) element - form definition file](#) 79  
[field element - form definition file](#) 63  
[fieldExtension element - form definition file extension](#) 149  
**Fields**  
[vendor-extensible](#) 23

[Fields - vendor-extensible](#) 23  
[fields \(list view\) element - form definition file](#) 78  
[fieldsExtension element - form definition file extension](#) 148  
[File attachment control - form view file](#) 258  
[File attachment control - XML schema file](#) 171  
[File attachment control example](#) 339  
[file element - form definition file](#) 95  
[fileName element - form definition file](#) 65  
[fileNew element - form definition file](#) 93  
[fileProperties element - form definition file](#) 96  
[files element - form definition file](#) 95  
[folderURL element - form definition file](#) 64  
[footer element - form definition file](#) 113  
[Form definition \(.xsf\) file- overview](#) 15  
[Form definition file \(XSF\) extension specification](#) 132  
 Form definition file elements  
   [action](#) 91  
   [adoAdapter](#) 56  
   [allowedActions](#) 91  
   [allowedControl](#) 99  
   [allowedTasks](#) 92  
   [applicationParameters](#) 47  
   [assignmentAction](#) 128  
   [attachmentFileName](#) 70  
   [attributeData](#) 101  
   [autoRecovery](#) 51  
   [bcc](#) 68  
   [button](#) 101  
   [button \(with event\)](#) 107  
   [calculatedField](#) 132  
   [calculations](#) 131  
   [cc](#) 68  
   [chooseFragment](#) 103  
   [closeDocumentAction](#) 129  
   [customCategory](#) 94  
   [customValidation](#) 73  
   [dataAdapters](#) 71  
   [dataObject](#) 54  
   [dataObjects](#) 53  
   [davAdapter](#) 63  
   [dialogBoxExpressionAction](#) 127  
   [dialogBoxMessageAction](#) 127  
   [documentSchema](#) 72  
   [documentSchemas](#) 72  
   [documentSignatures](#) 120  
   [documentVersionUpgrade](#) 122  
   [domEventHandler](#) 76  
   [domEventHandlers](#) 75  
   [editing](#) 108  
   [editWith](#) 104  
   [emailAdapter](#) 65  
   [errorCondition](#) 73  
   [errorMessage \(section 2.2.64](#) 74, [section 2.2.75](#) 84)  
   [exitRuleSet](#) 127  
   [exportToExcel](#) 50  
   [exportToWeb](#) 50  
   [extension](#) 124  
   [extensions](#) 123  
   [externalView](#) 100  
   [externalViews](#) 99  
   [featureRestrictions](#) 48  
   [field](#) 63  
   [field \(list view\)](#) 79  
   [fields \(list view\)](#) 78  
   [file](#) 95  
   [fileName](#) 65  
   [fileNew](#) 93  
   [fileProperties](#) 96  
   [files](#) 95  
   [folderURL](#) 64  
   [footer](#) 113  
   [fragmentToInsert](#) 109  
   [getUserNameFromData](#) 88  
   [group](#) 89  
   [header](#) 113  
   [hwsAdapter](#) 57  
   [hwsOperation](#) 59  
   [hwsWorkflow](#) 90  
   [importParameters](#) 77  
   [importSource](#) 77  
   [initialXmlDocument](#) 93  
   [input](#) 60  
   [intro](#) 69  
   [listProperties](#) 78  
   [location](#) 90  
   [mainpane](#) 109  
   [masterDetail](#) 108  
   [membership](#) 88  
   [menu](#) 114  
   [menuArea](#) 115  
   [message](#) 122  
   [onLoad](#) 85  
   [openNewDocumentAction](#) 129  
   [operation](#) 58  
   [override](#) 46  
   [package](#) 94  
   [partFragment](#) 60  
   [permissions](#) 98  
   [print](#) 50  
   [printSettings](#) 110  
   [property](#) 96  
   [query](#) 52  
   [query \(secondary\)](#) 55  
   [queryAction](#) 129  
   [role](#) 87  
   [roles](#) 86  
   [rule](#) 125  
   [ruleSet](#) 130  
   [ruleSetAction](#) 124  
   [ruleSets](#) 130  
   [save](#) 49  
   [save \(disabled\)](#) 86  
   [schemaErrorMessages](#) 46  
   [script](#) 53  
   [scripts](#) 52  
   [sendMail](#) 51  
   [sharepointListAdapter](#) 62  
   [signedDataBlock](#) 121  
   [solutionProperties](#) 47

