[MS-IPFF2]:
InfoPath Form Template Format Version 2

Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

Preliminary Documentation. This Open Specification provides documentation for past and current releases and/or for the pre-release (beta) version of this technology. This Open Specification is final

documentation for past or current releases as specifically noted in the document, as applicable; it is preliminary documentation for the pre-release (beta) versions. Microsoft will release final documentation in connection with the commercial release of the updated or new version of this technology. As the documentation may change between this preliminary version and the final version of this technology, there are risks in relying on preliminary documentation. To the extent that you incur additional development obligations or any other costs as a result of relying on this preliminary documentation, you do so at your own risk.

Revision Summary

Date	Revision History	Revision Class	Comments
07/13/2009	0.1	Major	Initial Availability
08/28/2009	0.2	Editorial	Revised and edited the technical content
11/06/2009	0.3	Editorial	Revised and edited the technical content
02/19/2010	1.0	Minor	Updated the technical content
03/31/2010	1.01	Editorial	Revised and edited the technical content
04/30/2010	1.02	Editorial	Revised and edited the technical content
06/07/2010	1.03	Editorial	Revised and edited the technical content
06/29/2010	1.04	Editorial	Changed language and formatting in the technical content.
07/23/2010	1.05	Minor	Clarified the meaning of the technical content.
09/27/2010	1.05	No change	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	1.05	No change	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	1.05	No change	No changes to the meaning, language, or formatting of the technical content.
03/18/2011	1.05	No change	No changes to the meaning, language, or formatting of the technical content.
06/10/2011	1.05	No change	No changes to the meaning, language, or formatting of the technical content.
01/20/2012	1.6	Minor	Clarified the meaning of the technical content.
04/11/2012	1.6	No change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1.1 Glossary 14
1.2 References 16
1.2.1 Normative References 16
1.2.2 Informative References 18
1.3 Structure Overview (Synopsis) 18
1.3.1 Form Definition File (XSF) 19
1.3.2 XML Schema Files (XSD) 19
1.3.3 Form View Files (XSLT) 19
1.3.4 Print View Files (XSLT) 28
1.3.5 Submit Files (XML) 29
1.3.6 Template.XML File 29
1.3.7 Upgrade.XSL File 29
1.3.8 Resource Files 30
1.3.9 Unused Files 30
1.4 Relationship to Protocols and Other Structures 30
1.5 Applicability Statement 30
1.6 Versioning and Localization 30
1.7 Vendor-Extensible Fields 31

2 Structures 32
2.1 The InfoPath Form Template Format 32
2.1.1 manifest.xsf 32
2.1.2 Primary Schema File 32
2.1.3 View Files 32
2.1.4 Sample Data File 32
2.1.5 Template File 32
2.1.6 Secondary Schema File 32
2.1.7 Submit Data Files 33
2.1.8 View Upgrade File 33
2.1.9 Merge View File 33
2.1.10 Offline Secondary Schema Files 33
2.1.11 Business Object 33
2.1.12 Javascript Files 33
2.1.13 VBScript Files 33
2.1.14 Importer Errors File 33
2.1.15 Irm Template File 33
2.1.16 Resource Files 33
2.2 Form Definition File (XSF) Specification 34
2.2.1 Form Definition File (XSF) Specification 34
2.2.1.1 Form Definition File XSF Enumerations 40
2.2.1.1.1 xdTitle 40
2.2.1.1.2 xdViewName 42
2.2.1.1.3 xdRoleName 42
2.2.1.1.4 xdYesNo 43
2.2.1.1.5 xdEnabledDisabled 46
2.2.1.1.6 xdManualAuto 47
2.2.1.1.7 xdExpressionLiteral 47
2.2.1.1.8 xdFileName 48
2.2.1.1.9 xdScriptLanguage 49
2.2.1.1.10 xdSolutionVersion 49
2.2.1.1.11 xdEmptyString 50
2.2.1.1.12 xdErrorMessage 50
2.2.1.1.13 xdDesignMode 50

2.2.1.1.14 xdTrustLevel 51
2.2.1.1.15 xdSignedDataBlockName 51
2.2.1.1.16 xdSignedDataBlockMessage 52
2.2.1.1.17 xdSignatureRelationEnum 52
2.2.1.1.18 xdHWSname 52
2.2.1.1.19 xdHWSCaption 53
2.2.1.1.20 xdSignSignatureLineRuleEnum 53
2.2.1.2 Form Definition File XSF Elements 54
2.2.1.2.1 xDocumentClass 54
2.2.1.2.2 schemaErrorMessages 57
2.2.1.2.3 override 58
2.2.1.2.4 applicationParameters 58
2.2.1.2.5 solutionProperties 59
2.2.1.2.6 featureRestrictions 60
2.2.1.2.7 save 61
2.2.1.2.8 exportToWeb 61
2.2.1.2.9 exportToExcel 62
2.2.1.2.10 print 62
2.2.1.2.11 sendMail 63
2.2.1.2.12 autoRecovery 63
2.2.1.2.13 query 63
2.2.1.2.14 scripts 64
2.2.1.2.15 script 65
2.2.1.2.16 dataObjects 65
2.2.1.2.17 dataObject 66
2.2.1.2.18 query 67
2.2.1.2.19 adoAdapter 68
2.2.1.2.20 webServiceAdapter 69
2.2.1.2.21 hwsAdapter 70
2.2.1.2.22 operation 70
2.2.1.2.23 hwsOperation 71
2.2.1.2.24 input 72
2.2.1.2.25 partFragment 72
2.2.1.2.26 xmlFileAdapter 73
2.2.1.2.27 sharepointListAdapter 74
2.2.1.2.28 field 75
2.2.1.2.29 davAdapter 76
2.2.1.2.30 folderURL 77
2.2.1.2.31 fileName 77
2.2.1.2.32 emailAdapter 78
2.2.1.2.33 to 79
2.2.1.2.34 cc 80
2.2.1.2.35 bcc 81
2.2.1.2.36 subject 81
2.2.1.2.37 intro 82
2.2.1.2.38 attachmentFileName 82
2.2.1.2.39 submitToHostAdapter 83
2.2.1.2.40 dataAdapters 83
2.2.1.2.41 documentSchemas 84
2.2.1.2.42 documentSchema 85
2.2.1.2.43 customValidation 86
2.2.1.2.44 errorCondition 86
2.2.1.2.45 errorMessage 87

2.2.1.2.46 domEventHandlers 88
2.2.1.2.47 domEventHandler 89
2.2.1.2.48 importParameters 89
2.2.1.2.49 importSource 90
2.2.1.2.50 listProperties 91
2.2.1.2.51 fields 91
2.2.1.2.52 field 92
2.2.1.2.53 submit 94
2.2.1.2.54 submitAction 96
2.2.1.2.55 successMessage 97
2.2.1.2.56 errorMessage 97
2.2.1.2.57 useHttpHandler 97
2.2.1.2.58 useScriptHandler 98
2.2.1.2.59 useQueryAdapter 98
2.2.1.2.60 onLoad 98
2.2.1.2.61 save 99
2.2.1.2.62 roles 99
2.2.1.2.63 role 101
2.2.1.2.64 membership 101
2.2.1.2.65 getUserNameFromData 102
2.2.1.2.66 userName 102
2.2.1.2.67 group 103
2.2.1.2.68 hwsWorkflow 103
2.2.1.2.69 location 104
2.2.1.2.70 allowedActions 104
2.2.1.2.71 action 105
2.2.1.2.72 allowedTasks 105
2.2.1.2.73 task 106
2.2.1.2.74 fileNew 106
2.2.1.2.75 initialXmlDocument 107
2.2.1.2.76 customCategory 108
2.2.1.2.77 package 108
2.2.1.2.78 files 109
2.2.1.2.79 file 109
2.2.1.2.80 fileProperties 110
2.2.1.2.81 property 110
2.2.1.2.82 permissions 112
2.2.1.2.83 allowedControl 112
2.2.1.2.84 externalViews 113
2.2.1.2.85 externalView 114
2.2.1.2.86 attributeData 114
2.2.1.2.87 button 115
2.2.1.2.88 chooseFragment 117
2.2.1.2.89 editWith 117
2.2.1.2.90 unboundControls 121
2.2.1.2.91 button 121
2.2.1.2.92 editing 122
2.2.1.2.93 masterDetail 123
2.2.1.2.94 fragmentToInsert 123
2.2.1.2.95 mainpane 124
2.2.1.2.96 printSettings 124
2.2.1.2.97 header 127
2.2.1.2.98 footer 128

2.2.1.2.99 toolbar 128
2.2.1.2.100 menu 129
2.2.1.2.101 menuArea 130
2.2.1.2.102 taskpane 131
2.2.1.2.103 views 131
2.2.1.2.104 view 132
2.2.1.2.105 xmlToEdit 134
2.2.1.2.106 documentSignatures 135
2.2.1.2.107 signedDataBlock 136
2.2.1.2.108 message 137
2.2.1.2.109 documentVersionUpgrade 137
2.2.1.2.110 useTransform 137
2.2.1.2.111 extensions 138
2.2.1.2.112 extension 139
2.2.1.2.113 ruleSetAction 139
2.2.1.2.114 rule 140
2.2.1.2.115 submitAction 142
2.2.1.2.116 exitRuleSet 142
2.2.1.2.117 dialogBoxMessageAction 143
2.2.1.2.118 dialogBoxExpressionAction 143
2.2.1.2.119 switchViewAction 143
2.2.1.2.120 assignmentAction 144
2.2.1.2.121 changeAdapterProperty 144
2.2.1.2.122 queryAction 145
2.2.1.2.123 openNewDocumentAction 145
2.2.1.2.124 closeDocumentAction 146
2.2.1.2.125 ruleSet 146
2.2.1.2.126 ruleSets 147
2.2.1.2.127 calculations 147
2.2.1.2.128 calculatedField 148
2.2.1.2.129 bdcAdapter 149
2.2.1.2.130 grooveAdapter 150
2.2.1.2.131 field 151
2.2.1.2.132 sharepointListAdapterRW 152
2.2.1.2.133 field 156
2.2.1.2.134 webPartConnectionAction 157
2.2.1.2.135 signSignatureLineAction 158
2.2.2 Form Definition File (XSF2) Extension Specification 158
2.2.2.1 Form Definition File XSF2 Enumerations 161
2.2.2.1.1 serverCommandActionType 161
2.2.2.1.2 emailAttachmentType 161
2.2.2.1.3 compatibilityModesType 162
2.2.2.1.4 solutionType 162
2.2.2.1.5 formDescriptionType 162
2.2.2.1.6 formLocaleType 163
2.2.2.1.7 managedCodeType 163
2.2.2.2 Form Definition File XSF2 Elements 163
2.2.2.2.1 solutionDefinition 164
2.2.2.2.2 server 165
2.2.2.2.3 toolbar 166
2.2.2.2.4 commands 167
2.2.2.2.5 command 167
2.2.2.2.6 solutionPropertiesExtension 168

2.2.2.2.7 install 170
2.2.2.2.8 wss 170
2.2.2.2.9 contentType 171
2.2.2.2.10 contentTypeTemplate 171
2.2.2.2.11 share 172
2.2.2.2.12 mail 172
2.2.2.2.13 admin 173
2.2.2.2.14 mergedPrintView 173
2.2.2.2.15 includedViews 174
2.2.2.2.16 includedView 175
2.2.2.2.17 offline 175
2.2.2.2.18 listPropertiesExtension 176
2.2.2.2.19 fieldsExtension 176
2.2.2.2.20 fieldExtension 177
2.2.2.2.21 dataConnections 177
2.2.2.2.22 useHttpHandlerExtension 178
2.2.2.2.23 connectoid 179
2.2.2.2.24 davAdapterExtension 180
2.2.2.2.25 adoAdapterExtension 181
2.2.2.2.26 webServiceAdapterExtension 181
2.2.2.2.27 relativeQuery 182
2.2.2.2.28 emailAdapterExtension 183
2.2.2.2.29 xmlFileAdapterExtension 183
2.2.2.2.30 sharepointListAdapterExtension 184
2.2.2.2.31 sendByMail 185
2.2.2.2.32 warnings 186
2.2.2.2.33 warning 186
2.2.2.2.34 viewsExtension 187
2.2.2.2.35 viewExtension 187
2.2.2.2.36 xmlToEditExtension 188
2.2.2.2.37 preview 189
2.2.2.2.38 autoUpdatePrompt 189
2.2.2.2.39 inputScopes 190
2.2.2.2.40 inputScope 190
2.2.2.2.41 words 191
2.2.2.2.42 word 192
2.2.2.2.43 managedCode 192
2.2.2.2.44 submit 193
2.2.2.2.45 submitAction 194
2.2.2.2.46 successMessage 194
2.2.2.2.47 errorMessage 194
2.2.2.2.48 featureRestrictionsExtension 195
2.2.2.2.49 exportToPDForXPS 195
2.2.2.2.50 list 196
2.2.2.2.51 entity 196
2.2.2.2.52 workflowInitAssoc 197
2.2.2.2.53 groove 197
2.2.2.2.54 sharepointListAdapterRWExtension 197
2.2.3 Form Definition File (XSF3) Extension Specification 198
2.2.3.1 Form Definition File XSF3 Enumerations 199
2.2.3.1.1 xdParameterType 199
2.2.3.1.2 xdModeType 200
2.2.3.1.3 xdLineStamp 201

2.2.3.2 Form Definition File XSF3 Elements 201
2.2.3.2.1 solutionDefinition 201
2.2.3.2.2 baseUrl 202
2.2.3.2.3 webPartProperties 203
2.2.3.2.4 webPartFields 203
2.2.3.2.5 webPartField 204
2.2.3.2.6 solutionPropertiesExtension2009 204
2.2.3.2.7 solutionMode 205
2.2.3.2.8 viewsExtension 206
2.2.3.2.9 viewExtension 206
2.2.3.2.10 xmlToEditExtension 207
2.2.3.2.11 signatureLines 207
2.2.3.2.12 signatureLine 208
2.2.3.2.13 confirmationMessage 209
2.2.3.2.14 suggestedSignerName 209
2.2.3.2.15 suggestedSignerTitle 210
2.2.3.2.16 suggestedSignerEmailAddress 210
2.2.3.2.17 customValidation 211
2.2.3.2.18 errorBlank 211
2.3 XML Schema Files (XSD) Specification 212
2.3.1 Control Representation 212
2.3.1.1 Button Control 214
2.3.1.2 Check Box Control 214
2.3.1.3 Contact Selector Control 215
2.3.1.4 Date Picker Control 215
2.3.1.5 Drop-Down List Control 216
2.3.1.6 Expression Box Control 216
2.3.1.7 File Attachment Control 216
2.3.1.8 Hyperlink Control 217
2.3.1.9 List Box Control 217
2.3.1.10 Option Button Control 218
2.3.1.11 Repeating Section Control 218
2.3.1.12 Repeating Table Control 219
2.3.1.13 Rich Text Box Control 220
2.3.1.14 Section Control and Optional Section Control 220
2.3.1.15 Table Control 221
2.3.1.16 Text Box Control 221
2.3.2 Controls Introduced in Version 2 of the Structure Specification 222
2.3.2.1 Choice Group Control and Choice Section Control 222
2.3.2.2 Combo Box Control 223
2.3.2.3 Date and Time Picker Control 224
2.3.2.4 External Item Picker Control 224
2.3.2.5 Embedded Picture Control 225
2.3.2.6 Hyperlink Input Control 225
2.3.2.7 List Controls (Bulleted List Control, Numbered List Control and Plain List Control) 227
2.3.2.8 Linked Picture Control 228
2.3.2.9 Multiple-Selection List Box Control 230
2.3.2.10 Picture Button Control 230
2.3.2.11 SharePoint File Attachment Control 231
2.4 Form View Files (XSLT) Specification 231
2.4.1 View Representation 231
2.4.1.1 View Syntax 233

2.4.1.2 XSL Root Template 239
2.4.1.3 XSL Root Template Style Sheets 241
2.4.1.4 Control Data Formatting 272
2.4.1.5 Button Control 274
2.4.1.6 Check Box Control 280
2.4.1.7 Contact Selector Control 282
2.4.1.8 Date Picker Control 285
2.4.1.9 Drop-Down List Control 291
2.4.1.10 Expression Box Control 298
2.4.1.11 File Attachment Control 301
2.4.1.12 Hyperlink Control 301
2.4.1.13 List Box Control 303
2.4.1.14 Option Button Control 306
2.4.1.15 Repeating Section Control 308
2.4.1.16 Repeating Table Control 310
2.4.1.17 Rich Text Box Control 313
2.4.1.18 Section Control and Optional Section Control 315
2.4.1.19 Table Control 322
2.4.1.20 Text Box Control 323
2.4.1.21 View Representation for Controls Introduced in Version 2 of the Structure Specification 332
2.4.1.21.1 Choice Group Control and Choice Section Control 332
2.4.1.21.2 Combo Box Control 334
2.4.1.21.3 Date and Time Picker Control 344
2.4.1.21.4 External Item Picker Control 345
2.4.1.21.5 Embedded Picture Control 348
2.4.1.21.6 Hyperlink Input Control 350
2.4.1.21.7 List Controls (Bulleted List Control, Numbered List Control and Plain List Control) 358
2.4.1.21.8 Linked Picture Control 360
2.4.1.21.9 Multiple-Selection List Box Control 363
2.4.1.21.10 Picture Button Control 372
2.4.1.21.11 SharePoint File Attachment Control 375
2.4.1.22 Ignored Controls 376
2.4.1.23 Invalid Controls 376
2.4.1.24 Invalid Constructs 377
2.4.2 Control-Specific Attributes 377
2.4.2.1 action 380
2.4.2.2 allownonmatching 380
2.4.2.3 autoAdvance 381
2.4.2.4 auxDom 381
2.4.2.5 backgroundPicture 381
2.4.2.6 binding 381
2.4.2.7 bindingProperty 384
2.4.2.8 bindingType 384
2.4.2.9 boundProp 384
2.4.2.10 CtrlId 386
2.4.2.11 datafmt 387
2.4.2.12 disableEditing 393
2.4.2.13 enabledProperty 393
2.4.2.14 enabledValue 394
2.4.2.15 ghosted 394
2.4.2.16 ictID 394

2.4.2.17 ictVersion 394
2.4.2.18 inline 394
2.4.2.19 innerCtrl 395
2.4.2.20 inputscope 395
2.4.2.21 inputScopeId 395
2.4.2.22 layoutText 395
2.4.2.23 linkedToMaster 396
2.4.2.24 masterID 396
2.4.2.25 masterName 396
2.4.2.26 num 396
2.4.2.27 offValue 397
2.4.2.28 onValue 397
2.4.2.29 postbackModel 397
2.4.2.30 ref 399
2.4.2.31 SignatureBlock 399
2.4.2.32 SignedSectionDisplaySignatures 399
2.4.2.33 SignedSectionName 400
2.4.2.34 value 400
2.4.2.35 xctname 400
2.4.2.36 xmlToEdit 402
2.4.2.37 Control-Specific Attributes Introduced in Version 2 of the Structure Specification 403
2.4.2.37.1 AllowMultiple 403
2.4.2.37.2 binding_secondary 403
2.4.2.37.3 boundPropSecondary 403
2.4.2.37.4 datafmt2 404
2.4.2.37.5 HideInPrintView 405
2.4.2.37.6 HoverSrc 405
2.4.2.37.7 SearchPeopleOnly 405
2.4.2.37.8 server 406
2.4.2.37.9 SharePointGroup 406
2.4.2.37.10 widgetIndex 406
2.4.2.37.11 caption_ 407
2.4.3 XSL Function Extensions 407
2.4.3.1 msxsl 408
2.4.3.1.1 string-compare 408
2.4.3.2 xdDate 408
2.4.3.2.1 AddDays 408
2.4.3.2.2 AddSeconds 409
2.4.3.2.3 Now 409
2.4.3.2.4 Today 409
2.4.3.3 xdEnvironment 409
2.4.3.3.1 IsBrowser 409
2.4.3.3.2 IsMobile 410
2.4.3.4 xdFormatting 410
2.4.3.5 xdImage 410
2.4.3.6 xdMath 410
2.4.3.6.1 Avg 410
2.4.3.6.2 Eval 410
2.4.3.6.3 Max 411
2.4.3.6.4 Min 411
2.4.3.6.5 Nz 411
2.4.3.7 xdUser 411