- [subject](#) 69
- [submit](#) 81
- [submitAction](#) ([section 2.2.73](#) 83, [section 2.2.134](#) 126)
- [submitToHostAdapter](#) 70
- [successMessage](#) 83
- [switchViewAction](#) 128
- [task](#) 92
- [taskpane](#) 116
- [to](#) 67
- [toolbar](#) 114
- [unboundControls](#) 106
- [useHttpHandler](#) 84
- [useQueryAdapter](#) 85
- [userName](#) 89
- [useScriptHandler](#) 85
- [useTransform](#) 122
- [views](#) 117
- [webServiceAdapter](#) 56
- [xDocumentClass](#) 42
- [xmlFileAdapter](#) 61
- [xmlToEdit](#) ([section 2.2.123](#) 118, [section 2.2.124](#) 119)
- [Form definition file example](#) 327
- Form definition file extension elements
  - [admin](#) 145
  - [adoAdapterExtension](#) 152
  - [autoUpdatePrompt](#) 161
  - [command](#) 140
  - [commands](#) 140
  - [connectoid](#) 151
  - [contentType](#) 143
  - [contentTypeTemplate](#) 144
  - [dataConnections](#) 149
  - [davAdapterExtension](#) 152
  - [emailAdapterExtension](#) 155
  - [errorMessage](#) 166
  - [exportToPDFForXPS](#) 167
  - [featureRestrictionsExtension](#) 166
  - [fieldExtension](#) 149
  - [fieldsExtension](#) 148
  - [includedView](#) 147
  - [includedViews](#) 146
  - [inputScope](#) 162
  - [inputScopes](#) 161
  - [install](#) 142
  - [listPropertiesExtension](#) 148
  - [mail](#) 145
  - [managedCode](#) 163
  - [mergedPrintView](#) 146
  - [offline](#) 147
  - [preview](#) 160
  - [relativeQuery](#) 154
  - [sendByMail](#) 157
  - [server](#) 138
  - [share](#) 144
  - [sharepointListAdapterExtension](#) 156
  - [solutionDefinition](#) 137
  - [solutionPropertiesExtension](#) 141
  - [submit](#) 164
  - [submitAction](#) 165
  - [successMessage](#) 166
  - [toolbar](#) 139
  - [useHttpHandlerExtension](#) 150
  - [viewExtension](#) 159
  - [viewsExtension](#) 158
  - [warning](#) 158
  - [warnings](#) 157
  - [webServiceAdapterExtension](#) 153
  - [word](#) 163
  - [words](#) 162
  - [wss](#) 143
  - [xmlFileAdapterExtension](#) 155
  - [xmlToEditExtension](#) 160
- Form definition file extension simple types
  - [compatibilityModesType](#) 135
  - [emailAttachmentType](#) 135
  - [formDescriptionType](#) 136
  - [formLocaleType](#) 136
  - [managedCodeType](#) 136
  - [serverCommandActionType](#) 134
  - [solutionType](#) 135
- Form definition file simple types
  - [xdDesignMode](#) 39
  - [xdEmptyString](#) 39
  - [xdEnabledDisabled](#) 36
  - [xdErrorMessage](#) 39
  - [xdExpressionLiteral](#) 37
  - [xdFileName](#) 37
  - [xdHWSCaption](#) 42
  - [xdHWSname](#) 42
  - [xdManualAuto](#) 36
  - [xdRoleName](#) 33
  - [xdScriptLanguage](#) 38
  - [xdSignatureRelationEnum](#) 41
  - [xdSignedDataBlockMessage](#) 41
  - [xdSignedDataBlockName](#) 40
  - [xdSolutionVersion](#) 38
  - [xdTitle](#) 31
  - [xdTrustLevel](#) 40
  - [xdViewName](#) 32
  - [xdYesNo](#) 33
- [Form definition file structure](#) 25
- Form template file components
  - [business object](#) 25
  - [importerrors.xml](#) 25
  - [irm\\_template](#) 25
  - [manifest.xsf](#) 24
  - [merge.xsl](#) 25
  - [primaryschema.xsd](#) 24
  - [resource files](#) 25
  - [sampledata.xml](#) 24
  - [script.js](#) 25
  - [script.vbs](#) 25
  - [secondaryschema.xsd](#) 24
  - [secondaryschema\\_offline.xml](#) 25
  - [submitdata.xml](#) 24
  - [template.xml](#) 24
  - [upgrade.xsl](#) 25
  - [view.xsl](#) 24
- [Form template file structure](#) 24
- [Form template format - overview](#) 14

Form view file  
[control data formatting](#) 222  
[invalid constructs](#) 296  
[invalid controls](#) 295  
[view syntax](#) 177  
[XSL root template style sheets](#) 183  
[Form view file - control-specific attributes](#) 296  
[Form view file - view representation](#) 176  
[Form view file - XSL function extensions](#) 319  
[Form view file \(XSLT\) structure](#) 175  
Form view file attributes  
[action](#) 298  
[allownonmatching](#) 299  
[autoAdvance](#) 299  
[auxDom](#) 299  
[backgroundPicture](#) 300  
[binding](#) 300  
[bindingProperty](#) 301  
[bindingType](#) 301  
[boundProp](#) 302  
[CtrlId](#) 303  
[datafmt](#) 304  
[disableEditing](#) 309  
[enabledProperty](#) 310  
[enabledValue](#) 310  
[ghosted](#) 311  
[ictID](#) 311  
[ictVersion](#) 311  
[inline](#) 311  
[innerCtrl](#) 311  
[inputscope](#) 312  
[inputScopeId](#) 312  
[layoutText](#) 312  
[linkedToMaster](#) 312  
[masterID](#) 312  
[masterName](#) 313  
[num](#) 313  
[offValue](#) 313  
[onValue](#) 314  
[postbackModel](#) 314  
[ref](#) 315  
[SignatureBlock](#) 315  
[SignedSectionDisplaySignatures](#) 316  
[SignedSectionName](#) 316  
[value](#) 317  
[xctname](#) 317  
[xmlToEdit](#) 318  
Form view file controls  
[button](#) 223  
[check box](#) 230  
[contact selector](#) 232  
[date picker](#) 236  
[drop-down list](#) 245  
[expression box](#) 255  
[file attachment](#) 258  
[hyperlink](#) 259  
[Ignored](#) 295  
[list box](#) 261  
[option button](#) 266  
[repeating section](#) 267  
[repeating table](#) 270

[rich text box](#) 272  
[section/optional section](#) 275  
[table](#) 283  
[text box](#) 284  
Form view file elements  
[XSL root template](#) 181  
[Form view file example](#) 331  
Form view file function extensions  
[msxsl](#) 319  
[xdDate](#) 319  
[xdEnvironment](#) 320  
[xdFormatting](#) 321  
[xdImage](#) 321  
[xdMath](#) 321  
[xdUser](#) 322  
[xdUtil](#) 323  
[xdXDocument](#) 323  
[formDescriptionType simple type - form definition file extension](#) 136  
[formLocaleType simple type - form definition file extension](#) 136  
[fragmentToInsert element - form definition file](#) 109  
[Full XML schema](#) 371  
[InfoPath XSF XSD](#) 371  
[InfoPath XSF2 XSD](#) 399  
[the InfoPath XSF2 XSD](#) 399

## G

[getUserNameFromData element - form definition file](#) 88  
[ghosted attribute - form view file](#) 311  
Glossary ([section 1.1](#) 11, [section 1.1](#) 11)  
[group element - form definition file](#) 89

## H

[header element - form definition file](#) 113  
[hwsAdapter element - form definition file](#) 57  
[hwsOperation element - form definition file](#) 59  
[hwsWorkflow element - form definition file](#) 90  
[Hyperlink control - form view file](#) 259  
[Hyperlink control - XML schema file](#) 171  
[Hyperlink control example](#) 339

## I

[ictID attribute - form view file](#) 311  
[ictVersion attribute - form view file](#) 311  
[Ignored controls - form view file](#) 295  
[importerrors.xml - form template file component](#) 25  
[importParameters element - form definition file](#) 77  
[importSource element - form definition file](#) 77  
[includedView element - form definition file extension](#) 147  
[includedViews element - form definition file extension](#) 146  
Informative references ([section 1.2.2](#) 14, [section 1.2.2](#) 14)  
[initialXmlDocument element - form definition file](#) 93  
[inline attribute - form view file](#) 311  
[innerCtrl attribute - form view file](#) 311