2.4.3.7.1 get-UserName 412
2.4.3.8 xdUtil 412
2.4.3.8.1 Match 412
2.4.3.9 xdXDocument 412
2.4.3.9.1 get-dom 412
2.4.3.9.2 getDOM 412
2.4.3.9.3 getnamednodeproperty 412
2.4.3.10 xdServerInfo 413
2.4.3.10.1 get-SharePointListUrl 413
2.4.3.10.2 get-SharePointSiteUrl 413
2.4.3.10.3 get-SharePointServerRootUrl 413
2.4.3.10.4 get-SharePointSiteCollectionUrl 413
2.4.3.11 ipApp 413
2.5 Print View Files (XSLT) Specification 414
2.6 Submit Files (XML) Specification 414
2.6.1 myFields 414
2.6.2 dataFields 414
2.7 Template.XML Specification 415
2.8 Upgrade.XSL Specification 415
2.8.1 MSXSL:Node-Set() 415

3 Structure Examples 416
3.1 The InfoPath Form Template Format 416
3.1.1 Simple Form Template 416
3.1.2 Complex Form Template 417
3.2 Form Definition File (XSF) Examples 417
3.2.1 Form Definition File (XSF) Examples: Browser-Compatible Form 417
3.2.2 Form Definition File (XSF) Examples: List Form 421
3.3 XML Schema Files (XSD) Examples 427
3.4 Form View Files (XSL) Examples 428
3.4.1 Control Representation 428
3.4.1.1 Button Control 428
3.4.1.2 Check Box Control 430
3.4.1.3 Contact Selector Control 430
3.4.1.4 Date Picker Control 431
3.4.1.5 Drop-Down List Control 433
3.4.1.6 Expression Box Control 437
3.4.1.7 File Attachment Control 437
3.4.1.8 Hyperlink Control 438
3.4.1.9 List Box Control 438
3.4.1.10 Option Button Control 440
3.4.1.11 Repeating Section Control 442
3.4.1.12 Repeating Table Control 442
3.4.1.13 Rich Text Box Control 446
3.4.1.14 Section Control and Optional Section Control 446
3.4.1.15 Table Control 449
3.4.1.16 Text Box Control 450
3.4.2 Control Representation for Controls Introduced in Version 2 of the Structure Specification 452
3.4.2.1 Choice Group Control and Choice Section Control 452
3.4.2.2 Combo Box Control 453
3.4.2.3 Date and Time Picker Control 459
3.4.2.4 External Item Picker Control 460

3.4.2.5 Embedded Picture Control 461
3.4.2.6 Hyperlink Input Control 462
3.4.2.7 List Controls (Bulleted List Control, Numbered List Control and Plain List Control) 464
3.4.2.8 Linked Picture Control 466
3.4.2.9 Multiple-Selection List Box Control 467
3.4.2.10 Picture Button Control 473
3.4.2.11 SharePoint File Attachment Control 475
3.4.3 Control-Specific Attributes 476
3.4.4 XSL Function Extensions 482
3.5 Print View Files (XSLT) Examples 486
3.6 Submit Files (XML) Examples 488
3.7 Template.XML Examples 489
3.8 Upgrade.XSL Examples 489
3.8.1 Upgrade.XSL Example 489
3.8.2 MSXSL:Node-Set() Example 491

4 Security Considerations 493

5 Appendix A: Full XML Schemas 494
5.1 The InfoPath XSF XSD file 494
5.2 The InfoPath XSF2 XSD file 525
5.3 The InfoPath XSF3 XSD file 535
5.4 The Built-In ActiveX Controls XSD file 539

6 Appendix B: Product Behavior 541

7 Change Tracking 542

8 Index 543

1 Introduction

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

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

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

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

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

Sections 1.7 and 2 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. All other sections and examples in this specification are informative.

1.1 Glossary

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

ASCII
Augmented Backus-Naur Form (ABNF)
GUID
Hypertext Transfer Protocol (HTTP)
language code identifier (LCID)
SHA-1 hash
Unicode
Universal Naming Convention (UNC)
user agent
UTF-16
XML

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

absolute URL
ActiveX control
ActiveX Data Objects (ADO)
ADO
ASP.NET control
assembly
attachment
background color
blind carbon copy (bcc) recipient
Boolean
border formatting
browser-compatible form template
built-in control
business object
cabinet (.cab) file
CAML
carbon copy (cc) recipient
column
conditional formatting
connection string
content type
content type identifier
control
data adapter
data connection
data connection library
data source
digital signature
empty string
Extended Backus-Naur Form (EBNF)
external content type
external list
field
field definition
form
form definition (.xsf) file

form file
form library
form security level
form server
form template
form template (.xsn) file
form view
group
hash
HTTP method
Hypertext Markup Language (HTML)
Identifier
Information Rights Management (IRM)
left-to-right
list
list identifier
list item
list view
LobSystemInstance
locale
lookup field
main data connection
main data source
MIME type
mobile device
named property
offline
postback
print view
query
red-green-blue (RGB)
repeating group
Representational State Transfer (REST)
right-to-left
row
rule
secondary data connection
secondary data source
server-relative URL
site
site collection
SOAP action
SOAP message
SpecificFinder
SQL statement
Structured Query Language (SQL)
submit
symbol file
toolbar
Uniform Resource Locator (URL)
Uniform Resource Name (URN)
Universal Data Connection (.udc, .udcx) file
user name

Web Distributed Authoring and Versioning Protocol (WebDAV)
Web Part
Web Part connection
Web service
Web Services Description Language (WSDL)
workflow
XML document
XML element
XML fragment
XML node
XML schema
XML schema document
XPath expression
XSL
XSL Transformation (XSLT)

The following terms are specific to this document:

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

1.2 References

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the documents, which are updated frequently. References to other documents include a publishing year when one is available.

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[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/goglobal/bb896001.aspx

[MSDN-CALID] Microsoft Corporation, "Calendar Identifiers", http://msdn.microsoft.com/en-us/library/dd317732.aspx

[MS-IPFF] Microsoft Corporation, "InfoPath Form Template Format".

[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-WSSCAML] Microsoft Corporation, "Collaborative Application Markup Language (CAML) Structure Specification".

[MS-WSSFO2] Microsoft Corporation, "Windows SharePoint Services (WSS): File Operations Database Communications Version 2 Protocol Specification".

[MS-WSSTS] Microsoft Corporation, "Windows SharePoint Services Technical 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/

[XML10] World Wide Web Consortium, "Extensible Markup Language (XML) 1.0 (Third Edition)", February 2004, http://www.w3.org/TR/REC-xml

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

[XMLSCHEMA1] Thompson, H.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)

A **form template (.xsn) file** is a **cabinet (.cab) file**, as described 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 (1)** that has been filled out.

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

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

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

The following sections describe the files that can be found in a form template (.xsn) file and how they are used to render a form (1).

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

1.3.1 Form Definition File (XSF)

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

- the list of files that comprise the form template (.xsn) file and the relationships between them.
- properties of the form template, such as deployment information and user interface customizations.
- properties of certain **controls** within the form template. For example, a control can have associated **rules (1)** that perform tasks automatically based on events and values.

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

1.3.2 XML Schema Files (XSD)

There are one or more of the following **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 described in section 2.3. Each XSD file is described in section 2.2.1.2.41. Examples are provided in section 3.3.

1.3.3 Form View Files (XSLT)

A **form view** is a specific visualization of a form file in a Web browser. It specifies what controls are used to represent the **fields (3)** in the form (1) and how they are laid out. It also provides information to the form server about the editing behavior for each control. For example, it tells the server 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 of the following 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 (1) as it is edited.

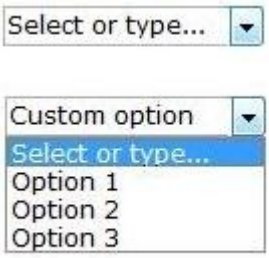
The default XSLT file is specified in the form definition (.xsf) file.

The contents of an XSLT file are used to transform the data in a form file into **HTML**, so that the form (1) can be rendered and edited in a Web browser. The resulting HTML visualization consists of

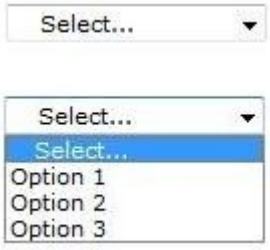
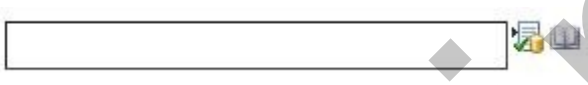
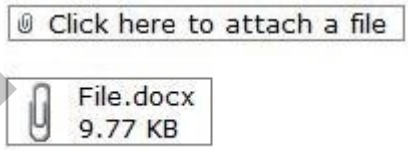
HTML used to lay out and represent the controls and informative text. The HTML for a control also contains properties specifying its behavior.

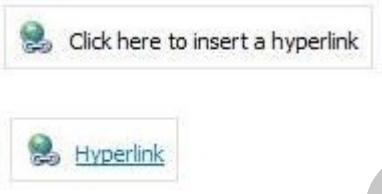
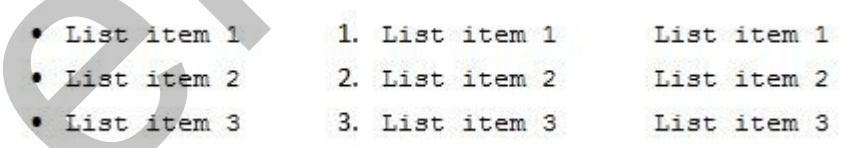
The following table lists the controls that can be used in a form (1). Controls are usually tied to specific fields (3) in the form file and are used to edit such fields (3). Some controls, as noted in the table, are not the direct representation of any particular fields (3).

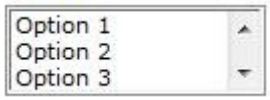
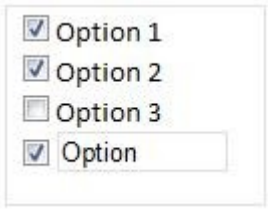
Control	Description and sample visual representation
Button	A control used to run an action. It is not tied to a field (3). The following figure illustrates a typical representation of a button control.  **Figure 2: A button control** The visualization and properties of the control are described in section 2.4.1.5. The relationship to the XML schema of the form template is described in section 2.3.1.1. Behaviors affecting the control are described in sections 2.2.1.2.126 and 2.2.1.2.43.
Check Box	A control that allows users to set yes/no or true/false values by adding or removing a check mark from a small square box. The following figure illustrates a typical representation of a check box control.  **Figure 3: A set of check box controls** The visualization and properties of the control are described in section 2.4.1.6. The relationship to the XML schema of the form template is described in section 2.3.1.2. Behaviors affecting the control are described in sections 2.2.1.2.126 and 2.2.1.2.43.
Choice Group, Choice Section	A choice group control allows users to choose between two or more choice sections to include in the form (1), where each choice section can contain one or more controls. When filling out a form (1), users can replace the default choice section with a different choice section. The following figure illustrates a typical representation of a choice group control and its default choice section control.

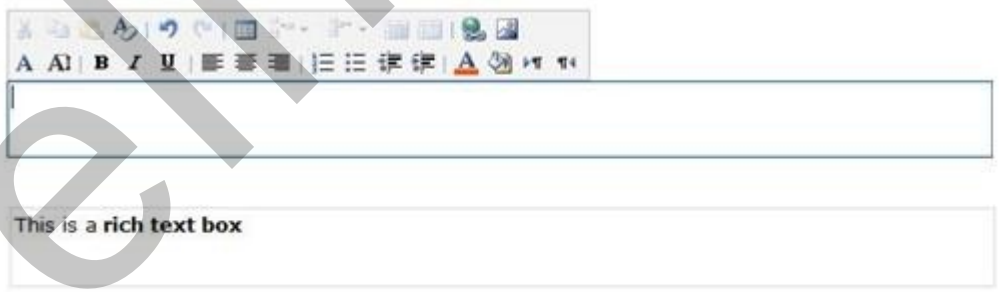
Control	Description and sample visual representation
	 **Figure 4: A choice group control** The visualization and properties of the control are described in section 2.4.1.21.1. The relationship to the XML schema of the form template is described in section 2.3.2.1. Behaviors affecting the control are described in sections 2.2.1.2.126 and 2.2.1.2.43.
Combo Box	A control that allows users to either type an entry or make a selection from a list of options. The following figure illustrates a typical representation of a combo box control.  **Figure 5: A combo box control** The visualization and properties of the control are described in section 2.4.1.21.2. The relationship to the XML schema of the form template is described in section 2.3.2.2. Behaviors affecting the control are described in sections 2.2.1.2.126 and 2.2.1.2.43.
Contact Selector	A control that allows users to select one or more contacts from a protocol server user list. The following figure illustrates a typical representation of a contact selector control.  **Figure 6: A contact selector control** The visualization and properties of the control are described in section 2.4.1.7. The relationship to the XML schema of the form template is described in section 2.3.1.3. Behaviors affecting the control are described in sections 2.2.1.2.126 and 2.2.1.2.43.
Date Picker	A control that contains a box where users can type dates and a calendar button that allows users to select a date. The following figure illustrates a typical representation of a date picker control.

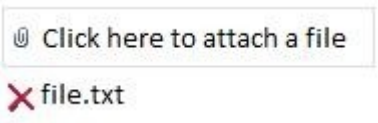
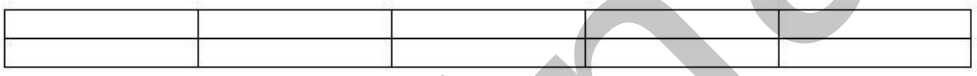
Control	Description and sample visual representation
	 **Figure 7: A date picker control** The visualization and properties of the control are described in section 2.4.1.8. The relationship to the XML schema of the form template is described in section 2.3.1.4. Behaviors affecting the control are described in sections 2.2.1.2.126 and 2.2.1.2.43.
Date and Time Picker	A control that allows users to type a date and time or select a date from a calendar display. The following figure illustrates a typical representation of a date and time picker control.  **Figure 8: A date and time picker control** The visualization and properties of the control are described in section 2.4.1.21.3. The relationship to the XML schema of the form template is described in section 2.3.2.3. Behaviors affecting the control are described in sections 2.2.1.2.126 and 2.2.1.2.43.
Drop-Down List Control	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 control.

Control	Description and sample visual representation
	 **Figure 9: A drop-down list control** The visualization and properties of the control are described in section 2.4.1.9. The relationship to the XML schema of the form template is described in section 2.3.1.5. Behaviors affecting the control are described in sections 2.2.1.2.126 and 2.2.1.2.43.
External Item Picker	A control that allows users to select or enter an instance of an external content type from a business data catalog on the protocol server. The following figure illustrates a typical representation of an external item picker control.  **Figure 10: An external item picker control** The visualization and properties of the control are described in section 2.4.1.21.4. The relationship to the XML schema of the form template is described in section 2.3.2.4. Behaviors affecting the control are described in sections 2.2.1.2.126 and 2.2.1.2.43.
Expression Box	A read-only text control used to display information. The following figure illustrates a typical representation of an expression box control.  **Figure 11: An expression box control** The visualization and properties of the control are described in section 2.4.1.10. The relationship to the XML schema of the form template is described in section 2.3.1.6. Behaviors affecting the control are described in sections 2.2.1.2.126 and 2.2.1.2.43.
File Attachment	A control that allows users to attach files to their form files. Each file attachment control permits one file to be attached. The following figure illustrates a typical representation of a file attachment control. 

Control	Description and sample visual representation
	Figure 12: A file attachment control The visualization and properties of the control are described in section 2.4.1.11. The relationship to the XML schema of the form template is described in section 2.3.1.7. Behaviors affecting the control are described in sections 2.2.1.2.126 and 2.2.1.2.43.
Hyperlink	A control that can be used to link to a **Uniform Resource Locator (URL)**. The following figure illustrates a typical representation of a hyperlink.  **Figure 13: A hyperlink control** The visualization and properties of the control are described in section 2.4.1.12. The relationship to the XML schema of the form template is described in section 2.3.1.8. Behaviors affecting the control are described in sections 2.2.1.2.126 and 2.2.1.2.43.
Hyperlink Input	A control that allows users to insert a hyperlink in a form (1) by entering a URL and display text. The following figure illustrates a typical representation hyperlink input control.  **Figure 14: A hyperlink input control** The visualization and properties of the control are described in section 2.4.1.21.6. The relationship to the XML schema of the form template is described in section 2.3.2.6. Behaviors affecting the control are described in sections 2.2.1.2.126 and 2.2.1.2.43.
List (Bulleted, Numbered, Plain)	A control that allows users to enter text in a bulleted, numbered, or plain list (1). The following figure illustrates a typical representation of a bulleted, numbered, and plain list control.  **Figure 15: A bulleted, numbered, and plain list control (left, center and right respectively)** The visualization and properties of the control are described in section 2.4.1.21.7. The relationship to the XML schema of the form template is described in section 2.3.2.7. Behaviors affecting the control are described in sections 2.2.1.2.126 and 2.2.1.2.43.

Control	Description and sample visual representation
List Box	A control that presents users with a list of choices in a box from which users select the appropriate item. The following figure illustrates a typical representation of a list box control.  **Figure 16: A list box control** The visualization and properties of the control are described in section 2.4.1.13. The relationship to the XML schema of the form template is described in section 2.3.1.9. Behaviors affecting the control are described in sections 2.2.1.2.126 and 2.2.1.2.43.
Multiple-Selection List Box	A control that allows users to select multiple values from a list of options. The following figure illustrates a typical representation of a multiple-selection list box control.  **Figure 17: A multiple-selection list box control** The visualization and properties of the control are described in section 2.4.1.21.9. The relationship to the XML schema of the form template is described in section 2.3.2.9. Behaviors affecting the control are described in sections 2.2.1.2.126 and 2.2.1.2.43.
Option Button	A control that lets users select from a set of mutually exclusive choices. When one option button in a **group (1)** is selected, the other option buttons are cleared. The following figure illustrates a typical representation of an option button control.  **Figure 18: A set of option button controls** The visualization and properties of the control are described in section 2.4.1.14. The relationship to the XML schema of the form template is described in section 2.3.1.10. Behaviors

Control	Description and sample visual representation
	affecting the control are described in sections 2.2.1.2.126 and 2.2.1.2.43.
Picture (Embedded and Linked)	A control that allows users to insert a picture in the form (1). The following figure illustrates a typical representation of an embedded and linked picture control.  **Figure 19: A picture control** The visualization and properties of the control are described in sections 2.4.1.21.5 and 2.4.1.21.8. The relationship to the XML schema of the form template is described in sections 2.3.2.5 and 2.3.2.8. Behaviors affecting the control are described in sections 2.2.1.2.126 and 2.2.1.2.43.
Picture Button	A control that allows users to run an action. Users can replace the button graphic with a custom picture that is displayed on the form (1). It is not tied to a field (3). The following figure illustrates a typical representation of a picture button control.  **Figure 20: A picture button control; the right image shows it in the hover state** The visualization and properties of the control are described in section 2.4.1.21.10. The relationship to the XML schema of the form template is described in section 2.3.2.10. Behaviors affecting the control are described in sections 2.2.1.2.126 and 2.2.1.2.43.
Rich Text Box	A text input control that can contain formatted text, including bold and italic text, and a variety of fonts, font sizes, and font colors. The following figure illustrates a typical representation of a rich text box control.  **Figure 21: A rich text box control; the top image shows it in the editing state** The visualization and properties of the control are described in section 2.4.1.17. The relationship to the XML schema of the form template is described in section 2.3.1.13. Behaviors

Control	Description and sample visual representation
	affecting the control are described in sections 2.2.1.2.126 and 2.2.1.2.43.
SharePoint File Attachment	A control that allows users to attach multiple files to the form (1). The following figure illustrates a typical representation of a SharePoint file attachment control.  **Figure 22: A SharePoint file attachment control** The visualization and properties of the control are described in section 2.4.1.21.11. The relationship to the XML schema of the form template is described in section 2.3.2.11. Behaviors affecting the control are described in sections 2.2.1.2.126 and 2.2.1.2.43.
Table	A table used to lay out the form (1). It is not tied to a field (3). The following figure illustrates a typical representation of a table control.  **Figure 23: A table** The visualization and properties of the control are described in section 2.4.1.19. The relationship to the XML schema of the form template is described in section 2.3.1.15. Behaviors affecting the control are described in sections 2.2.1.2.126 and 2.2.1.2.43.
Text Box	A text input control that can contain any unformatted text. Text box controls cannot contain formatted text. The following figure illustrates a typical representation of a text box control.  **Figure 24: A text box control** The visualization and properties of the control are described in section 2.4.1.20. The relationship to the XML schema of the form template is described in section 2.3.1.16. Behaviors affecting the control are described in sections 2.2.1.2.126 and 2.2.1.2.43.
Repeating Table	A control that displays repeating information in a tabular structure. Each item appears in a new row in the repeating table control. When filling out a form (1), users can add or delete rows in a repeating table control as necessary. Repeating table controls can contain other controls. The following figure illustrates a typical representation of a repeating table control.

Control	Description and sample visual representation
	 **Figure 25: A repeating table control with three text box controls per row** The visualization and properties of the control are described in section 2.4.1.16. The relationship to the XML schema of the form template is described in section 2.3.1.12. Behaviors affecting the control are described in sections 2.2.1.2.126 and 2.2.1.2.43.
Section, Optional Section	A control that is a container for other controls. An optional section control is the same as the section control, but does not need to be initially displayed in the form (1). The following figure illustrates a typical representation of a section control.  **Figure 26: A section control containing two text box controls** The visualization and properties of the control are described in section 2.4.1.18. The relationship to the XML schema of the form template is described in section 2.3.1.14. Behaviors affecting the control are described in sections 2.2.1.2.126 and 2.2.1.2.43.
Repeating Section	A control that is a container for other controls and is used to display repeating information. When filling out the form (1) that includes a repeating section control, users can add additional occurrences of the repeating section control. The following figure illustrates a typical representation of a repeating section control.  **Figure 27: A repeating section control** The visualization and properties of the control are described in section 2.4.1.15. The relationship to the XML schema of the form template is described in section 2.3.1.11. Behaviors affecting the control are described in sections 2.2.1.2.126 and 2.2.1.2.43.

The structure of an XSLT file that defines a form view is described in section 2.4. The role of each XSLT file is described in section 2.2.1.2.103. Examples are provided in section 3.4.

1.3.4 Print View Files (XSLT)

Print views are visualizations of the form (1) used for printing. A print view is defined by an XSLT file that is optimized for printing the data instead of visualizing it in a Web browser. For example, a print view could use dark text on a white background, or suppress borders on the controls.

A print view is always associated with a form view so that when the form server is displaying the form view and the user chooses to print the form (1), the print view is sent to the printer.

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

The structure of an XSLT file that defines a print view is described in section 2.5. The role of each XSLT file is described in section 2.2.1.2.103. Examples are provided in section 3.5.

1.3.5 Submit Files (XML)

Submit files are **XML** files, as described in [W3C-XML], that specify the information used to submit form data to a **Web service**. This information is only needed when the form (1) 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 a 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 (3) and parameters to the Web service methods.

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

1.3.6 Template.XML File

The template.xml file is a form file based on the form template that contains it. It is used to store and load initial values of the fields (3) when creating a new form file based on the form template. For example, it is used when 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 described in section 2.7. Examples are provided in section 3.7.

1.3.7 Upgrade.XSL File

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

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

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

1. Copies fields (3) from the form file to the upgraded one.
2. Removes fields (3) that are no longer used.
3. Adds new fields (3) 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 described in section 2.8. Examples are provided in section 3.8.

1.3.8 Resource Files

The following files are present for certain form templates:

- **Business object** files
- Image files
- File **attachments**

The list of files in the form template is described in sections 2.1 and 2.2.1.2.78. Examples are provided in section 3.1.

1.3.9 Unused Files

The following files are never used by a form server to render or edit a form (1), but are often present for use by the InfoPath client application:

- 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 XSLT, as described in [W3C-XSLT], that can be used to combine multiple form files into a single form file.
- the sampledata.xml file is an XML file, as described in [W3C-XML] containing sample data for the form (1).
- the *script*.js and *script*.vbs files are used for scripting events.
- the *irm_template* file is used for **Information Rights Management (IRM)**.
- the importerrors.xml file contains errors resulting from importing files.

The list of files in the form template is described in sections 2.1 and 2.2.1.2.78.

1.4 Relationship to Protocols and Other Structures

The InfoPath form template format is an extension of the cabinet (.cab) file format described in [MC-MCF].

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

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

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

1.5 Applicability Statement

This structure is used by a form server to render and edit forms (1) based on a form template. The form (1) 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

This protocol defines vendor-extensible fields (3), as specified by the **solutionDefinition** element in section 2.2.2.2.1.

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 (1). The following sections list files that could appear in a form template (.xsn) file.

The name of the form template (.xsn) file MUST end with the .xsn file extension and MUST contain **Unicode UTF-16** characters, as specified in [RFC2781]. The form template (.xsn) file name MUST NOT contain the following characters: " # % & * : < > ? { | } ~. The form template (.xsn) file name MUST NOT contain characters that 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 2.2.1.2.78.

2.1.2 Primary Schema File

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

2.1.3 View Files

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

2.1.4 Sample Data File

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

2.1.5 Template File

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

2.1.6 Secondary Schema File

A form template (.xsn) file MUST contain one or more secondary schema files if there are data connections (1) within the form template. A secondary schema file MUST conform to the naming conventions for files stored in a cabinet (.cab) file. See section 2.3 and section 2.2.2.2.21.

2.1.7 Submit Data Files

A form template (.xsn) file MAY contain one or more submit data files if there are data connections (1) within the form template. Submit data files MUST conform to the naming conventions for files stored in a cabinet (.cab) file. See section 2.6.

2.1.8 View Upgrade File

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

2.1.9 Merge View File

A form template (.xsn) file MAY contain a merge view file. Merge view files MUST be ignored by the form server.

2.1.10 Offline Secondary Schema Files

A form template (.xsn) file MAY contain one or more files that are associated with a secondary schema file. Offline secondary schema 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 2.2.2.2.43.

2.1.12 Javascript Files

A form template (.xsn) file MAY contain one or more JavaScript files. See section 2.2.1.2.14.

2.1.13 VBScript Files

A form template (.xsn) file MAY contain one or more VBScript files. See section 2.2.1.2.14.

2.1.14 Importer Errors File

A form template (.xsn) file MAY contain an importer errors file. If present, the importer error file MUST be ignored.

2.1.15 Irm Template File

A form template (.xsn) file MUST NOT contain an irm_template file.

2.1.16 Resource Files

A form template (.xsn) file MAY contain other files specified in the form definition (.xsf) file. These files MUST conform to the naming conventions for files stored in a cabinet (.cab) file. Resource files MAY include files that would cause the form server to reject the form template (.xsn) file or return an error in other circumstances. See section 2.2.1.2.78.

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 sections listed in the following table.

Section	Summary
2.2.1 Form Definition File (XSF) Specification	This section specifies types and elements in the **XSF namespace**: http://schemas.microsoft.com/office/infopath/2003/solutionDefinition The **xDocumentClass** element (section 2.2.1.2.1) is the root element of the form definition (.xsf) file, contains the core set of properties, content, and files of the form template. The **extensions** element (section 2.2.1.2.111) contains extensions to the properties, content, and files of the form template.
2.2.2 Form Definition File (XSF2) Extension Specification	This section specifies types and elements in the **XSF2 namespace**: http://schemas.microsoft.com/office/infopath/2006/solutionDefinition/extensions The **XSF2:solutionDefinition** element (section 2.2.2.2.1) acts as the container for multiple extensions to the properties and content of the form template. This element is contained by the **extensions** element (section 2.2.1.2.111) of the **xDocumentClass** element (section 2.2.1.2.1).
2.2.3 Form Definition File (XSF3) Extension Specification	This section specifies types and elements in the **XSF3 namespace:** http://schemas.microsoft.com/office/infopath/2009/solutionDefinition/extensions The **XSF3:solutionDefinition** and **solutionPropertiesExtension2009** elements (section 2.2.3.2.1) act as the containers for multiple extensions to the properties and content of the form template. These elements are contained by the **extensions** element (section 2.2.1.2.111) of the **xDocumentClass** element (section 2.2.1.2.1).

2.2.1 Form Definition File (XSF) Specification

The following tables list, in alphabetical order, the types and elements used in the XML schema for the form definition (.xsf) file. The types and elements belong to the **XSF namespace** (http://schemas.microsoft.com/office/infopath/2003/solutionDefinition).

The XML schema is extended by the additional types and elements specified in sections 2.2.2 and 2.2.3.

The **xDocumentClass** element (section 2.2.1.2.1) MUST be the root element of the form definition (.xsf) file.

Type	Specified in Section
xdDesignMode	2.2.1.1.13
xdEmptyString	2.2.1.1.11
xdEnabledDisabled	2.2.1.1.5
xdErrorMessage	2.2.1.1.12
xdExpressionLiteral	2.2.1.1.7

Type	Specified in Section
xdFileName	2.2.1.1.8
xdHWSCaption	2.2.1.1.19
xdHWSname	2.2.1.1.18
xdManualAuto	2.2.1.1.6
xdRoleName	2.2.1.1.3
xdScriptLanguage	2.2.1.1.9
xdSignatureRelationEnum	2.2.1.1.17
xdSignedDataBlockMessage	2.2.1.1.16
xdSignedDataBlockName	2.2.1.1.15
xdSignSignatureLineRuleEnum	2.2.1.1.20
xdSolutionVersion	2.2.1.1.10
xdTitle	2.2.1.1.1
xdTrustLevel	2.2.1.1.14
xdViewName	2.2.1.1.2
xdYesNo	2.2.1.1.4

Element	Specified in Section
action	2.2.1.2.71
adoAdapter	2.2.1.2.19
allowedActions	2.2.1.2.70
allowedControl	2.2.1.2.83
allowedTasks	2.2.1.2.72
applicationParameters	2.2.1.2.4
assignmentAction	2.2.1.2.120
attachmentFileName	2.2.1.2.38
attributeData	2.2.1.2.86
autoRecovery	2.2.1.2.12
bcc	2.2.1.2.35

Element	Specified in Section
bdcAdapter	2.2.1.2.129
button	2.2.1.2.87
button	2.2.1.2.91
calculatedField	2.2.1.2.128
calculations	2.2.1.2.127
cc	2.2.1.2.34
changeAdapterProperty	2.2.1.2.121
chooseFragment	2.2.1.2.88
closeDocumentAction	2.2.1.2.124
customCategory	2.2.1.2.76
customValidation	2.2.1.2.43
dataAdapters	2.2.1.2.40
dataObject	2.2.1.2.17
dataObjects	2.2.1.2.16
davAdapter	2.2.1.2.29
dialogBoxExpressionAction	2.2.1.2.118
dialogBoxMessageAction	2.2.1.2.117
documentSchema	2.2.1.2.42
documentSchemas	2.2.1.2.41
documentSignatures	2.2.1.2.106
documentVersionUpgrade	2.2.1.2.109
domEventHandler	2.2.1.2.47
domEventHandlers	2.2.1.2.46
editing	2.2.1.2.92
editWith	2.2.1.2.89
emailAdapter	2.2.1.2.32
errorCondition	2.2.1.2.44
errorMessage	2.2.1.2.45
errorMessage	2.2.1.2.56

Element	Specified in Section
exitRuleSet	2.2.1.2.116
exportToExcel	2.2.1.2.9
exportToWeb	2.2.1.2.8
extension	2.2.1.2.112
extensions	2.2.1.2.111
externalView	2.2.1.2.85
externalViews	2.2.1.2.84
featureRestrictions	2.2.1.2.6
field	2.2.1.2.28
field	2.2.1.2.52
field	2.2.1.2.131
field	2.2.1.2.133
fields	2.2.1.2.51
file	2.2.1.2.79
fileName	2.2.1.2.31
fileNew	2.2.1.2.74
fileProperties	2.2.1.2.80
files	2.2.1.2.78
folderURL	2.2.1.2.30
footer	2.2.1.2.98
fragmentToInsert	2.2.1.2.94
getUserNameFromData	2.2.1.2.65
grooveAdapter	2.2.1.2.130
group	2.2.1.2.67
header	2.2.1.2.97
hwsAdapter	2.2.1.2.21
hwsOperation	2.2.1.2.23
hwsWorkflow	2.2.1.2.68
importParameters	2.2.1.2.48

Element	Specified in Section
importSource	2.2.1.2.49
initialXmlDocument	2.2.1.2.75
input	2.2.1.2.24
intro	2.2.1.2.37
listProperties	2.2.1.2.50
location	2.2.1.2.69
mainpane	2.2.1.2.95
masterDetail	2.2.1.2.93
membership	2.2.1.2.64
menu	2.2.1.2.100
menuArea	2.2.1.2.101
message	2.2.1.2.108
onLoad	2.2.1.2.60
openNewDocumentAction	2.2.1.2.123
operation	2.2.1.2.22
override	2.2.1.2.3
package	2.2.1.2.77
partFragment	2.2.1.2.25
permissions	2.2.1.2.82
print	2.2.1.2.10
printSettings	2.2.1.2.96
property	2.2.1.2.81
query	2.2.1.2.13
query	2.2.1.2.18
queryAction	2.2.1.2.122
role	2.2.1.2.63
roles	2.2.1.2.62
rule	2.2.1.2.114
ruleSet	2.2.1.2.125

Element	Specified in Section
ruleSetAction	2.2.1.2.113
ruleSets	2.2.1.2.126
save	2.2.1.2.7
save	2.2.1.2.61
schemaErrorMessages	2.2.1.2.2
script	2.2.1.2.15
scripts	2.2.1.2.14
sendMail	2.2.1.2.11
sharepointListAdapter	2.2.1.2.27
sharePointListAdapterRW	2.2.1.2.132
signedDataBlock	2.2.1.2.107
signSignatureLineAction	2.2.1.2.135
solutionProperties	2.2.1.2.5
subject	2.2.1.2.36
submit	2.2.1.2.53
submitAction	2.2.1.2.54
submitAction	2.2.1.2.115
submitToHostAdapter	2.2.1.2.39
successMessage	2.2.1.2.55
switchViewAction	2.2.1.2.119
task	2.2.1.2.73
taskpane	2.2.1.2.102
to	2.2.1.2.33
toolbar	2.2.1.2.99
unboundControls	2.2.1.2.90
useHttpHandler	2.2.1.2.57
useQueryAdapter	2.2.1.2.59
userName	2.2.1.2.66
useScriptHandler	2.2.1.2.58

Element	Specified in Section
useTransform	2.2.1.2.110
view	2.2.1.2.104
views	2.2.1.2.103
webPartConnectionAction	2.2.1.2.134
webServiceAdapter	2.2.1.2.20
xDocumentClass	2.2.1.2.1
xmlFileAdapter	2.2.1.2.26
xmlToEdit	2.2.1.2.105

2.2.1.1 Form Definition File XSF Enumerations

This section specifies the types used by elements and attributes in the **XSF namespace** (http://schemas.microsoft.com/office/infopath/2003/solutionDefinition).

2.2.1.1.1 xdTitle

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

Referenced By
adoAdapter.xsfschema@name
adoAdapterExtension.xsf2@ref
adoAdapterExtension.xsf2@submitAdapterName
bdcAdapter.xsfschema@name
bdcAdapter.xsfschema@submitAdapterName
button.xsfschema@caption
button.xsfschema@tooltip
command.xsf2@caption
customCategory.xsfschema@name
dataObject.xsfschema@name
davAdapter.xsfschema@name
davAdapterExtension.xsf2@ref
editWith.xsfschema@caption
emailAdapter.xsfschema@name

Referenced By
emailAdapterExtension.xsf2@ref
field.xsfschema@columnName
field.xsfschema@name
grooveAdapter.xsfschema@name
hwsAdapter.xsfschema@name
initialXmlDocument.xsfschema@caption
inputScope.xsf2@caption
menu.xsfschema@caption
sharepointListAdapter.xsfschema@name
sharepointListAdapterExtension.xsf2@ref
sharepointListAdapterRW.xsfschema@name
sharepointListAdapterRWExtension.xsf2@ref
submitAction.rule.xsfschema@adapter
submitAction.submit.xsf2@adapter
submitAction.submit.xsfschema@adapter
submitToHostAdapter.xsfschema@name
toolbar.xsfschema@caption
toolbar.xsfschema@name
viewExtension.xsf2@ref
viewExtension.xsf3@ref
webServiceAdapter.xsfschema@name
webServiceAdapterExtension.xsf2@ref
xmlFileAdapter.xsfschema@name
xmlFileAdapterExtension.xsf2@ref
xmlToEditExtension.xsf2@ref
xmlToEditExtension.xsf3@ref

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

```
<xsd:simpleType name="xdTitle">
  <xsd:restriction base="xsd:string">
```

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

2.2.1.1.2 xdViewName

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

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

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

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

2.2.1.1.3 xdRoleName

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

Referenced By
role.xsfschema@name

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

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

2.2.1.1.4 xdYesNo

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

no: This value evaluates to "no".

yes: This value evaluates to "yes".