[input element - form definition file](#) 60  
[inputscope attribute - form view file](#) 312  
[inputScope element - form definition file extension](#) 162  
[inputScopeId attribute - form view file](#) 312  
[inputScopes element - form definition file extension](#) 161  
[install element - form definition file extension](#) 142  
[intro element - form definition file](#) 69  
 Introduction ([section 1 11](#), [section 1 11](#))  
[Invalid constructs - form view file](#) 296  
[Invalid controls - form view file](#) 295  
[irm\\_template - form template file component](#) 25

## L

[layoutText attribute - form view file](#) 312  
[linkedToMaster attribute - form view file](#) 312  
[List box control - form view file](#) 261  
[List box control - XML schema file](#) 171  
[List box control example](#) 340  
[listProperties element - form definition file](#) 78  
[listPropertiesExtension element - form definition file extension](#) 148  
 Localization ([section 1.6 23](#), [section 1.6 23](#))  
[location element - form definition file](#) 90

## M

[mail element - form definition file extension](#) 145  
[mainpane element - form definition file](#) 109  
[managedCode element - form definition file extension](#) 163  
[managedCodeType simple type - form definition file extension](#) 136  
[manifest.xsf - form template file component](#) 24  
[masterDetail element - form definition file](#) 108  
[masterID attribute - form view file](#) 312  
[masterName attribute - form view file](#) 313  
[membership element - form definition file](#) 88  
[menu element - form definition file](#) 114  
[menuArea element - form definition file](#) 115  
[merge.xsl - form template file component](#) 25  
[mergedPrintView element - form definition file extension](#) 146  
[message element - form definition file](#) 122  
[msxsl function extension - form view file](#) 319  
[MSXSL Node-Set\(\) example](#) 369  
[MSXSL Node-Set\(\) function - upgrade.XSL file](#) 325  
[myFields element - submit file](#) 324

## N

Normative references ([section 1.2.1 12](#), [section 1.2.1 12](#))  
[num attribute - form view file](#) 313

## O

[offline element - form definition file extension](#) 147  
[offValue attribute - form view file](#) 313  
[onLoad element - form definition file](#) 85

[onValue attribute - form view file](#) 314  
[openNewDocumentAction element - form definition file](#) 129  
[operation element - form definition file](#) 58  
[Option button control - form view file](#) 266  
[Option button control - XML schema file](#) 172  
[Option button control example](#) 342  
[override element - form definition file](#) 46  
[Overview](#) 14  
 Overview (synopsis)  
[form definition \(.xsf\) file](#) 15  
[form template format](#) 14  
[form view files xsl function extensions \(XSLT\)](#) 16  
[print view files \(XSLT\)](#) 21  
[resource files](#) 22  
[submit files \(XML\)](#) 21  
[template.xml file](#) 22  
[unused files](#) 22  
[upgrade.xsl file](#) 22  
[XML schema files \(XSD\)](#) 15

## P

[package element - form definition file](#) 94  
[partFragment element - form definition file](#) 60  
[permissions element - form definition file](#) 98  
[postbackModel attribute - form view file](#) 314  
[preview element - form definition file extension](#) 160  
[primaryschema.xsd - form template file component](#) 24  
[print element - form definition file](#) 50  
[Print view file - structure](#) 324  
[Print view file examples](#) 364  
[printSettings element - form definition file](#) 110  
[Product behavior](#) 409  
[overview](#) 409  
[property element - form definition file](#) 96

## Q

[query \(secondary\) element - form definition file](#) 55  
[query element - form definition file](#) 52  
[queryAction element - form definition file](#) 129

## R

[ref attribute - form view file](#) 315  
 References  
     informative ([section 1.2.2 14](#), [section 1.2.2 14](#))  
     normative ([section 1.2.1 12](#), [section 1.2.1 12](#))  
     [overview](#) 12  
 Relationship to protocols and other structures ([section 1.4 23](#), [section 1.4 23](#))  
[relativeQuery element - form definition file extension](#) 154  
[Repeating section control - form view file](#) 267  
[Repeating section control - XML schema file](#) 173  
[Repeating section control example](#) 343  
[Repeating table control - form view file](#) 270  
[Repeating table control - XML schema file](#) 173  
[Repeating table control example](#) 344  
[Resource files - form template file component](#) 25