Referenced By
action.xsfschema@canInitiateWorkflow
adoAdapter.xsfschema@queryAllowed
adoAdapter.xsfschema@submitAllowed
autoUpdatePrompt.xsf2@showPrompt
baseUrl.xsf3@queryServerInfo
bdcAdapter.xsfschema@queryAllowed
bdcAdapter.xsfschema@submitAllowed
calculations.xsfschema@treatBlankValueAsZero
closeDocumentAction.xsfschema@promptToSaveChanges
contentTypeTemplate.xsf2@browserEnable
dataObject.xsfschema@initOnLoad
davAdapter.xsfschema@overwriteAllowed
davAdapter.xsfschema@queryAllowed
davAdapter.xsfschema@submitAllowed
documentSchema.xsfschema@rootSchema
editWith.xsfschema@autoComplete
editWith.xsfschema@proofing
emailAdapter.xsfschema@queryAllowed
emailAdapter.xsfschema@submitAllowed
field.grooveAdapter.xsfschema@isLookup
field.sharepointListAdapter.xsfschema@isLookup
field.sharepointListAdapterRW.xsfschema@appendOnly
field.sharepointListAdapterRW.xsfschema@required
field.xsfschema@required
field.xsfschema@viewable

Referenced By
fieldExtension.xsf2@readWrite
grooveAdapter.xsfschema@queryAllowed
grooveAdapter.xsfschema@queryThisFormOnly
grooveAdapter.xsfschema@submitAllowed
hwsAdapter.xsfschema@queryAllowed
hwsAdapter.xsfschema@submitAllowed
hwsWorkflow.xsfschema@taskpaneVisible
importParameters.xsfschema@enabled
importParameters.xsfschema@useScriptHandler
managedCode.xsf2@enabled
mergedPrintView.xsf2@isCustomizable
mergedPrintView.xsf2@isDefault
offline.xsf2@cacheQueries
offline.xsf2@openIfQueryFails
partFragment.xsfschema@sendAsString
printSettings.xsfschema@collate
roles.xsfschema@hideStatusBarDisplay
rule.xsfschema@isEnabled
scripts.xsfschema@enforceScriptTimeout
sendByMail.xsf2@disableEmailForms
server.xsf2@isMobileEnabled
server.xsf2@isPreSubmitPostBackEnabled
sharepointListAdapter.xsfschema@queryAllowed
sharepointListAdapter.xsfschema@submitAllowed
sharepointListAdapterExtension.xsf2@queryThisFormOnly
sharepointListAdapterRW.xsfschema@autogen
sharepointListAdapterRW.xsfschema@queryAllowed
sharepointListAdapterRW.xsfschema@queryOneItemOnly
sharepointListAdapterRW.xsfschema@sortAscending

Referenced By
sharepointListAdapterRW.xsfschema@submitAllowed
sharepointListAdapterRWExtension.xsf2@queryThisFormOnly
signatureLine.xsf3@showDate
signSignatureLineAction.xsfschema@checkHostEnabled
signSignatureLineAction.xsfschema@isExpression
signSignatureLineAction.xsfschema@isSignaturePictureExpression
signSignatureLineAction.xsfschema@signaturePictureEnabled
solutionDefinition.xsf2@allowClientOnlyCode
solutionDefinition.xsf2@verifyOnServer
solutionMode.xsf3@autogenerated
solutionMode.xsf3@isListEditForm
solutionProperties.xsfschema@allowCustomization
solutionProperties.xsfschema@automaticallyCreateNodes
submit.xsf2@disableMenuItem
submit.xsf2@showSignatureReminder
submit.xsf2@showStatusDialog
submit.xsfschema@disableMenuItem
submit.xsfschema@showSignatureReminder
submit.xsfschema@showStatusDialog
submitToHostAdapter.xsfschema@queryAllowed
submitToHostAdapter.xsfschema@submitAllowed
suggestedSignerEmailAddress.signatureLin@isCalculation
suggestedSignerName.signatureLine.xsf3@isCalculation
suggestedSignerTitle.signatureLine.xsf3@isCalculation
toolbar.xsf2@enabledBottom
toolbar.xsf2@enabledTop
view.xsfschema@showMenuItem
viewExtension.xsf2@clientOnly
viewExtension.xsf2@readOnly

Referenced By
warning.xsf2@hidden
webServiceAdapter.xsfschema@queryAllowed
webServiceAdapter.xsfschema@submitAllowed
webServiceAdapter.xsfschema@useDataSet
webServiceAdapterExtension.xsf2@trackDataSetChanges
wss.xsf2@browserEnable
xDocumentClass.xsfschema@dataFormSolution
xDocumentClass.xsfschema@requireFullTrust
xmlFileAdapterExtension.xsf2@isRest
xmlToEditExtension.xsf2@allowLinkedImages
xmlToEditExtension.xsf2@excludeEmbeddedImages
xmlToEditExtension.xsf3@excludeHyperlink
xmlToEditExtension.xsf3@excludeTables

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

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

2.2.1.1.5 xdEnabledDisabled

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

disabled: This value evaluates to "disabled".

enabled: This value evaluates to "enabled".

Referenced By
autoRecovery.xsfschema@feature
exportToExcel.xsfschema@ui
exportToPDForXPS.xsf2@ui
exportToWeb.xsfschema@ui

Referenced By
print.xsfschema@ui
save.featureRestrictions.xsfschema@ui
sendMail.xsfschema@ui

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

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

2.2.1.1.6 xdManualAuto

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

automatic: This value evaluates to "automatic".

manual: This value evaluates to "manual".

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

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

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

2.2.1.1.7 xdExpressionLiteral

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

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

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

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

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

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

2.2.1.1.8 xdFileName

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

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

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

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

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

2.2.1.1.9 xdScriptLanguage

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

The **language** attribute of the **scripts** element (section 2.2.1.2.14) MUST NOT be present. The **scriptLanguage** attribute of the **solutionProperties** element (section 2.2.1.2.5) MUST be ignored.

Referenced By
scripts.xsfschema@language
solutionProperties.xsfschema@scriptLanguage

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

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

2.2.1.1.10 xdSolutionVersion

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

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

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

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

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

2.2.1.1.11 xdEmptyString

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

Referenced By
useTransform.xsfschema@transform

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

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

2.2.1.1.12 xdErrorMessage

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

Referenced By
errorMessage.xsfschema

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

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

2.2.1.1.13 xdDesignMode

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

normal: This value evaluates to "normal".

protected: This value evaluates to "protected".

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

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

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

2.2.1.1.14 xdTrustLevel

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

domain: This value evaluates to "domain".

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

Referenced By
xDocumentClass.xsfschema@trustLevel

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

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

2.2.1.1.15 xdSignedDataBlockName

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

Referenced By
signedDataBlock.xsfschema@name

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

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

2.2.1.1.16 xdSignedDataBlockMessage

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

Referenced By
confirmationMessage.signatureLine.xsf3
message.signedDataBlock.xsfschema

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

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

2.2.1.1.17 xdSignatureRelationEnum

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

cosign: This value evaluates to "cosign".

countersign: This value evaluates to "countersign".

single: This value evaluates to "single".

Referenced By
signedDataBlock.xsfschema@mode

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

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

2.2.1.1.18 xdHWSname

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

Referenced By

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

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

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

2.2.1.1.19 xdHWSCaption

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

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

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

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

2.2.1.1.20 xdSignSignatureLineRuleEnum

The **xdSignSignatureLineRuleEnum** simple type specifies restrictions for an attribute that MUST be ignored.

signatureLineId : This value MUST be ignored.

suggestedSignerEmail : This value MUST be ignored.

suggestedSignerName : This value MUST be ignored.

Referenced By
signSignatureLineAction.xsfschema@matchCriteria

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

```
<xsd:simpleType name="xdSignSignatureLineRuleEnum">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="suggestedSignerName"/>
    <xsd:enumeration value="suggestedSignerEmail"/>
    <xsd:enumeration value="signatureLineId"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.1.2 Form Definition File XSF Elements

This section specifies elements in the **XSF namespace** (http://schemas.microsoft.com/office/infopath/2003/solutionDefinition).

2.2.1.2.1 xDocumentClass

The **xDocumentClass** element and its child elements specify the properties, appearance, content, and files of the form template. The values specified by this element determine the general behaviors, such as loading and upgrading, of all forms (1) generated from the form template. This element MUST be the root element of the form definition (.xsf) file.

Child Elements
applicationParameters
calculations
customValidation
dataAdapters
dataObjects
documentSchemas
documentSignatures
documentVersionUpgrade
domEventHandlers
extensions
externalViews
featureRestrictions
fileNew
hwsWorkflow
importParameters
listProperties
onLoad

Child Elements
package
permissions
query
roles
ruleSets
save
schemaErrorMessages
scripts
submit
taskpane
views

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 **Uniform Resource Name (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 MUST be "15.0.0.0".

publishUrl: This attribute MUST NOT be present.

requireFullTrust: This attribute specifies whether the form template requires an elevated **form security level**. If "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. This attribute's value MUST be "3.0.0.0" or "15.0.0.0". If the value is set to "2.0.0.0", "1.1.0.0" or "1.0.0.0", the form template MUST conform to [MS-IPFF].

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 attribute specifies the default security context that can be overridden using the **requireFullTrust** attribute. The value of this attribute MUST NOT be "restricted".

trustSetting: This attribute MUST be ignored.

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

```
<xsd:element name="xDocumentClass">
  <xsd:complexType>
    <xsd:all>
      <xsd:element ref="xsf:package" minOccurs="1"/>
      <xsd:element ref="xsf:permissions" minOccurs="0"/>
      <xsd:element ref="xsf:views" minOccurs="1"/>
      <xsd:element ref="xsf:hwsWorkflow" minOccurs="0"/>
      <xsd:element ref="xsf:externalViews" minOccurs="0"/>
      <xsd:element ref="xsf:scripts" minOccurs="0"/>
      <xsd:element ref="xsf:schemaErrorMessages" minOccurs="0"/>
      <xsd:element ref="xsf:documentSchemas" minOccurs="0"/>
      <xsd:element ref="xsf:applicationParameters" minOccurs="0"/>
      <xsd:element ref="xsf:featureRestrictions" minOccurs="0"/>
      <xsd:element ref="xsf:fileNew" minOccurs="0"/>
      <xsd:element ref="xsf:customValidation" minOccurs="0"/>
      <xsd:element ref="xsf:domEventHandlers" minOccurs="0"/>
      <xsd:element ref="xsf:importParameters" minOccurs="0"/>
      <xsd:element ref="xsf:listProperties" minOccurs="0"/>
      <xsd:element ref="xsf:taskpane" minOccurs="0"/>
      <xsd:element ref="xsf:documentSignatures" minOccurs="0"/>
      <xsd:element ref="xsf:dataObjects" minOccurs="0"/>
      <xsd:element ref="xsf:dataAdapters" minOccurs="0"/>
      <xsd:element ref="xsf:query" minOccurs="0"/>
      <xsd:element ref="xsf:submit" minOccurs="0"/>
      <xsd:element ref="xsf:save" minOccurs="0"/>
      <xsd:element ref="xsf:roles" minOccurs="0"/>
      <xsd:element ref="xsf:onLoad" minOccurs="0"/>
      <xsd:element ref="xsf:documentVersionUpgrade" minOccurs="0"/>
      <xsd:element ref="xsf:extensions" minOccurs="0"/>
      <xsd:element ref="xsf:ruleSets" minOccurs="0"/>
      <xsd:element ref="xsf:calculations" minOccurs="0"/>
    </xsd:all>
    <xsd:attribute name="name" type="xsd:string" use="optional"/>
    <xsd:attribute name="author" type="xsd:string" use="optional"/>
    <xsd:attribute name="description" use="optional">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="255"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="solutionVersion" type="xsf:xdSolutionVersion" use="optional"/>
    <xsd:attribute name="productVersion" type="xsd:string" use="optional"/>
    <xsd:attribute name="solutionFormatVersion" type="xsf:xdSolutionVersion" use="required"/>
    <xsd:attribute name="dataFormSolution" type="xsf:xdYesNo" use="optional"/>
    <xsd:attribute name="requireFullTrust" type="xsf:xdYesNo" use="optional"/>
    <xsd:attribute name="trustLevel" type="xsf:xdTrustLevel" use="optional"/>
```

```
      <xsd:attribute name="trustSetting" type="xsf:xdManualAuto" use="optional"/>
      <xsd:attribute name="publishUrl" type="xsd:string" use="optional"/>
    </xsd:complexType>
    <xsd:key name="view_name_key">
      <xsd:selector xpath="./xsf:views/xsf:view"/>
      <xsd:field xpath="@name"/>
    </xsd:key>
    <xsd:key name="externalView_name_key">
      <xsd:selector xpath="./xsf:externalViews/xsf:externalView"/>
      <xsd:field xpath="@name"/>
    </xsd:key>
    <xsd:key name="view_or_externalView_name_key">
      <xsd:selector xpath="./xsf:views/xsf:view | ./xsf:externalViews/xsf:externalView"/>
      <xsd:field xpath="@name"/>
    </xsd:key>
    <xsd:key name="ruleset_name_key">
      <xsd:selector xpath="./xsf:ruleSets/xsf:ruleSet"/>
      <xsd:field xpath="@name"/>
    </xsd:key>
    <xsd:key name="dataObject_name_key">
      <xsd:selector xpath="./xsf:dataObjects/xsf:dataObject"/>
      <xsd:field xpath="@name"/>
    </xsd:key>
    <xsd:unique name="adapter_name_unique">
      <xsd:selector xpath="./xsf:dataObjects/xsf:dataObject/xsf:query/* | ./xsf:query/* |
  ./xsf:dataAdapters/* | ./xsf:submit/xsf:webServiceAdapter | ./xsf:submit/xsf:davAdapter |
  ./xsf:submit/xsf:emailAdapter | ./xsf:submit/xsf:submitToHostAdapter"/>
      <xsd:field xpath="@name"/>
    </xsd:unique>
    <xsd:key name="adapter_name_key">
      <xsd:selector xpath="./xsf:dataAdapters/*"/>
      <xsd:field xpath="@name"/>
    </xsd:key>
    <xsd:unique name="view_external_name_unique">
      <xsd:selector xpath="./xsf:views/xsf:view | ./xsf:externalViews/xsf:externalView"/>
      <xsd:field xpath="@name"/>
    </xsd:unique>
  </xsd:element>
```

2.2.1.2.2 schemaErrorMessages

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

Parent Elements
xDocumentClass

Child Elements
override

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

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

2.2.1.2.3 override

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

Parent Elements
schemaErrorMessages

Child Elements
errorMessage

Attributes:

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

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

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

2.2.1.2.4 applicationParameters

The **applicationParameters** element MUST be ignored.

Parent Elements
xDocumentClass

Child Elements
solutionProperties

Attributes:

application: This attribute MUST be ignored.

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

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

2.2.1.2.5 solutionProperties

The **solutionProperties** element MUST be ignored.

Parent Elements
applicationParameters

Attributes:

allowCustomization: This attribute MUST be ignored.

automaticallyCreateNodes: This attribute MUST be ignored.

fullyEditableNamespace: This attribute MUST be ignored.

lastOpenView: This attribute MUST be ignored.

lastVersionNeedingTransform: This attribute MUST be ignored.

publishSaveUrl: This attribute MUST be ignored.

scriptLanguage: This attribute MUST be ignored.

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

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

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

2.2.1.2.6 featureRestrictions

The **featureRestrictions** 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
xDocumentClass

Child Elements
autoRecovery
exportToExcel
exportToWeb
print
save
sendMail

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

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

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

2.2.1.2.7 save

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

Parent Elements
featureRestrictions

Attributes:

ui: This attribute specifies whether the save feature is restricted via the form's menus, toolbars, or keyboard shortcuts.

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>
```

2.2.1.2.8 exportToWeb

The **exportToWeb** element MUST be ignored.

Parent Elements
featureRestrictions

Attributes:

ui: This attribute MUST be ignored.

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

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

2.2.1.2.9 exportToExcel

The **exportToExcel** element MUST be ignored.

Parent Elements
featureRestrictions

Attributes:

ui: This attribute MUST be ignored.

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

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

2.2.1.2.10 print

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

Parent Elements
featureRestrictions

Attributes:

ui: This attribute specifies whether the print feature is restricted via the form's menus, toolbars, or keyboard shortcuts.

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

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

2.2.1.2.11 sendMail

The **sendMail** element MUST be ignored.

Parent Elements
featureRestrictions

Attributes:

ui: This attribute MUST be ignored.

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

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

2.2.1.2.12 autoRecovery

The **autoRecovery** element MUST be ignored.

Parent Elements
featureRestrictions

Attributes:

feature: This attribute MUST be ignored.

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

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

2.2.1.2.13 query

The **query** 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, a **list (1)** as specified in section 2.2.1.2.132, or a Web service.

If the form template is designed based on a main data connection, as specified by the **dataFormSolution** attribute of the **xDocumentClass** element (section 2.2.1.2.1), the main data source XML schema is derived from the XML schema provided by the main data connection. If the

form template is not designed based on a main data connection, the main data source XML schema is derived from manual modifications or an external XML schema document.

Parent Elements
xDocumentClass

Child Elements
adoAdapter
bdcAdapter
queryAction
sharepointListAdapter
sharepointListAdapterRW
webServiceAdapter
xmlFileAdapter

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

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

2.2.1.2.14 scripts

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

Parent Elements
xDocumentClass

Child Elements

Child Elements
script

Attributes:

enforceScriptTimeout: This attribute MUST NOT be present.

language: This attribute MUST NOT be present.

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

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

2.2.1.2.15 script

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

Parent Elements
scripts

Attributes:

src: This attribute MUST NOT be present.

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

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

2.2.1.2.16 dataObjects

The **dataObjects** element specifies all **secondary data connections** that query a **secondary data source**. Secondary data sources are used only to provide data to populate the form file or to be used for form functionality.

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

Parent Elements
xDocumentClass

Child Elements
dataObject

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

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

2.2.1.2.17 dataObject

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

Parent Elements
dataObjects

Child Elements
query

Attributes:

initOnLoad: 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:element ref="xsf:sharepointListAdapterRW"/>
          </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>
```

2.2.1.2.18 query

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

Parent Elements
dataObject

Child Elements
adoAdapter
sharepointListAdapter
sharepointListAdapterRW
webServiceAdapter
xmlFileAdapter

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

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

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

2.2.1.2.19 adoAdapter

The **adoAdapter** element specifies the properties of a data adapter that MUST be created to query data from a database. The **ActiveX Data Objects (ADO)** data adapter MUST NOT support submitting the form file and MUST NOT support querying an Access data source (2).

Parent Elements
dataAdapters
query

Attributes:

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

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

2.2.1.2.20 webServiceAdapter

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

When a form (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 template file, as specified in section 2.6, which is populated by data extracted from the form file.

Parent Elements
dataAdapters
query
submit

Child Elements
operation

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 of the Web service. It MUST be either an **absolute URL** or a **server-relative URL** or relative to the form template's location. 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>
```

2.2.1.2.21 hwsAdapter

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

Parent Elements
dataAdapters

Child Elements
hwsOperation

Attributes:

name: This attribute MUST NOT be present.

queryAllowed: This attribute MUST NOT be present.

submitAllowed: This attribute MUST NOT be present.

wsdlUrl: This attribute MUST NOT be present.

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

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

2.2.1.2.22 operation

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

Parent Elements
webServiceAdapter

Child Elements
input

Attributes:

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

serviceUrl: This attribute specifies the URL of the Web service to which the request is sent. It MUST be either an absolute URL or a server-relative URL or relative to the form template's location.

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>
```

2.2.1.2.23 hwsOperation

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

Parent Elements
hwsAdapter

Child Elements
input

Attributes:

serviceUrl: This attribute MUST NOT be present.

type: This attribute MUST NOT be present.

typeID: This attribute MUST NOT be present.

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

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

2.2.1.2.24 input

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

Parent Elements
hwsOperation
operation

Child Elements
partFragment

Attributes:

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

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

```
<xsd:element name="input">
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element ref="xsf:partFragment"/>
    </xsd:choice>
    <xsd:attribute name="source" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.1.2.25 partFragment

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

Parent Elements
input

Attributes:

dataObject: This attribute MUST be ignored.

filter: This attribute specifies an XPath expression that MUST evaluate to an XML sub-tree 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 sub-tree MUST replace a part of the SOAP message. If the filter attribute is not present, an XML node MUST replace a part of the SOAP message.

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

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

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

2.2.1.2.26 xmlFileAdapter

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

Parent Elements
dataAdapters
query

Attributes:

fileUrl: This attribute specifies either the URL of an XML file that is not contained in the form template or the name of an XML file that is contained in the form template. It MUST be either an

absolute URL or server-relative URL or relative to the form template's location if it is specifying the location of a file that is not contained in the form template. It MUST start with "x-soln:///" if it is specifying the name of an XML 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="xsf:xdTitle" use="optional"/>
    <xsd:attribute name="fileUrl" type="xsd:anyURI" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.1.2.27 sharepointListAdapter

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

Parent Elements
dataAdapters
query

Child Elements
field

Attributes:

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

name: 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 the 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 list (1).

siteUrl: This attribute specifies the URL of the parent **site (2)**.

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

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

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

2.2.1.2.28 field

The **field** element specifies mapping information for a list (1) field (3) that is used by the list (1) data adapter to query a list (1). Each list (1) field (3) returned from a query MUST be specified by an instance of this element.

Parent Elements
sharepointListAdapter

Attributes:

infopathName: This attribute specifies the name of the field (3) in the form view that corresponds to the list (1) field (3) name, as specified by the value of the **sharepointName** attribute.

isLookup: This attribute specifies whether the field (3) is considered a **lookup field**. If this attribute is not present, its value MUST be interpreted as "no".

sharepointName: This attribute specifies the list (1) field (3) name that corresponds to the name of the field (3) 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>
```

2.2.1.2.29 davAdapter

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

Parent Elements
dataAdapters
submit

Child Elements
fileName
folderURL

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>
```

2.2.1.2.30 folderURL

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

Parent Elements
davAdapter

Attributes:

value: This attribute specifies the server URL. The specified value MUST be either an absolute URL that begins with "http://" or "https://" or a server-relative URL, or relative to the form template's location.

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>
```

2.2.1.2.31 fileName

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

Parent Elements
davAdapter

Attributes:

value: 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 of the **value** attribute MUST be interpreted. If this attribute is set to "expression", the value of the **value** attribute MUST be evaluated as an XPath

expression. If this attribute is set to "literal", the value of the **value** attribute MUST be evaluated as a literal string.

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

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

2.2.1.2.32 emailAdapter

The **emailAdapter** element specifies the information needed to submit the form as an attachment to an e-mail with a specified set of recipients, subject, and an introduction. The e-mail MUST have a set of recipients specified by the **to** element (section 2.2.1.2.33), **cc** element (section 2.2.1.2.34), or **bcc** element (section 2.2.1.2.35).

Parent Elements
dataAdapters
submit

Child Elements
attachmentFileName
bcc
cc
intro
subject
to

Attributes:

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

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

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

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

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

2.2.1.2.33 to

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

Parent Elements

Parent Elements
emailAdapter

Attributes:

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

valueType: This attribute specifies how the value of the **value** attribute MUST be interpreted. If this attribute is set to "expression", the value of the **value** attribute MUST be evaluated as an XPath expression. If this attribute is set to "literal", the value of the **value** attribute 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>
```

2.2.1.2.34 cc

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

Parent Elements
emailAdapter

Attributes:

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

valueType: This attribute specifies how the value of the **value** attribute MUST be interpreted. If this attribute is set to "expression", the value of the **value** attribute MUST be evaluated as an XPath expression. If this attribute is set to "literal", the value of the **value** attribute 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>
```

2.2.1.2.35 bcc

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

Parent Elements
emailAdapter

Attributes:

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

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

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

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

2.2.1.2.36 subject

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

Parent Elements
emailAdapter

Attributes:

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

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

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

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

2.2.1.2.37 intro

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

Parent Elements
emailAdapter

Attributes:

value: This attribute specifies the body text.

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

```
<xsd:element name="intro" minOccurs="0">
  <xsd:complexType>
    <xsd:attribute name="value" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.1.2.38 attachmentFileName

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

Parent Elements
emailAdapter

Attributes:

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

valueType: This attribute specifies how the value of the **value** attribute MUST be interpreted. If this attribute is set to "expression", the value of the **value** attribute MUST be evaluated as an XPath

expression. If this attribute is set to "literal", the value of the **value** attribute MUST be evaluated as a literal string.

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

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

2.2.1.2.39 submitToHostAdapter

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

Parent Elements
dataAdapters
submit

Attributes:

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

queryAllowed: This attribute MUST be ignored.

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

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

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

2.2.1.2.40 dataAdapters

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

Parent Elements

Parent Elements
xDocumentClass

Child Elements
adoAdapter
davAdapter
emailAdapter
hwsAdapter
sharepointListAdapter
submitToHostAdapter
webServiceAdapter
xmlFileAdapter

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

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

2.2.1.2.41 documentSchemas

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

The root element of the authoritative XML schema is defined in the XML schema identified by the **rootSchema** attribute of the **documentSchema** element (section 2.2.1.2.42). Additional XML schemas are included by using the XML schema's **import** or **include** constructs as follows:

- If the XML schema is included using the XML schema **import** construct, as specified by [XMLSCHEMA1], a **documentSchema** element MUST exist for that imported XML schema.

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

Parent Elements
xDocumentClass

Child Elements
documentSchema

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

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

2.2.1.2.42 documentSchema

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

Parent Elements
documentSchemas

Attributes:

location: This attribute specifies the XML schema location as either 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 (section 2.2.1.2.81) where the **name** attribute is "namespace". All XML schema documents in the form template MUST use different namespaces.

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

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

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

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

2.2.1.2.43 customValidation

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

Parent Elements
xDocumentClass

Child Elements
errorCondition

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

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

2.2.1.2.44 errorCondition

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

Any XML processing instruction named **Caption** inside this element MUST be ignored.

Parent Elements
customValidation

Child Elements
errorMessage

Attributes:

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

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

2.2.1.2.45 errorMessage

The **errorMessage** element specifies the error message that MUST be returned if the value of the specified XML node is considered invalid according to the value specified by the **expression** attribute of the **errorCondition** element (section 2.2.1.2.44).

Parent Elements
errorCondition
override

Attributes:

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

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

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

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

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

2.2.1.2.46 domEventHandlers

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

Parent Elements
xDocumentClass

Child Elements
domEventHandler

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

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

2.2.1.2.47 domEventHandler

The **domEventHandler** element specifies a handler for events triggered when the specified XML nodes change. The child rule set is run when this handler is called. The various types of child rules (1) specify supported actions to take on the form file.

Parent Elements
domEventHandlers

Child Elements
ruleSetAction

Attributes:

dataObject: This attribute specifies the name of the secondary data source that MUST be used in the event handler. The specified name MUST match the value specified by the corresponding name attribute of the **dataObject** element (section 2.2.1.2.17). 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>
```

2.2.1.2.48 importParameters

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

Parent Elements

Parent Elements
xDocumentClass

Child Elements
importSource

Attributes:

enabled: This attribute specifies whether form merging is enabled.

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

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

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

2.2.1.2.49 importSource

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

Parent Elements
importParameters

Attributes:

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

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

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>
```

2.2.1.2.50 listProperties

The **listProperties** element specifies a collection of fields (3) that are promoted from the form file and made available to the default **list view** of a **form library** as properties on that form library.

Parent Elements
xDocumentClass

Child Elements
fields

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

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

2.2.1.2.51 fields

The **fields** element specifies a collection of fields (3) that are promoted from the form file and made available to the default list view of a form library.

Parent Elements
listProperties

Child Elements

Child Elements
field

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

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

2.2.1.2.52 field

The **field** element specifies a field (3) that is promoted from the form file and made available to the default list view of a form library. Each promoted field (3) MUST be specified by an instance of this element.

Parent Elements
fields

Attributes:

aggregation: This attribute specifies how a single XML node or a collection of XML nodes returned from evaluating the **node** attribute value is aggregated to obtain a value for the field (3). 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 aggregate MUST be determined by calculating the average of all XML node values. This value MUST NOT be specified if any of the XML nodes is not of an XML schema **number** data type.
- **count:** The aggregate value MUST be determined by calculating the number of XML nodes.
- **first:** The aggregate value MUST be determined by returning the first XML node value in the collection.
- **last:** The aggregate value MUST be determined by returning the last XML node value in the collection.
- **max:** The aggregate value MUST be determined by calculating the maximum XML node value in the collection. This value MUST NOT be specified if any of the XML nodes is not of an XML schema **number** data type.
- **merge:** The aggregate value MUST be determined by concatenating all XML node values in the collection separated by **newline** characters.

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

columnName: This attribute specifies the internal name of the corresponding **column (2)** in the **SQL** database underlying the list view. The specified value MUST match the **columnName** attribute for the **fieldExtension** element, as specified in section 2.2.2.2.20.

maxLength: This attribute specifies the maximum length of the field (3) in the number of bytes. If this attribute is not present, its value MUST be determined by the field type and site settings.

name: This attribute specifies the friendly name of the field (3) used on the list view.

node: This attribute specifies the XPath expression that evaluates to the corresponding field (3) in the form file.

required: This attribute specifies whether this field (3) 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 (3).

viewable: This attribute specifies whether this field (3) is added to the default list view. If this attribute is not present, its value MUST be interpreted as "yes".

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

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

```
            <xsd:enumeration value="plaintext"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:complexType>
  </xsd:element>
```

2.2.1.2.53 submit

The **submit** element specifies the necessary information for configuring the submit operation for the form (1). This includes information regarding the method to use, UI elements that call the operation, and related actions to perform after the submit operation is complete. A **submit** element can be associated with a data adapter, rule set, script handler, or HTTP handler as follows:

- **Data adapter:** The form file is submitted using a data adapter if the target for the submit operation is a data source (2) with an associated data adapter, as specified by the **davAdapter** element (section 2.2.1.2.29), **emailAdapter** element (section 2.2.1.2.32), **submitToHostAdapter** element (section 2.2.1.2.39), and **webServiceAdapter** element (section 2.2.1.2.20). The **data adapter** child element MUST have the **submitAllowed** attribute set to "yes".
- **Rule Set:** The form file is submitted by an associated collection of rules (1) that run associated actions specified by the **ruleSetAction** element (section 2.2.1.2.113).
- **Script Handler:** The form file is submitted by associated form code specified by the **useScriptHandler** element (section 2.2.1.2.58).
- **HTTP Handler:** The form file is submitted using the **HTTP method** specified by the **useHttpHandler** element (section 2.2.1.2.57).

Prior to the **submit** operation being performed, the form file MUST fully conform to the XML schema specified in section 2.3 and to any custom validation defined in the form template, as specified by the **customValidation** element (section 2.2.1.2.43).

Parent Elements
xDocumentClass

Child Elements
davAdapter
emailAdapter
errorMessage
ruleSetAction
submitAction
submitToHostAdapter

Child Elements
successMessage
useHttpHandler
useQueryAdapter
useScriptHandler
webServiceAdapter

Attributes:

caption: This attribute specifies the name of the submit button. A corresponding button MUST appear on the form view toolbar when the form (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 "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:** If this value is specified, the form (1) closes on successful submission of the form (1).
- **keepOpen:** If this value is specified, the form (1) does NOT close on successful submission of the form (1).
- **openNew:** If this value is specified, the form (1) closes and a new instance of the form (1) is opened on successful submission of the 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: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>
```

2.2.1.2.54 submitAction

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

Parent Elements
submit

Attributes:

adapter: This attribute MUST NOT be present.

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

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

2.2.1.2.55 successMessage

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

Parent Elements
submit

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

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

2.2.1.2.56 errorMessage

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

Parent Elements
submit

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

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

2.2.1.2.57 useHttpHandler

The **useHttpHandler** element specifies that the form (1) MUST be submitted to the specified URL using the specified HTTP method.

Parent Elements
submit

Attributes:

href: This attribute specifies the URL to which the form (1) is submitted. It MUST be either an absolute URL or server-relative URL or relative to the form template's location.

method: This attribute specifies the HTTP method that is used to submit the form (1). This value MUST be "POST", as specified in [HTML] section 17.13.1.

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

```
<xsd:element name="useHttpHandler">
  <xsd:complexType>
    <xsd:attribute name="method" use="required">
```

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

2.2.1.2.58 useScriptHandler

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

Parent Elements
documentVersionUpgrade
save
submit

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

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

2.2.1.2.59 useQueryAdapter

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

Parent Elements
submit

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

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

2.2.1.2.60 onLoad

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

Parent Elements
xDocumentClass

Child Elements
ruleSetAction

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

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

2.2.1.2.61 save

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

Parent Elements
xDocumentClass

Child Elements
useScriptHandler

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

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

2.2.1.2.62 roles

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

Parent Elements
xDocumentClass

Child Elements
membership
role

Attributes:

default: This attribute MUST NOT be present.

hideStatusBarDisplay: This attribute MUST NOT be present.

initiator: This attribute MUST NOT be present.

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

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

2.2.1.2.63 role

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

Parent Elements
roles

Attributes:

name: This attribute MUST NOT be present.

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

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

2.2.1.2.64 membership

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

Parent Elements
roles

Child Elements
getUserNameFromData
group
userName

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

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

2.2.1.2.65 getUserNameFromData

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

Parent Elements
membership

Attributes:

dataObject: This attribute MUST NOT be present.

memberOf: This attribute MUST NOT be present.

select: This attribute MUST NOT be present.

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

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

2.2.1.2.66 userName

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

Parent Elements
membership

Attributes:

memberOf: This attribute MUST NOT be present.

name: This attribute MUST NOT be present.

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

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

2.2.1.2.67 group

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

Parent Elements
membership

Attributes:

memberOf: This attribute MUST NOT be present.

name: This attribute MUST NOT be present.

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

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

2.2.1.2.68 hwsWorkflow

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

Parent Elements
xDocumentClass

Child Elements
allowedActions
allowedTasks
location

Attributes:

taskpaneVisible: This attribute MUST NOT be present.

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

```
<xsd:element name="hwsWorkflow">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:location" minOccurs="1" maxOccurs="1"/>
```

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

2.2.1.2.69 location

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

Parent Elements
hwsWorkflow

Attributes:

url: This attribute MUST NOT be present.

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

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

2.2.1.2.70 allowedActions

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

Parent Elements
hwsWorkflow

Child Elements
action

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

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

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

2.2.1.2.71 action

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

Parent Elements
allowedActions

Attributes:

actionTypeID: This attribute MUST NOT be present.

canInitiateWorkflow: This attribute MUST NOT be present.

caption: This attribute MUST NOT be present.

name: This attribute MUST NOT be present.

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

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

2.2.1.2.72 allowedTasks

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

Parent Elements
hwsWorkflow

Child Elements

Child Elements
task

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

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

2.2.1.2.73 task

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

Parent Elements
allowedTasks

Attributes:

caption: This attribute MUST NOT be present.

name: This attribute MUST NOT be present.

taskTypeID: This attribute MUST NOT be present.

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

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

2.2.1.2.74 fileNew

The **fileNew** element specifies the name and location of the XML template file that contains default values for a new form (1) based on the form template.

Parent Elements

Parent Elements
xDocumentClass

Child Elements
initialXmlDocument

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

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

2.2.1.2.75 initialXmlDocument

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

When the new form (1) is created, its field (3) values MUST be populated with existing default values specified by the template.xml file, as specified in section 2.7.

Parent Elements
fileNew

Child Elements
customCategory

Attributes:

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

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

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

```
<xsd:element name="initialXmlDocument">
  <xsd:complexType>
    <xsd:sequence>
```

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

2.2.1.2.76 customCategory

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

Parent Elements
initialXmlDocument

Attributes:

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

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

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

2.2.1.2.77 package

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

Parent Elements
xDocumentClass

Child Elements
files

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

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

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

2.2.1.2.78 files

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

Parent Elements
package

Child Elements
file

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

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

2.2.1.2.79 file

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

Parent Elements
files

Child Elements
fileProperties

Attributes:

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

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

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

2.2.1.2.80 fileProperties

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

Parent Elements
file

Child Elements
property

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

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

2.2.1.2.81 property

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

Parent Elements
fileProperties

Attributes:

name: This attribute specifies the name of the file property and MUST be set to one of the following acceptable values. Each **property** element MUST have a unique name within the set of file properties specified by each **fileProperties** element (section 2.2.1.2.80). The corresponding acceptable values of the **value** attribute are listed for each possible value of the **name** attribute:

- **componentId:** 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:** Specifies that the parent **file** element (section 2.2.1.2.79) 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:** 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:
  - **full:** The XML schema document MUST be fully editable.
  - **partial:** 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:** Specifies that the corresponding **value** attribute value specifies the file type. The **value** attribute value MUST be set to one of the following:
  - **pdb:** The file MUST be a **symbol file**. A form template containing business objects MAY have any number of files of this type.
  - **refAssembly:** The file MUST be a form code **assembly** other than the main form code assembly. A form template containing business objects MAY have any number of files of this type.
  - **rootAssembly:** 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:** The file MUST be used as a secondary data source.
  - **sampleData:** The file MUST be a file containing sample data for the form (1).
- **lang:** Specifies that the corresponding **value** attribute value MUST be the language of the form file. A form view file, as specified in section 2.4, MUST have a file property with this name. The **value** attribute value MUST be set to the **LCID**, as specified in [MS-LCID], corresponding to the user **locale**.
- **namespace:** 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**: 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 the root element of the XML schema document.
- **useOnDemandAlgorithm:** This value MUST be ignored.
- **xmlToEditName:** Specifies that the corresponding **value** attribute value MUST be equal to the value of the **value** attribute for the **componentId** file property. A form view file, as specified in section 2.4, MUST have a file property with this name. The **value** attribute value MUST be set to a positive integer.

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

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

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

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

2.2.1.2.82 permissions

The **permissions** element MUST be ignored.

Parent Elements
xDocumentClass

Child Elements
allowedControl

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

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

2.2.1.2.83 allowedControl

The **allowedControl** element MUST be ignored.

Parent Elements
permissions

Attributes:

cabFile: This attribute MUST be ignored.

clsid: This attribute MUST be ignored.

version: This attribute MUST be ignored.

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

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

2.2.1.2.84 externalViews

The **externalViews** element MUST be ignored.

Parent Elements
xDocumentClass

Child Elements
externalView

Attributes:

default: This attribute MUST be ignored.

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

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

2.2.1.2.85 externalView

The **externalView** element MUST be ignored.

Parent Elements
externalViews

Child Elements
mainpane

Attributes:

designMode: This attribute MUST be ignored.

name: This attribute MUST be ignored.

target: This attribute MUST be ignored.

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

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

2.2.1.2.86 attributeData

The **attributeData** element specifies the name and associated value of an attribute that MUST be inserted, or MUST be modified if it already exists, by the **insert** action of the **xCollection** or **xOptional** controls. This element MUST be a child element of the **chooseFragment** element, as specified in section 2.2.1.2.88.

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>
```

2.2.1.2.87 button

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

Parent Elements
menu
menuArea
toolbar

Attributes:

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

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

The specified value MUST be one of the following:

- **xCollection::insert:** This action inserts a new section after all existing sections.
- **xCollection::insertBefore:** This action inserts a new section before the current section.
- **xCollection::insertAfter:** This action inserts a new section after the current section.
- **xCollection::refreshFilter:** This action refreshes the control.
- **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.

- **xReplace::replace:** This action deletes the current section and inserts a new section in place of the previous current section.

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

icon: This attribute MUST be ignored.

name: This attribute MUST be ignored.

showIf: This attribute MUST<1> be ignored. The value MUST be one of the following:

- **always:** This value is deprecated.
- **enabled:** This value is deprecated.
- **immediate:** If this value is specified, this button element MUST be shown when the parent **menuArea** element (section 2.2.1.2.101) or **menu** element (section 2.2.1.2.100) are rendered.

tooltip: This attribute MUST be ignored.

xmlToEdit: This attribute specifies the name of the control for which the **button** is used. The specified value MUST match the value of the **name** attribute of the corresponding **xmlToEdit** element (section 2.2.1.2.105). This attribute MUST be defined for buttons used with collection controls.

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

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

```
          <xsd:restriction base="xsd:NMTOKEN">
            <xsd:enumeration value="always"/>
            <xsd:enumeration value="enabled"/>
            <xsd:enumeration value="immediate"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:complexType>
  </xsd:element>
```

2.2.1.2.88 chooseFragment

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

Parent Elements
fragmentToInsert

Attributes:

followingSiblings: This attribute specifies a relative XPath expression from the parent node. The parent node specifies the XML node prior to which the insertion of the XML fragment occurs. If the node is not found, the insertion action MUST be an append. If this attribute is not present, its value MUST be interpreted as an empty string (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:attribute name="innerFragment" type="xsd:string" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.1.2.89 editWith

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

Parent Elements
xmlToEdit

Child Elements
fragmentToInsert
masterDetail

Attributes:

allowedFileTypes: This attribute MUST be ignored.

autoComplete: This attribute specifies whether auto-completion of fields (3) is on. If this attribute is not present, its value MUST be interpreted as "no".

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

component: This attribute specifies the name of the control that is referenced by an instance of the **xmlToEdit** element, as specified in section 2.2.1.2.105. 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:** A choice section control, as specified in section 2.4.1.21.1.
- **xTextList:** A list control, as specified in section 2.4.1.21.7.
- **xField:** Specifies one of the following controls:
  - Check box control, as specified in section 2.4.1.6.
  - Combo box control, as specified in section 2.4.1.21.2.
  - Date picker control, as specified in section 2.4.1.8.
  - Drop-down list control, as specified in section 2.4.1.9.
  - Hyperlink input control, as specified in section 2.4.1.21.6.
  - 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:** An image attachment control, as specified in section 2.4.1.21.8.