[Rich text box control - form view file](#) 272  
[Rich text box control - XML schema file](#) 174  
[Rich text box control example](#) 347  
[role element - form definition file](#) 87  
[roles element - form definition file](#) 86  
[rule element - form definition file](#) 125  
[ruleSet element - form definition file](#) 130  
[ruleSetAction element - form definition file](#) 124  
[ruleSets element - form definition file](#) 130

## S

[sampledata.xml - form template file component](#) 24  
[save \(disabled\) element - form definition file](#) 86  
[save element - form definition file](#) 49  
[schemaErrorMessages element - form definition file](#) 46  
[script element - form definition file](#) 53  
[script.js - form template file component](#) 25  
[script.vbs - form template file component](#) 25  
[scripts element - form definition file](#) 52  
[secondaryschema.xsd - form template file component](#) 24  
[secondaryschema\\_offline.xml - form template file component](#) 25  
[Section/optional section control - form view file](#) 275  
[Section/optional section control - XML schema file](#) 174  
[Section/optional section control example](#) 347  
[Security - form template format](#) 370  
[Security - Template.XML specification](#) 370  
[Security considerations overview](#) 370  
[sendByMail element - form definition file extension](#) 157  
[sendMail element - form definition file](#) 51  
[server element - form definition file extension](#) 138  
[serverCommandActionType simple type - form definition file extension](#) 134  
[share element - form definition file extension](#) 144  
[sharepointListAdapter element - form definition file](#) 62  
[sharepointListAdapterExtension element - form definition file extension](#) 156  
[SignatureBlock attribute - form view file](#) 315  
[signedDataBlock element - form definition file](#) 121  
[SignedSectionDisplaySignatures attribute - form view file](#) 316  
[SignedSectionName attribute - form view file](#) 316  
[Simple form template example](#) 326  
[solutionDefinition element - form definition file extension](#) 137  
[solutionProperties element - form definition file](#) 47  
[solutionPropertiesExtension element - form definition file extension](#) 141  
[solutionType simple type - form definition file extension](#) 135  
[Structures](#)  
[form definition file](#) 25  
[form template file](#) 24  
[form view file - control-specific attributes](#) 296  
[form view file - view representation](#) 176

[form view file - XSL function extensions](#) 319  
[form view file \(XSLT\)](#) 175  
[overview](#) 24  
[print view file](#) 324  
[submit file](#) 324  
[template.XML file](#) 325  
[upgrade.XSL file](#) 325  
[XML schema file](#) 167  
[XML schema file - control representation](#) 167  
[subject element - form definition file](#) 69  
[submit element - form definition file](#) 81  
[submit element - form definition file extension](#) 164  
[Submit file - structure](#) 324  
[Submit file elements](#)  
[dataFields](#) 325  
[myFields](#) 324  
[Submit file examples](#) 365  
[submitAction element - form definition file \(\[section 2.2.73\]\(#\) 83, \[section 2.2.134\]\(#\) 126\)](#)  
[submitAction element - form definition file extension](#) 165  
[submitdata.xml - form template file component](#) 24  
[submitToHostAdapter element - form definition file](#) 70  
[successMessage element - form definition file](#) 83  
[successMessage element - form definition file extension](#) 166  
[switchViewAction element - form definition file](#) 128

## T

[Table control - form view file](#) 283  
[Table control - XML schema file](#) 175  
[Table control example](#) 350  
[task element - form definition file](#) 92  
[taskpane element - form definition file](#) 116  
[template.xml - form template file component](#) 24  
[Template.XML example](#) 366  
[Template.XML file - structure](#) 325  
[Text box control - form view file](#) 284  
[Text box control - XML schema file](#) 175  
[Text box control example](#) 351  
[to element - form definition file](#) 67  
[toolbar element - form definition file](#) 114  
[toolbar element - form definition file extension](#) 139  
[Tracking changes](#) 410