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

field: This attribute MUST be ignored.

filterDependency: This attribute specifies a relative XPath expression to a control that is dependent on the **predicate_xpath** expression. If multiple controls are dependent on the XPath expression, each control MUST be separated by **ASCII** character "124", which represents a logical OR.

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". The specified value MUST be one of the following:

- **plain:** The instance of the control specified by this **editWidth** element MUST only allow the input of unformatted text.
- **formatted:** This value is deprecated.
- **plainMultiline:** Same as "plain", with the additional support of **newline** characters.
- **formattedMultiline:** This value is deprecated.
- **rich:** The instance of the control specified by this **editWidth** element MUST allow rich formatting using HTML, as specified in [HTML], as a valid XML 1.0 fragment, as specified in [W3C-XML].If the value is "rich", the following MUST be true:
  - If the **xmlToEditExtension** element, as specified in section 2.2.2.2.36, 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, as specified in section 2.2.2.2.35, that is the parent of the **xmlToEditExtension** element MUST be set to "no".

useFilter: This attribute MUST be "yes" if a **predicate_xpath** is specified for the control. Otherwise, the attribute MUST be "no" if there is not a **predicate_xpath** specified for the control.

widgetIcon: This attribute specifies if an icon is displayed when a **predicate_xpath** has been evaluated. The value MUST be set to "filter" if a **predicate_xpath** is specified for the control.

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

```
<xsd:element name="editWith">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:masterDetail" minOccurs="0" maxOccurs="1"/>
      <xsd:element ref="xsf:fragmentToInsert" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="component" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:NMTOKEN">
```

```
            <xsd:enumeration value="xCollection"/>
            <xsd:enumeration value="xOptional"/>
            <xsd:enumeration value="xReplace"/>
            <xsd:enumeration value="xTextList"/>
            <xsd:enumeration value="xField"/>
            <xsd:enumeration value="xImage"/>
            <xsd:enumeration value="xFileAttachment"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="caption" type="xsf:xdTitle" use="optional"/>
      <xsd:attribute name="autoComplete" type="xsf:xdYesNo" use="optional"/>
      <xsd:attribute name="proofing" type="xsf:xdYesNo" use="optional"/>
      <xsd:attribute name="type" use="optional">
        <xsd:simpleType>
          <xsd:restriction base="xsd:NMTOKEN">
            <xsd:enumeration value="plain"/>
            <xsd:enumeration value="formatted"/>
            <xsd:enumeration value="plainMultiline"/>
            <xsd:enumeration value="formattedMultiline"/>
            <xsd:enumeration value="rich"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="useFilter" use="optional">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="yes"/>
            <xsd:enumeration value="no"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="widgetIcon" use="optional">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="standard"/>
            <xsd:enumeration value="filter"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="filterDependency" type="xsd:string" use="optional"/>
      <xsd:attribute name="field" type="xsd:string" use="optional"/>
      <xsd:attribute name="removeAncestors" type="xsd:nonNegativeInteger" use="optional"/>
      <xsd:attribute name="maxLength" use="optional">
        <xsd:simpleType>
          <xsd:restriction base="xsd:integer">
            <xsd:minInclusive value="-1"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="allowedFileTypes" type="xsd:string" use="optional"/>
      <xsd:anyAttribute namespace="http://schemas.microsoft.com/office/infopath/2003"
    processContents="skip"/>
    </xsd:complexType>
  </xsd:element>
```

2.2.1.2.90 unboundControls

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

Parent Elements
view

Child Elements
button

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

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

2.2.1.2.91 button

The **button** element specifies a **button** that MAY have an associated event handler or **ruleSetAction**, as specified in section 2.2.1.2.113.

Parent Elements
unboundControls

Child Elements
ruleSetAction

Attributes:

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

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

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

2.2.1.2.92 editing

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

Parent Elements
view

Child Elements
xmlToEdit

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

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

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

2.2.1.2.93 masterDetail

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

Parent Elements
editWith

Attributes:

detailKey: This attribute MUST NOT be present.

master: This attribute MUST NOT be present.

masterKey: This attribute MUST NOT be present.

masterViewContext: This attribute MUST NOT be present.

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

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

2.2.1.2.94 fragmentToInsert

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

Parent Elements
editWith

Child Elements
chooseFragment

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

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

2.2.1.2.95 mainpane

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

Parent Elements
externalView
view

Attributes:

transform: This attribute specifies the name of the XSLT file that is used to transform the form (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>
```

2.2.1.2.96 printSettings

The **printSettings** element MUST be ignored.

Parent Elements
mergedPrintView
view

Child Elements
footer

Child Elements
header

Attributes:

bottomMargin: This attribute MUST be ignored.

collate: This attribute MUST be ignored.

copies: This attribute MUST be ignored.

footer: This attribute MUST be ignored.

header: This attribute MUST be ignored.

leftMargin: This attribute MUST be ignored.

marginUnitsType: This attribute MUST be ignored.

orientation: This attribute MUST be ignored.

pageRangeEnd: This attribute MUST be ignored.

pageRangeStart: This attribute MUST be ignored.

paperSize: This attribute MUST be ignored.

paperSource: This attribute MUST be ignored.

printerName: This attribute MUST be ignored.

printerSpecificSettings: This attribute MUST be ignored.

rightMargin: This attribute MUST be ignored.

topMargin: This attribute MUST be ignored.

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

```
<xsd:element name="printSettings">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:header" minOccurs="0" maxOccurs="1"/>
      <xsd:element ref="xsf:footer" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="orientation">
      <xsd:simpleType>
        <xsd:restriction base="xsd:NMTOKEN">
          <xsd:enumeration value="portrait"/>
          <xsd:enumeration value="landscape"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="header">
      <xsd:simpleType>
```

```
      <xsd:restriction base="xsd:string">
        <xsd:maxLength value="255"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="footer">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:maxLength value="255"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="marginUnitsType">
    <xsd:simpleType>
      <xsd:restriction base="xsd:NMTOKEN">
        <xsd:enumeration value="in"/>
        <xsd:enumeration value="cm"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="rightMargin">
    <xsd:simpleType>
      <xsd:restriction base="xsd:float">
        <xsd:minInclusive value="0"/>
        <xsd:maxInclusive value="100"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="leftMargin">
    <xsd:simpleType>
      <xsd:restriction base="xsd:float">
        <xsd:minInclusive value="0"/>
        <xsd:maxInclusive value="100"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="topMargin">
    <xsd:simpleType>
      <xsd:restriction base="xsd:float">
        <xsd:minInclusive value="0"/>
        <xsd:maxInclusive value="100"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="bottomMargin">
    <xsd:simpleType>
      <xsd:restriction base="xsd:float">
        <xsd:minInclusive value="0"/>
        <xsd:maxInclusive value="100"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="printerName">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:maxLength value="255"/>
      </xsd:restriction>
```

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

2.2.1.2.97 header

The **header** element MUST be ignored.

Parent Elements
printSettings

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

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

2.2.1.2.98 footer

The **footer** element MUST be ignored.

Parent Elements
printSettings

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

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

2.2.1.2.99 toolbar

The **toolbar** element MUST be ignored.

Parent Elements
view

Child Elements
button
menu

Attributes:

caption: This attribute MUST be ignored.

name: This attribute MUST be ignored.

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

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

2.2.1.2.100 menu

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

Parent Elements
menu
menuArea
toolbar
view

Child Elements
button
menu

Attributes:

caption: This attribute specifies the caption for the **menu**.

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

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

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

2.2.1.2.101 menuArea

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

Parent Elements
view

Child Elements
button
menu

Attributes:

name: This attribute MUST be set to "msoStructuralEditingContextMenu". Possible values for this attribute are as follows:

- **msoEditMenu:** This value MUST NOT be specified.
- **msoFileMenu:** This value MUST NOT be specified.
- **msoHelpMenu:** This value MUST NOT be specified.
- **msoInsertMenu:** This value MUST NOT be specified.
- **msoFormatMenu:** This value MUST NOT be specified.
- **msoStructuralEditingContextMenu:** Specifies the context menu that is shown for the following controls:
  - Choice group control and choice section (section 2.4.1.21.1)
  - 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:** This value MUST NOT be specified.
- **msoToolsMenu:** This value MUST NOT be specified.
- **msoViewMenu:** This value MUST NOT be specified.

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>
```

2.2.1.2.102 taskpane

The **taskpane** element MUST be ignored.

Parent Elements
xDocumentClass

Attributes:

caption: This attribute MUST be ignored.

href: This attribute MUST be ignored.

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

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

2.2.1.2.103 views

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

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

- The **clientOnly** attribute of the **viewExtension** element, as specified in section 2.2.2.2.35, is set to "no".
- The **designMode** attribute of the **view** element, as specified in section 2.2.1.2.104, is set to "normal".

Parent Elements
xDocumentClass

Child Elements
view

Attributes:

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

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

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

2.2.1.2.104 view

The **view** element specifies information about a form view, which is a specific visualization of a form file. It specifies what controls are used to represent the fields (3) in the form (1) and how they are rendered using HTML. A form view is represented by an XSLT file, as specified in section 2.4.

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

Parent Elements
views

Child Elements
editing
mainpane
menu
menuArea
printSettings
toolbar
unboundControls

Attributes:

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

2.2.1.2.105 xmlToEdit

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

Parent Elements
editing

Child Elements
editWith

Attributes:

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

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

name: This attribute specifies the name of the control.

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

2.2.1.2.106 documentSignatures

The **documentSignatures** element specifies the digital signatures (1), as specified in [MS-IPFFX], which are used to sign the form file, according to [XMLDSig].

Parent Elements
xDocumentClass

Child Elements
signedDataBlock

Attributes:

signatureLocation: This attribute MUST NOT be present.

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

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

2.2.1.2.107 signedDataBlock

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

Parent Elements
documentSignatures

Child Elements
message

Attributes:

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

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

- **countersign:** The digital signature (1) signs all previous digital signatures (1).
- **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>
```

2.2.1.2.108 message

The **message** 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
signedDataBlock

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

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

2.2.1.2.109 documentVersionUpgrade

The **documentVersionUpgrade** 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
xDocumentClass

Child Elements
useScriptHandler
useTransform

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

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

2.2.1.2.110 useTransform

The **useTransform** 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
documentVersionUpgrade

Attributes:

maxVersionToUpgrade: This attribute specifies the inclusive value for the maximum form template version, specified by the **solutionVersion** attribute of the **xDocumentClass** element, as specified in section 2.2.1.2.1, 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, which 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 or the value MUST be an empty string (1).

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>
```

2.2.1.2.111 extensions

The **extensions** element specifies the extensions to the properties and content of the form definition (.xsf) file. Each extension MUST conform to the XML schema specified in section 2.2.2 or section 2.2.3.

Parent Elements
xDocumentClass

Child Elements
extension

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

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

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

2.2.1.2.112 extension

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

Each extension MUST conform to the XML schema specified in section 2.2.2 or section 2.2.3.

The **XSF2:solutionDefinition** element, as specified in section 2.2.2.2.1, MUST be the root element of all **XSF2 extensions**, as specified in section 2.2.2.

The **XSF3:solutionDefinition** element, as specified in section 2.2.3.2.1, or **solutionPropertiesExtension2009** element, as specified in section 2.2.3.2.6, MUST be the containers of all **XSF3 extensions**, as specified in section 2.2.3.

Parent Elements
extensions

Attributes:

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

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

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

2.2.1.2.113 ruleSetAction

The **ruleSetAction** element specifies the rule set, as defined by the **ruleSet** element, as specified in section 2.2.1.2.125, that MUST be called by a form (1) or form file event.

Parent Elements
button
domEventHandler
onLoad
submit

Attributes:

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

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

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

2.2.1.2.114 rule

The **rule** 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 this element and the **ruleSet** element, as specified in section 2.2.1.2.125. The event is defined by the following elements:

- **button**, as specified in section 2.2.1.2.91.
- **domEventHandler**, as specified in section 2.2.1.2.47.
- **onLoad**, as specified in section 2.2.1.2.60.
- **submit**, as specified in section 2.2.1.2.53.

It is also defined by the **ruleSetAction** element associated with each of these elements.

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

Rules (1) are grouped together as a rule set, as specified by the **ruleSet** element specified in section 2.2.1.2.125, containing one or more rules (1). A rule set is bound to one of the following events with the **ruleSetAction** element, as specified in section 2.2.1.2.113:

- A form file change, such as a change in an XML node's value.
- 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, as specified in section 2.2.1.2.126.

Parent Elements

Parent Elements
ruleSet

Child Elements
assignmentAction
changeAdapterProperty
closeDocumentAction
dialogBoxExpressionAction
dialogBoxMessageAction
exitRuleSet
openNewDocumentAction
queryAction
signSignatureLineAction
submitAction
switchViewAction
webPartConnectionAction

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 ref="xsf:changeAdapterProperty"/>
```

```
        <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:element ref="xsf:webPartConnectionAction"/>
        <xsd:element ref="xsf:signSignatureLineAction"/>
      </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>
```

2.2.1.2.115 submitAction

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

Parent Elements
rule

Attributes:

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

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

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

2.2.1.2.116 exitRuleSet

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

Parent Elements
rule

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

```
<xsd:element name="exitRuleSet" minOccurs="0">
  <xsd:complexType/>
</xsd:element>
```

2.2.1.2.117 dialogBoxMessageAction

The **dialogBoxMessageAction** element MUST be ignored.

Parent Elements
rule

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

```
<xsd:element name="dialogBoxMessageAction">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="1024"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

2.2.1.2.118 dialogBoxExpressionAction

The **dialogBoxExpressionAction** element MUST be ignored.

Parent Elements
rule

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

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

2.2.1.2.119 switchViewAction

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

Parent Elements
rule

Attributes:

view: This attribute specifies the name of the form view that is shown. The specified name MUST match an existing **name** attribute of the **view** element, as specified in section 2.2.1.2.104.

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>
```

2.2.1.2.120 assignmentAction

The **assignmentAction** element specifies an action that MUST set the value of a field (3).

Parent Elements
rule

Attributes:

expression: This attribute specifies an XPath expression to populate the value of the **targetField** attribute.

targetField: This attribute specifies an XPath expression that MUST evaluate to the target XML node.

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

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

2.2.1.2.121 changeAdapterProperty

The **changeAdapterProperty** element specifies an action that MUST set the value of the data source (2) of a data adapter.

Parent Elements
rule

Attributes:

adapter : This attribute specifies the name of the data adapter that MUST change its data source (2). The referenced data adapter MUST be a **REST** data adapter.

adapterProperty : This attribute MUST be ignored.

expression : This attribute specifies an XPath expression to populate the data source (2) of the **adapter** attribute.

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

```
<xsd:element name="changeAdapterProperty">
  <xsd:complexType>
    <xsd:attribute name="adapter" type="xsd:string" use="required"/>
    <xsd:attribute name="adapterProperty" type="xsd:string" use="required"/>
    <xsd:attribute name="expression" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.1.2.122 queryAction

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

Parent Elements
query
rule

Attributes:

adapter: This attribute specifies the name of the data adapter that MUST query its data source (2).

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

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

2.2.1.2.123 openNewDocumentAction

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

Parent Elements
rule

Attributes:

solutionURI: This attribute MUST be ignored.

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

```
<xsd:element name="openNewDocumentAction">
  <xsd:complexType>
    <xsd:attribute name="solutionURI" type="xsd:anyURI" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.1.2.124 closeDocumentAction

The **closeDocumentAction** element specifies an action to close the form (1).

Parent Elements
rule

Attributes:

promptToSaveChanges: This attribute MUST be ignored.

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

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

2.2.1.2.125 ruleSet

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

Parent Elements
ruleSets

Child Elements
rule

Attributes:

name: This attribute specifies the name of the set of rules (1).

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

```
<xsd:element name="ruleSet">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:rule" minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.1.2.126 ruleSets

The **ruleSets** element specifies the rule sets for the form (1).

Parent Elements
xDocumentClass

Child Elements
ruleSet

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

```
<xsd:element name="ruleSets">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:ruleSet" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:unique name="ruleSets_name_unique">
    <xsd:selector xpath="./xsf:ruleSet"/>
    <xsd:field xpath="@name"/>
  </xsd:unique>
</xsd:element>
```

2.2.1.2.127 calculations

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

Parent Elements
xDocumentClass

Child Elements

Child Elements
calculatedField

Attributes:

treatBlankValueAsZero: This attribute specifies whether an empty string (1) is equivalent to the integer zero (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>
```

2.2.1.2.128 calculatedField

The **calculatedField** element specifies an individual calculation, when the calculation is to be performed, and where there the result is stored.

Parent Elements
calculations

Attributes:

expression: This attribute specifies the formula, as an XPath expression, to be evaluated. The result MUST be stored in the **target** attribute.

refresh: This attribute specifies when the expression MUST be evaluated. The value MUST be one of the following values:

- **onInit:** The value is evaluated when the node is initialized
- **onChange:** The value is evaluated when a parameter of the expression changes.

target: This attribute specifies the XPath expression location where the result of evaluating the **expression** attribute MUST be stored.

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

```
<xsd:element name="calculatedField">
  <xsd:complexType>
    <xsd:attribute name="target" type="xsd:string" use="required"/>
    <xsd:attribute name="expression" type="xsd:string" use="required"/>
    <xsd:attribute name="refresh" type="xsd:string" use="required"/>
```

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

2.2.1.2.129 bdcAdapter

The **bdcAdapter** element specifies the properties of a data adapter that MUST be created to query and submit data from and to an **external content type**.

Parent Elements
query

Attributes:

entityName : This attribute specifies the name of the external content type. Its length MUST be greater than or equal to 1 and less than or equal to 255.

entityNamespace : This attribute specifies the namespace of an external content type. Its length MUST be greater than or equal to 1 and less than or equal to 255.

entitySchemaVersion : This attribute specifies the version of the external content type. It MUST conform to the following **ABNF**:

```
EntitySchemaVersion = 1*(DIGIT) "." 1*(DIGIT) "." 1*(DIGIT) "." 1*(DIGIT)
```

lobSystemInstance : This attribute specifies the name of the **LobSystemInstance**. Its length MUST be greater than or equal to 1 and less than or equal to 255.

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 the list (1) for data. The value for this attribute MUST be "yes". If this attribute is not present, its value MUST be interpreted as "yes".

specificFinder : This attribute specifies the name of the **SpecificFinder** method. Its length MUST be greater than or equal to 1 and less than or equal to 255.

submitAdapterName : This attribute specifies the name of the data adapter that is used for submitting data.

submitAllowed : This attribute specifies whether the data adapter is allowed to submit data to the list (1). 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="bdcAdapter">
  <xsd:complexType>
    <xsd:attribute name="lobSystemInstance" type="xsd:string" use="optional"/>
    <xsd:attribute name="entityNamespace" type="xsd:string" use="required"/>
    <xsd:attribute name="entityName" type="xsd:string" use="required"/>
```

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

2.2.1.2.130 grooveAdapter

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

Child Elements
field

Attributes:

infopathGroup : This attribute MUST NOT be present.

name : This attribute MUST NOT be present.

queryAllowed : This attribute MUST NOT be present.

queryThisFormOnly : This attribute MUST NOT be present.

spaceBindableUrl : This attribute MUST NOT be present.

spaceCanonicalUrl : This attribute MUST NOT be present.

spaceCanonicalUrlFormattedForSandboxing : This attribute MUST NOT be present.

spaceName : This attribute MUST NOT be present.

submitAllowed : This attribute MUST NOT be present.

toolBindableUrl : This attribute MUST NOT be present.

toolCanonicalUrl : This attribute MUST NOT be present.

toolDisplayName : This attribute MUST NOT be present.

toolName : This attribute MUST NOT be present.

viewDisplayName : This attribute MUST NOT be present.

viewName : 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="grooveAdapter">
    <xsd:complexType>
      <xsd:sequence>
```

```
      <xsd:element name="field" minOccurs="0" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:attribute name="name" type="xsd:string" use="required"/>
          <xsd:attribute name="displayName" type="xsd:string" use="optional"/>
          <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="spaceName" type="xsd:string" use="required"/>
    <xsd:attribute name="spaceBindableUrl" type="xsd:string" use="required"/>
    <xsd:attribute name="spaceCanonicalUrl" type="xsd:string" use="required"/>
    <xsd:attribute name="spaceCanonicalUrlFormattedForSandboxing" type="xsd:string"
use="required"/>
    <xsd:attribute name="toolName" type="xsd:string" use="required"/>
    <xsd:attribute name="toolDisplayName" type="xsd:string" use="required"/>
    <xsd:attribute name="toolBindableUrl" type="xsd:string" use="required"/>
    <xsd:attribute name="toolCanonicalUrl" type="xsd:string" use="required"/>
    <xsd:attribute name="viewName" type="xsd:string" use="required"/>
    <xsd:attribute name="viewDisplayName" type="xsd:string" use="required"/>
    <xsd:attribute name="infopathGroup" type="xsd:string" use="required"/>
    <xsd:attribute name="queryAllowed" type="xsf:xdYesNo" use="optional" default="yes"/>
    <xsd:attribute name="submitAllowed" type="xsf:xdYesNo" use="optional" default="no"/>
    <xsd:attribute name="queryThisFormOnly" type="xsf:xdYesNo" use="optional" default="no"/>
  </xsd:complexType>
</xsd:element>
```

2.2.1.2.131 field

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

Parent Elements
grooveAdapter

Attributes:

displayName : This attribute MUST NOT be present.

infopathName : This attribute MUST NOT be present.

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

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

2.2.1.2.132 sharepointListAdapterRW

The **sharepointListAdapterRW** element specifies the properties of a data adapter that MUST be created to query and submit data from and to a list (1). The data adapter MUST NOT support submitting data to the list (1) as a secondary data source (2).

Parent Elements
query

Child Elements
field

Attributes:

autogen : This attribute specifies whether this data adapter is associated with a lookup field defined in the main data source. If this attribute is not present, its value MUST be interpreted as "no".

contentTypeID : This attribute specifies the **content type identifier**, as specified in [MS-WSSTS] section 2.1.2.8.1, of the **content type** associated with the data adapter. This attribute MUST be set to an empty string (1) for a secondary data source (2).

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 the list (1) for data. The value for this attribute MUST be set to "yes". If this attribute is not present, its value MUST be interpreted as "yes".

queryOneItemOnly : This attribute specifies whether the main data source is allowed to query one or multiple **list items**. The value for this attribute MUST NOT be present for a secondary data source (2). If this attribute is not present, its value MUST be interpreted as "no".

relativeListUrl : This attribute specifies the URL of the list (1). If the **siteURL** attribute is an absolute URL, the value for this attribute MUST be relative to **siteURL**. If the **siteURL** attribute is a server-relative URL, the value for this attribute MUST also be a server-relative URL. Otherwise it MUST be relative to the form template's location.

sharePointListChoices : This attribute specifies the name of the secondary data source (2) associated with this data adapter. The value for this attribute MUST be the name of an **xmlFileAdapter** element, as specified in section 2.2.1.2.26, that exists in the form template. This attribute MUST NOT be present for a secondary data source (2). This attribute MUST be present only if this data adapter contains a **field** element, as specified in section 2.2.1.2.133, that specifies a **type** value set to "Choice" or "MultiChoice".

sharePointListID : This attribute specifies the **list identifier**, as specified in [MS-WSSTS] section 2.1.2.7, of the list (1) associated with the data adapter.

siteURL : This attribute specifies the URL of the parent site (2). It MUST be an absolute URL, a server-relative URL, or a URL relative to the form template's location.

sortAscending : This attribute specifies the value to use for the **Ascending** attribute of the **OrderBy** element, as specified in [MS-WSSCAML] section 2.2.2.1.3, to query the list (1) for data. This attribute MUST NOT be present for a main data source or for a secondary data source (2) with the **autogen** attribute set to "yes". If this attribute is not present, its value MUST be interpreted as "no".

sortBy : This attribute specifies the value to use for the **SortBy** attribute of the **OrderBy** element, as specified in [MS-WSSCAML] section 2.2.2.1.3, to query the list (1) for data. This attribute MUST NOT be present for a main data source or for a secondary data source (2) with the **autogen** attribute set to "yes".

submitAllowed : This attribute specifies whether the data adapter is allowed to submit data to the list (1). This attribute MUST be set to "yes" for a main data source and MUST be set to "no" for a secondary data source (2). If this attribute is not present, its value MUST be interpreted as "no".

version : This attribute specifies the version of the content type against which this main data connection was created. This attribute MUST NOT be present for a secondary data source (2).

This attribute MUST be a **hash** of the **field property summary** generated using the **SHA-1 hash** algorithm. The **field property summary** is a delimited list (1) of all properties for every field (3) defined in the content type. It has the following structure:

- "*<Field1Properties>*\r\n*<Field2Properties>*\r\n...*<FieldNProperties>*\r\n" where *n* is the number of fields (3) defined in the content type.

Each *<FieldNProperties>* string MUST follow this structure:

- "*<FieldAttributesNameValuePairs>**<FieldElementsNameValuePairs>*\r\n"

The *FieldAttributesNameValuePairs* structure MUST be a list (1) of **strings**, one for each attribute, which MUST follow this structure:

- "*<FieldDefinitionAttributeName>:<FieldDefinitionAttributeValue>*\r\n"

Each *FieldDefinitionAttributeName/FieldDefinitionAttributeValue* pair MUST be constructed using the ordered set of **field definition** attributes in the following table.

Field Definition Attribute Name	Field Definition Attribute Value
Name	Value set for the field's **Name** attribute in the content type, as specified in [MS-WSSFO2] section 2.2.8.3.3.1.
Type	Value set for the field's **Type** attribute in the content type, as specified in [MS-WSSFO2] section 2.2.8.3.3.1.
MaxLength	Value set for the field's **MaxLength** attribute in the content type, as specified in [MS-WSSFO2] section 2.2.8.3.3.1.
Min	Value set for the field's **Min** attribute in the content type, as specified in [MS-WSSFO2] section 2.2.8.3.3.1.
Max	Value set for the field's **Max** attribute in the content type, as specified in

Field Definition Attribute Name	Field Definition Attribute Value
	[MS-WSSFO2] section 2.2.8.3.3.1.
Required	Value set for the field's **Required** attribute in the content type, as specified in [MS-WSSFO2] section 2.2.8.3.3.1.
RichText	Value set for the field's **RichText** attribute in the content type, as specified in [MS-WSSFO2] section 2.2.8.3.3.1.
RichTextMode	Value set for the field's **RichTextMode** attribute in the content type, as specified in [MS-WSSFO2] section 2.2.8.3.3.1.
List	Value set for the field's **List** attribute in the content type, as specified in [MS-WSSFO2] section 2.2.8.3.3.1.
ShowField	Value set for the field's **ShowField** attribute in the content type, as specified in [MS-WSSFO2] section 2.2.8.3.3.1.
UserSelectionScope	Value set for the field's **UserSelectionScope** attribute in the content type, as specified in [MS-WSSFO2] section 2.2.8.3.3.1.
FillInChoice	Value set for the field's **FillInChoice** attribute in the content type, as specified in [MS-WSSFO2] section 2.2.8.3.3.1.
Format	Value set for the field's **Format** attribute in the content type, as specified in [MS-WSSFO2] section 2.2.8.3.3.1.
Decimals	Value set for the field's **Decimals** attribute in the content type, as specified in [MS-WSSFO2] section 2.2.8.3.3.1.
Percentage	Value set for the field's **Percentage** attribute in the content type, as specified in [MS-WSSFO2] section 2.2.8.3.3.1.
LCID	Value set for the field's **LCID** attribute in the content type, as specified in [MS-WSSFO2] section 2.2.8.3.3.1.
ReadOnly	Value set for the field's **ReadOnly** attribute in the content type, as specified in [MS-WSSFO2] section 2.2.8.3.3.1.
Format	Value set for the field's **Format** attribute in the content type, as specified in [MS-WSSFO2] section 2.2.8.3.3.1.
AppendOnly	Value set for the field's **AppendOnly** attribute in the content type, as specified in [MS-WSSFO2] section 2.2.8.3.3.1.

Any field definition attributes, as specified in the preceding table, that are not present in the content type **CAML** for a field (3) MUST NOT be included in the *FieldAttributesNameValuePairs* structure.

The *FieldElementsNameValuePairs* structure MUST be a "\r\n" separated list constructed using the ordered set of field definition elements in the following table.

Field Definition Element Name	Field Definition Element Value
CHOICES	The field's choices XML node element in the content type CAML, as specified in [MS-WSSFO2] section 2.2.8.3.3.1.

Field Definition Element Name	Field Definition Element Value
	The **string** value for this element MUST be set to a concatenation of **strings**. Each **string** MUST be set to the text within every child XML node with leading and trailing white space characters removed.
Default	Specifies the field's default XML node element in the content type CAML, as specified in [MS-WSSFO2] section 2.2.8.3.3.1. The **string** value for this element MUST be set to a concatenation of **strings**. Each **string** MUST be set to the text within every child XML node with leading and trailing white space characters removed.

The field definition elements specified in the preceding table that are not present in the content type CAML for a field (3) MUST NOT be included in the *FieldElementsNameValuePairs* structure.

The *FieldElementsNameValuePairs* **string** value MUST NOT contain leading or trailing white space characters.

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

```
<xsd:element name="sharepointListAdapterRW">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="field" minOccurs="0" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:attribute name="internalName" type="xsd:string" use="required"/>
          <xsd:attribute name="hiddenFieldName" type="xsd:string" use="optional"/>
          <xsd:attribute name="type" type="xsd:string" use="required"/>
          <xsd:attribute name="auxDomName" type="xsd:string" use="optional"/>
          <xsd:attribute name="showFieldName" type="xsd:string" use="optional"/>
          <xsd:attribute name="required" type="xsf:xdYesNo" use="optional"/>
          <xsd:attribute name="appendOnly" 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="sharePointListID" type="xsd:string" use="required"/>
    <xsd:attribute name="contentTypeID" 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="autogen" type="xsf:xdYesNo" use="optional"/>
    <xsd:attribute name="relativeListUrl" type="xsd:string" use="optional"/>
    <xsd:attribute name="version" type="xsd:string" use="optional"/>
    <xsd:attribute name="sharePointListChoices" type="xsd:string" use="optional"/>
    <xsd:attribute name="queryOneItemOnly" type="xsf:xdYesNo" use="optional"/>
    <xsd:attribute name="sortBy" type="xsd:string" use="optional"/>
    <xsd:attribute name="sortAscending" type="xsf:xdYesNo" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.1.2.133 field

The **field** element specifies mapping information for a field (3) that is used by the list data adapter to query or submit data to and from a list (1).

Parent Elements
sharepointListAdapterRW

Attributes:

appendOnly : This attribute specifies whether the field (3) is an append-only field (3), as specified in [MS-WSSFO2] section 2.2.8.3.3.1. If this attribute is not present, its value MUST be interpreted as "no".

auxDomName : This attribute specifies the name of the data adapter associated with a lookup field. The value for this attribute MUST be the name of a secondary list data adapter that exists in the form template. This attribute MUST be present for a field (3) that specifies a **type** value set to "Lookup" or "LookupMulti".

hiddenFieldName : This attribute MUST be ignored.

internalName : This attribute specifies the field's (3) internal name.

required : This attribute specifies whether the list's field (3) is required, as specified in [MS-WSSFO2] section 2.2.8.3.3.1. If this attribute is not present, its value MUST be interpreted as "no".

showFieldName : This attribute specifies the name of the show field, as specified in [MS-WSSFO2] section 2.2.8.3.3, associated with a lookup field. This attribute MUST be present for a field (3) that specifies a **type** value set to "Lookup" or "LookupMulti".

type : This attribute specifies the type for the field (3). The value for this attribute MUST be one listed in the following table.

Field Type	Description
Counter	As specified in [MS-WSSTS] section 2.3.1.
Integer	As specified in [MS-WSSTS] section 2.3.1. This attribute MUST NOT be present for a main data source.
Number	As specified in [MS-WSSTS] section 2.3.1.
Currency	As specified in [MS-WSSTS] section 2.3.1.
Text	As specified in [MS-WSSTS] section 2.3.1.
Choice	As specified in [MS-WSSTS] section 2.3.1.
Plain	Specifies a **Note** field (3), as specified in [MS-WSSTS] section 2.3.1, with no formatted text.
Compatible	Specifies a **Note** field (3), as specified in [MS-WSSTS] section 2.3.1, with compatible formatting, as specified in [MS-WSSFO2] section 2.2.8.2.5.
FullHTML	Specifies a **Note** field (3), as specified in [MS-WSSTS] section 2.3.1, with full HTML

Field Type	Description
	formatting, as specified in [MS-WSSFO2] section 2.2.8.2.5.
DateTime	As specified in [MS-WSSTS] section 2.3.1.
Boolean	As specified in [MS-WSSTS] section 2.3.1.
Lookup	As specified in [MS-WSSTS] section 2.3.1.
LookupMulti	As specified in [MS-WSSTS] section 2.3.1.
MultiChoice	As specified in [MS-WSSTS] section 2.3.1.
URL	As specified in [MS-WSSTS] section 2.3.1.
User	As specified in [MS-WSSTS] section 2.3.1.
UserMulti	As specified in [MS-WSSTS] section 2.3.1.
Calculated	As specified in [MS-WSSTS] section 2.3.1.
Attachments	As specified in [MS-WSSTS] section 2.3.1.
HybridUser	Specifies an aggregation of fields (3). This field (3) type value MUST only be set for a form template with the **solutionMode** attribute, as specified in section 2.2.3.2.7, set to "workflowInitAssoc".

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="internalName" type="xsd:string" use="required"/>
    <xsd:attribute name="hiddenFieldName" type="xsd:string" use="optional"/>
    <xsd:attribute name="type" type="xsd:string" use="required"/>
    <xsd:attribute name="auxDomName" type="xsd:string" use="optional"/>
    <xsd:attribute name="showFieldName" type="xsd:string" use="optional"/>
    <xsd:attribute name="required" type="xsf:xdYesNo" use="optional"/>
    <xsd:attribute name="appendOnly" type="xsf:xdYesNo" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.1.2.134 webPartConnectionAction

The **webPartConnectionAction** element specifies that the form (1) MUST submit its data to any connected **Web Parts** when called by a form action and the form (1) is a data provider in a **Web Part connection**.

Parent Elements
rule

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

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

2.2.1.2.135 signSignatureLineAction

The **signSignatureLineAction** element MUST be ignored.

Parent Elements
rule

Attributes:

checkHost : This attribute MUST be ignored.

checkHostEnabled : This attribute MUST be ignored.

defaultSignaturePicture : This attribute MUST be ignored.

isExpression : This attribute MUST be ignored.

isSignaturePictureExpression : This attribute MUST be ignored.

matchCriteria : This attribute MUST be ignored.

matchValue : This attribute MUST be ignored.

signaturePictureEnabled : This attribute MUST be ignored.

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

```
<xsd:element name="signSignatureLineAction">
  <xsd:complexType>
    <xsd:attribute name="matchCriteria" type="xsf:xdSignSignatureLineRuleEnum"
use="required"/>
    <xsd:attribute name="matchValue" type="xsd:string" use="required"/>
    <xsd:attribute name="isExpression" type="xsf:xdYesNo" use="required"/>
    <xsd:attribute name="signaturePictureEnabled" type="xsf:xdYesNo" use="required"/>
    <xsd:attribute name="defaultSignaturePicture" type="xsd:string" use="optional"/>
    <xsd:attribute name="isSignaturePictureExpression" type="xsf:xdYesNo" use="required"/>
    <xsd:attribute name="checkHostEnabled" type="xsf:xdYesNo" use="required"/>
    <xsd:attribute name="checkHost" type="xsd:string" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.2 Form Definition File (XSF2) Extension Specification

The following tables list, in alphabetical order, the types and elements used in the XML schema for the XSF2 extensions to the form definition (.xsf) file. The types and elements belong to the **XSF2** namespace (http://schemas.microsoft.com/office/infopath/2006/solutionDefinition/extensions).

The **XSF2** XML schema is an extension of the XML schema for the form definition (.xsf) file specified in section 2.2.1.

The **XSF2:solutionDefinition** element, as specified in section 2.2.2.2.1, MUST be the root element for all XSF2 extensions.

Type	Specified in Section
compatibilityModesType	2.2.2.1.3
emailAttachmentType	2.2.2.1.2
formDescriptionType	2.2.2.1.5
formLocaleType	2.2.2.1.6
managedCodeType	2.2.2.1.7
serverCommandActionType	2.2.2.1.1
solutionType	2.2.2.1.4

Element	Specified in Section
admin	2.2.2.2.13
adoAdapterExtension	2.2.2.2.25
autoUpdatePrompt	2.2.2.2.38
command	2.2.2.2.5
commands	2.2.2.2.4
connectoid	2.2.2.2.23
contentType	2.2.2.2.9
contentTypeTemplate	2.2.2.2.10
dataConnections	2.2.2.2.21
davAdapterExtension	2.2.2.2.24
emailAdapterExtension	2.2.2.2.28
entity	2.2.2.2.51
errorMessage	2.2.2.2.47
exportToPDForXPS	2.2.2.2.49
featureRestrictionsExtension	2.2.2.2.48
fieldExtension	2.2.2.2.20
fieldsExtension	2.2.2.2.19
groove	2.2.2.2.53

Element	Specified in Section
includedView	2.2.2.2.16
includedViews	2.2.2.2.15
inputScope	2.2.2.2.40
inputScopes	2.2.2.2.39
install	2.2.2.2.7
list	2.2.2.2.50
listPropertiesExtension	2.2.2.2.18
mail	2.2.2.2.12
managedCode	2.2.2.2.43
mergedPrintView	2.2.2.2.14
offline	2.2.2.2.17
preview	2.2.2.2.37
relativeQuery	2.2.2.2.27
sendByMail	2.2.2.2.31
server	2.2.2.2.2
share	2.2.2.2.11
sharepointListAdapterExtension	2.2.2.2.30
sharepointListAdapterRWExtension	2.2.2.2.54
solutionDefinition	2.2.2.2.1
solutionPropertiesExtension	2.2.2.2.6
submit	2.2.2.2.44
submitAction	2.2.2.2.45
successMessage	2.2.2.2.46
toolbar	2.2.2.2.3
useHttpHandlerExtension	2.2.2.2.22
viewExtension	2.2.2.2.35
viewsExtension	2.2.2.2.34
warning	2.2.2.2.33
warnings	2.2.2.2.32

Element	Specified in Section
webServiceAdapterExtension	2.2.2.2.26
word	2.2.2.2.42
words	2.2.2.2.41
workflowInitAssoc	2.2.2.2.52
wss	2.2.2.2.8
xmlFileAdapterExtension	2.2.2.2.29
xmlToEditExtension	2.2.2.2.36

2.2.2.1 Form Definition File XSF2 Enumerations

This section specifies the types used by elements and attributes in the **XSF2 namespace** (http://schemas.microsoft.com/office/infopath/2006/solutionDefinition/extensions).

2.2.2.1.1 serverCommandActionType

The **serverCommandActionType** simple type specifies restrictions for specifying a form (1) action on the form (1) **toolbar**.

Referenced By
command.xsf2@action

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

```
<xsd:simpleType name="serverCommandActionType">
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:pattern value="[a-zA-Z0-9_]*"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.2.1.2 emailAttachmentType

The **emailAttachmentType** 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
emailAdapterExtension.xsf2@emailAttachmentType
sendByMail.xsf2@emailAttachmentType

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

```
<xsd:simpleType name="emailAttachmentType">
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:pattern value="[a-zA-Z0-9_]*"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.2.1.3 compatibilityModesType

The **compatibilityModesType** simple type specifies restrictions for specifying the compatibility mode for the form template.

Referenced By
solutionDefinition.xsf2@runtimeCompatibility

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

```
<xsd:simpleType name="compatibilityModesType">
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:pattern value="[a-zA-Z0-9_]*"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.2.1.4 solutionType

The **solutionType** simple type specifies restrictions for an attribute that, if present, MUST be ignored.