## U

[unboundControls element - form definition file](#) 106  
[upgrade.xsl - form template file component](#) 25  
[Upgrade.XSL example](#) 367  
[Upgrade.XSL file - structure](#) 325  
[Upgrade.XSL file functions](#)  
[MSXSL Node-Set\(\)](#) 325  
[useHttpHandler element - form definition file](#) 84  
[useHttpHandlerExtension element - form definition file extension](#) 150  
[useQueryAdapter element - form definition file](#) 85  
[userName element - form definition file](#) 89  
[useScriptHandler element - form definition file](#) 85  
[useTransform element - form definition file](#) 122

## V

[value attribute - form view file](#) 317  
[Vendor-extensible fields \(section 1.7 23, section 1.7 23\)](#)  
[Versioning \(section 1.6 23, section 1.6 23\)](#)  
[view element - form definition file](#) 118  
[View syntax - form view file](#) 177  
[view.xsl - form template file component](#) 24  
[viewExtension element - form definition file extension](#) 159  
[views element - form definition file](#) 117  
[viewsExtension element - form definition file extension](#) 158

## W

[warning element - form definition file extension](#) 158  
[warnings element - form definition file extension](#) 157  
[webServiceAdapter element - form definition file](#) 56  
[webServiceAdapterExtension element - form definition file extension](#) 153  
[word element - form definition file extension](#) 163  
[words element - form definition file extension](#) 162  
[wss element - form definition file extension](#) 143

## X

[xctname attribute - form view file](#) 317  
[xdDate function extension - form view file](#) 319  
[xdDesignMode simple type - form definition file](#) 39  
[xdEmptyString simple type - form definition file](#) 39  
[xdEnabledDisabled simple type - form definition file](#) 36  
[xdEnvironment function extension - form view file](#) 320  
[xdErrorMessage simple type - form definition file](#) 39  
[xdExpressionLiteral simple type - form definition file](#) 37  
[xdFileName simple type - form definition file](#) 37  
[xdFormatting function extension - form view file](#) 321  
[xdHWSCaption simple type - form definition file](#) 42  
[xdHWSname simple type - form definition file](#) 42  
[xdImage function extension - form view file](#) 321  
[xdManualAuto simple type - form definition file](#) 36  
[xdMath function extension - form view file](#) 321  
[xdDocumentClass element - form definition file](#) 42  
[xdRoleName simple type - form definition file](#) 33  
[xdScriptLanguage simple type - form definition file](#) 38  
[xdSignatureRelationEnum simple type - form definition file](#) 41  
[xdSignedDataBlockMessage simple type - form definition file](#) 41  
[xdSignedDataBlockName simple type - form definition file](#) 40  
[xdSolutionVersion simple type - form definition file](#) 38  
[xdTitle simple type - form definition file](#) 31

[xdTrustLevel simple type - form definition file](#) 40  
[xdUser function extension - form view file](#) 322  
[xdUtil function extension - form view file](#) 323  
[xdViewName simple type - form definition file](#) 32  
[xdXDocument function extension - form view file](#) 323  
[xdYesNo simple type - form definition file](#) 33  
[XML schema](#) 371  
[InfoPath XSF XSD](#) 371  
[InfoPath XSF2 XSD](#) 399  
[the InfoPath XSF2 XSD](#) 399  
[XML schema file - control representation](#) 167  
[XML schema file controls](#)  
[button](#) 168  
[check box](#) 168  
[contact selector](#) 169  
[date picker](#) 169  
[drop-down list](#) 170  
[expression box](#) 170  
[file attachment](#) 171  
[hyperlink](#) 171  
[list box](#) 171  
[option button](#) 172  
[repeating section](#) 173  
[repeating table](#) 173  
[rich text box](#) 174  
[section/optional section](#) 174  
[table](#) 175  
[text box](#) 175  
[XML schema file example](#) 330  
[XML schema file structure](#) 167  
[xmlFileAdapter element - form definition file](#) 61  
[xmlFileAdapterExtension element - form definition file extension](#) 155  
[xmlToEdit attribute - form view file](#) 318  
[xmlToEdit element - form definition file](#) 119  
[xmlToEditExtension element - form definition file extension](#) 160  
[XSF extension example](#) 329  
[XSL function extensions example](#) 360  
[XSL root template element - form view file](#) 181  
[XSL root template style sheets - form view file](#) 183