Referenced By
solutionDefinition.xsf2@solutionType

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

```
<xsd:simpleType name="solutionType">
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:pattern value="[a-zA-Z0-9]*"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.2.1.5 formDescriptionType

The **formDescriptionType** simple type specifies restrictions for specifying the form template description.

Referenced By
solutionDefinition.xsf2@description

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

```
<xsd:simpleType name="formDescriptionType">
  <xsd:restriction base="xsd:string">
    <xsd:maxLength value="1024"/>
    <xsd:minLength value="1"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.2.1.6 formLocaleType

The **formLocaleType** simple type specifies restrictions for specifying the locale of the form template.

Referenced By
server.xsf2@formLocale

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

```
<xsd:simpleType name="formLocaleType">
  <xsd:restriction base="xsd:token">
    <xsd:minLength value="1"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.2.1.7 managedCodeType

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

Referenced By
managedCode.xsf2@language

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

```
<xsd:simpleType name="managedCodeType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[a-zA-Z0-9\.]*"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.2.2 Form Definition File XSF2 Elements

This section specifies the elements and attributes in the **XSF2** namespace (http://schemas.microsoft.com/office/infopath/2006/solutionDefinition/extensions).

2.2.2.2.1 solutionDefinition

The **solutionDefinition** element acts as the container for extensions to the properties and content of the form template.

This element MUST be the root element of all XSF2 extensions, as specified in section 2.2.2.

This element MUST be contained by the **extension** element, as specified in section 2.2.1.2.112.

This element also enables extending the form definition (.xsf) file with custom attributes not specified by this protocol document. Custom attributes MUST NOT be defined under the **XSF**, **XSF2** or **XSF3** namespaces (http://schemas.microsoft.com/office/infopath/2003/solutionDefinition, http://schemas.microsoft.com/office/infopath/2006/solutionDefinition/extensions, http://schemas.microsoft.com/office/infopath/2009/solutionDefinition/extensions respectively). Any custom attribute MUST be ignored.

Child Elements
autoUpdatePrompt
dataConnections
featureRestrictionsExtension
inputScopes
listPropertiesExtension
managedCode
mergedPrintView
offline
preview
sendByMail
server
solutionPropertiesExtension
submit
viewsExtension
warnings

Attributes:

allowClientOnlyCode: This attribute specifies whether a browser-compatible form template is designed to enable the inclusion of client-specific object model code that is not compatible with the protocol server. If this attribute is "yes", a warning MUST be generated when browser-enabling a form template containing incompatible code. If this attribute is "no", an error MUST be generated when browser-enabling a form template containing incompatible code.

description: This attribute is the description of the form template.

runtimeCompatibility: This attribute MUST be set to "client server".

runtimeCompatibilityURL: This attribute MUST be ignored.

solutionType: This attribute MUST be ignored.

verifyOnServer: This attribute MUST be ignored.

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

```
<xsd:element name="solutionDefinition">
  <xsd:complexType>
    <xsd:all>
      <xsd:element ref="xsf2:server" minOccurs="0"/>
      <xsd:element ref="xsf2:solutionPropertiesExtension" minOccurs="0"/>
      <xsd:element ref="xsf2:mergedPrintView" minOccurs="0"/>
      <xsd:element ref="xsf2:offline" minOccurs="0"/>
      <xsd:element ref="xsf2:listPropertiesExtension" minOccurs="0"/>
      <xsd:element ref="xsf2:dataConnections" minOccurs="0"/>
      <xsd:element ref="xsf2:sendByMail" minOccurs="0"/>
      <xsd:element ref="xsf2:warnings" minOccurs="0"/>
      <xsd:element ref="xsf2:viewsExtension" minOccurs="0"/>
      <xsd:element ref="xsf2:preview" minOccurs="0"/>
      <xsd:element ref="xsf2:autoUpdatePrompt" minOccurs="0"/>
      <xsd:element ref="xsf2:inputScopes" minOccurs="0"/>
      <xsd:element ref="xsf2:managedCode" minOccurs="0"/>
      <xsd:element ref="xsf2:submit" minOccurs="0"/>
      <xsd:element ref="xsf2:featureRestrictionsExtension" minOccurs="0"/>
    </xsd:all>
    <xsd:attribute name="runtimeCompatibility" use="required">
      <xsd:simpleType>
        <xsd:list itemType="xsf2:compatibilityModesType"/>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="solutionType" type="xsf2: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>
```

2.2.2.2.2 server

The **server** element specifies display and functional properties for the form template.

Parent Elements
solutionDefinition

Child Elements
toolbar

Attributes:

formLocale<2>**:** 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 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 postbacked, the user MUST be notified that the form file will 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>
```

2.2.2.2.3 toolbar

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

Parent Elements
server

Child Elements
commands

Attributes:

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

enabledTop: 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>
```

2.2.2.2.4 commands

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

Parent Elements
toolbar

Child Elements
command

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

```
<xsd:element name="commands">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:command" maxOccurs="unbounded" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.5 command

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

Parent Elements
commands

Attributes:

action: This attribute specifies an action that MUST be performed when the toolbar button is clicked. The value MUST be one of the following 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>
```

2.2.2.2.6 solutionPropertiesExtension

The **solutionPropertiesExtension** element MUST be ignored.

Parent Elements
solutionDefinition

Child Elements
admin
contentType
contentTypeTemplate
entity
groove
install

Child Elements
list
mail
share
workflowInitAssoc
wss

Attributes:

branch: This attribute MUST be ignored.

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

```
<xsd:element name="solutionPropertiesExtension">
  <xsd:complexType>
    <xsd:all>
      <xsd:element ref="xsf2:install" minOccurs="0"/>
      <xsd:element ref="xsf2:wss" minOccurs="0"/>
      <xsd:element ref="xsf2:contentType" minOccurs="0"/>
      <xsd:element ref="xsf2:share" minOccurs="0"/>
      <xsd:element ref="xsf2:mail" minOccurs="0"/>
      <xsd:element ref="xsf2:admin" minOccurs="0"/>
      <xsd:element ref="xsf2:contentTypeTemplate" minOccurs="0"/>
      <xsd:element ref="xsf2:list" minOccurs="0"/>
      <xsd:element ref="xsf2:entity" minOccurs="0"/>
      <xsd:element ref="xsf2:workflowInitAssoc" minOccurs="0"/>
      <xsd:element ref="xsf2:groove" 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:enumeration value="list"/>
          <xsd:enumeration value="entity"/>
          <xsd:enumeration value="workflowInitAssoc"/>
          <xsd:enumeration value="groove"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.7 install

The **install** element MUST be ignored.

Parent Elements
solutionPropertiesExtension

Attributes:

companyName: This attribute MUST be ignored.

language: This attribute MUST be ignored.

path: This attribute MUST be ignored.

updatePath: This attribute MUST be ignored.

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

```
<xsd:element name="install">
  <xsd:complexType>
    <xsd:attribute name="companyName" type="xsd:string" use="required"/>
    <xsd:attribute name="language" type="xsd:string" use="required"/>
    <xsd:attribute name="path" type="xsd:string" use="required"/>
    <xsd:attribute name="updatePath" type="xsd:string" use="optional"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.8 wss

The **wss** element MUST be ignored.

Parent Elements
solutionPropertiesExtension

Attributes:

browserEnable: This attribute MUST be ignored.

description: This attribute MUST be ignored.

name: This attribute MUST be ignored.

path: This attribute MUST be ignored.

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

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

```
      <xsd:sequence/>
      <xsd:attribute name="path" type="xsd:string" use="required"/>
      <xsd:attribute name="name" type="xsd:string" use="required"/>
      <xsd:attribute name="description" type="xsd:string" use="required"/>
      <xsd:attribute name="browserEnable" type="xsf:xdYesNo" use="optional"/>
      <xsd:anyAttribute processContents="skip"/>
    </xsd:complexType>
  </xsd:element>
```

2.2.2.2.9 contentType

The **contentType** element MUST be ignored.

Parent Elements
solutionPropertiesExtension

Attributes:

path: This attribute MUST be ignored.

sharepointContentTypeId: This attribute MUST be ignored.

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

```
<xsd:element name="contentType">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="path" type="xsd:string" use="required"/>
    <xsd:attribute name="sharepointContentTypeId" type="xsd:string" use="required"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.10 contentTypeTemplate

The **contentTypeTemplate** element MUST be ignored.

Parent Elements
solutionPropertiesExtension

Attributes:

browserEnable: This attribute MUST be ignored.

description: This attribute MUST be ignored.

name: This attribute MUST be ignored.

path: This attribute MUST be ignored.

site: This attribute MUST be ignored.

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

```
<xsd:element name="contentTypeTemplate">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="site" type="xsd:string" use="required"/>
    <xsd:attribute name="path" type="xsd:string" use="required"/>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="description" type="xsd:string" use="required"/>
    <xsd:attribute name="browserEnable" type="xsf:xdYesNo" use="optional"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.11 share

The **share** element MUST be ignored.

Parent Elements
solutionPropertiesExtension

Attributes:

accessPath: This attribute MUST be ignored.

formName: This attribute MUST be ignored.

path: This attribute MUST be ignored.

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

```
<xsd:element name="share">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="formName" type="xsd:string" use="required"/>
    <xsd:attribute name="path" type="xsd:string" use="required"/>
    <xsd:attribute name="accessPath" type="xsd:string" use="required"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.12 mail

The **mail** element MUST be ignored.

Parent Elements
solutionPropertiesExtension

Attributes:

formName: This attribute MUST be ignored.

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

```
<xsd:element name="mail">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="formName" type="xsd:string" use="required"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.13 admin

The **admin** element MUST be ignored.

Parent Elements
solutionPropertiesExtension

Attributes:

path: This attribute MUST be ignored.

site: This attribute MUST be ignored.

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

```
<xsd:element name="admin">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="path" type="xsd:string" use="required"/>
    <xsd:attribute name="site" type="xsd:string" use="required"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.14 mergedPrintView

The **mergedPrintView** element MUST be ignored.

Parent Elements
solutionDefinition

Child Elements

Child Elements
includedViews
printSettings

Attributes:

isCustomizable: This attribute MUST be ignored.

isDefault: This attribute MUST be ignored.

viewBreak: This attribute MUST be ignored.

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

```
<xsd:element name="mergedPrintView">
  <xsd:complexType>
    <xsd:all>
      <xsd:element ref="xsf:printSettings" minOccurs="0"/>
      <xsd:element ref="xsf2:includedViews" minOccurs="0"/>
    </xsd:all>
    <xsd:attribute name="isDefault" type="xsf:xdYesNo" use="optional" default="no"/>
    <xsd:attribute name="isCustomizable" type="xsf:xdYesNo" use="optional" default="no"/>
    <xsd:attribute name="viewBreak" type="xsd:string" use="required"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.15 includedViews

The **includedViews** element MUST be ignored.

Parent Elements
mergedPrintView

Child Elements
includedView

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

```
<xsd:element name="includedViews">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:includedView" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
```

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

2.2.2.2.16 includedView

The **includedView** element MUST be ignored.

Parent Elements
includedViews

Attributes:

name: This attribute MUST be ignored.

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

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

2.2.2.2.17 offline

The **offline** element MUST be ignored.

Parent Elements
solutionDefinition

Attributes:

cacheQueries: This attribute MUST be ignored.

expirationTime: This attribute MUST be ignored.

openIfQueryFails: This attribute MUST be ignored.

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

```
<xsd:element name="offline">
  <xsd:complexType>
    <xsd:attribute name="openIfQueryFails" type="xsf:xdYesNo" default="no" use="optional"/>
    <xsd:attribute name="cacheQueries" type="xsf:xdYesNo" default="no" use="optional"/>
    <xsd:attribute name="expirationTime" type="xsd:nonNegativeInteger" use="optional"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.18 listPropertiesExtension

The **listPropertiesExtension** element MUST be ignored.

Parent Elements
solutionDefinition

Child Elements
fieldsExtension

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

```
<xsd:element name="listPropertiesExtension">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:fieldsExtension" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.19 fieldsExtension

The **fieldsExtension** element MUST be ignored.

Parent Elements
listPropertiesExtension

Child Elements
fieldExtension

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

```
<xsd:element name="fieldsExtension">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:fieldExtension" maxOccurs="unbounded" minOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.20 fieldExtension

The **fieldExtension** element MUST be ignored.

Parent Elements
fieldsExtension

Attributes:

columnId: This attribute MUST be ignored.

columnName: This attribute MUST be ignored.

readWrite: This attribute MUST be ignored.

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

```
<xsd:element name="fieldExtension">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="columnName" type="xsd:string" use="required"/>
    <xsd:attribute name="readWrite" type="xsf:xdYesNo" use="optional" default="no"/>
    <xsd:attribute name="columnId" type="xsd:string" use="optional"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.21 dataConnections

The **dataConnections** element contains elements that specify extensions to data adapter connection settings.

Parent Elements
solutionDefinition

Child Elements
adoAdapterExtension
davAdapterExtension
emailAdapterExtension
sharepointListAdapterExtension
sharepointListAdapterRWExtension
useHttpHandlerExtension

Child Elements
webServiceAdapterExtension
xmlFileAdapterExtension

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

```
<xsd:element name="dataConnections">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:useHttpHandlerExtension" minOccurs="0"/>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element ref="xsf2:davAdapterExtension" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="xsf2:adoAdapterExtension" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="xsf2:webServiceAdapterExtension" minOccurs="0"
maxOccurs="unbounded"/>
        <xsd:element ref="xsf2:emailAdapterExtension" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="xsf2:xmlFileAdapterExtension" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="xsf2:sharepointListAdapterExtension" minOccurs="0"
maxOccurs="unbounded"/>
        <xsd:element ref="xsf2:sharepointListAdapterRWExtension" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:choice>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.22 useHttpHandlerExtension

The **useHttpHandlerExtension** element specifies extended information for the **useHttpHandler** element, as specified in section 2.2.1.2.57.

Parent Elements
dataConnections

Child Elements
connectoid

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

```
<xsd:element name="useHttpHandlerExtension">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:connectoid" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
```

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

2.2.2.2.23 connectoid

The **connectoid** element specifies the location of a **Universal Data Connection (.udc, .udcx) file**, as specified by [MS-UDCX], containing data adapter connection settings that MUST override the connection settings specified in the form definition (.xsf) file. A 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 specified in section 2.2.1.2.1, 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
adoAdapterExtension
davAdapterExtension
sharepointListAdapterExtension
sharepointListAdapterRWExtension
useHttpHandlerExtension
webServiceAdapterExtension
xmlFileAdapterExtension

Attributes:

connectionLinkType: This attribute specifies the location context of the Universal Data Connection (.udc, .udcx) file. The value MUST be one of the following:

- **relative:** The file is located in the current **site collection**.
- **store:** The file is located in the global **data connection library**.

name: This attribute MUST be ignored.

siteCollection: This attribute specifies the URL to the root of the site collection where the original Universal Data Connection (.udc, .udcx) file is located. It MUST be either an absolute URL or a server-relative URL or relative to the form template's location. This value MUST be identical for all **connectoid** elements in the form template.

source: This attribute specifies a URL that specifies the relative path from the **siteCollection** attribute to the 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>
```

2.2.2.2.24 davAdapterExtension

The **davAdapterExtension** element specifies the extended information for the **davAdapter** element, as specified in section 2.2.1.2.29.

Parent Elements
dataConnections

Child Elements
connectoid

Attributes:

ref: This attribute specifies the associated **davAdapter** element that is being extended. The specified value SHOULD match the value specified by the **name** attribute of the corresponding **davAdapter** element.

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

```
<xsd:element name="davAdapterExtension">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:connectoid" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="ref" type="xsf:xdTitle" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.25 adoAdapterExtension

The **adoAdapterExtension** element specifies extended information for the **adoAdapter** element, as specified in section 2.2.1.2.19.

Parent Elements
dataConnections

Child Elements
connectoid

Attributes:

queryFile: This attribute MUST be ignored.

queryKey: This attribute MUST be ignored.

ref: This attribute specifies the associated **adoAdapter** element that is being extended. The specified value SHOULD match the value specified by the **name** attribute of the corresponding **adoAdapter** element.

submitAdapterName: This attribute MUST be ignored.

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

```
<xsd:element name="adoAdapterExtension">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:connectoid" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="ref" type="xsf:xdTitle" use="required"/>
    <xsd:attribute name="submitAdapterName" type="xsf:xdTitle" use="optional"/>
    <xsd:attributeGroup ref="xsf2:queryKeyFile"/>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.26 webServiceAdapterExtension

The **webServiceAdapterExtension** element specifies extended information for the **webServiceAdapter** element, as specified in section 2.2.1.2.20. This element MUST NOT have both a **connectoid** child element, as specified in section 2.2.2.2.23, and a **relativeQuery** child element, as specified in section 2.2.2.2.27, present.

Parent Elements
dataConnections

Child Elements
connectoid
relativeQuery

Attributes:

queryFile: This attribute MUST be ignored.

queryKey: This attribute MUST be ignored.

ref: This attribute specifies the associated **webServiceAdapter** element that is being extended. The specified value SHOULD match the value specified by the **name** attribute of the corresponding **webServiceAdapter** element.

trackDataSetChanges: This attribute MUST be "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>
```

2.2.2.2.27 relativeQuery

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

Parent Elements
webServiceAdapterExtension

Attributes:

replace: This attribute specifies the **substring** of the Web service URL that is replaced at run time. The specified URL MUST be an absolute path and MUST NOT be a local or **Universal Naming Convention (UNC)** path. The specified URL MUST match the beginning of the value specified by the **serviceUrl** attribute of the **operation** element, as specified in section 2.2.1.2.22.

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>
```

2.2.2.2.28 emailAdapterExtension

The **emailAdapterExtension** element specifies the method of submitting the form file using the e-mail data adapter.

Parent Elements
dataConnections

Attributes:

emailAttachmentType: This attribute specifies the file type an attachment MUST be sent as when the form file is submitted using the e-mail data adapter. This attribute MUST be set to one of the following values:

- **none:** The form file is sent in the body of the e-mail.
- **xml:** The form file is attached to the e-mail as an XML file.
- **xmlXsn:** 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, as specified in section 2.2.1.2.32, 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>
```

2.2.2.2.29 xmlFileAdapterExtension

The **xmlFileAdapterExtension** element specifies extended information for the **xmlFileAdapter** element, as specified in section 2.2.1.2.26.

Parent Elements
dataConnections

Child Elements
connectoid

Attributes:

isRest : This attribute specifies whether the associated **xmlFileAdapter** is a REST data adapter. If this attribute is not present, its value MUST be interpreted as "no".

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:attribute name="isRest" type="xsf:xdYesNo" use="optional"/>
    <xsd:attributeGroup ref="xsf2:queryKeyFile"/>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.30 sharepointListAdapterExtension

The **sharepointListAdapterExtension** element specifies extended information for the **sharepointListAdapter** element, as specified in section 2.2.1.2.27.

Parent Elements
dataConnections

Child Elements
connectoid

Attributes:

queryFile: This attribute MUST be ignored.

queryKey: This attribute MUST be ignored.

queryThisFormOnly: This attribute specifies whether the list (1) data adapter MUST query the 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: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>
```

2.2.2.2.31 sendByMail

The **sendByMail** element specifies whether the form file or form template is attached to the e-mail generated by the e-mail data adapter as a control-specific **MIME type**.

Parent Elements
solutionDefinition

Attributes:

disableEmailForms: This attribute specifies the MIME type of the attached form file or form template. This attribute must have one of the following values:

- **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>
```

2.2.2.2.32 warnings

The **warnings** element MUST be ignored.

Parent Elements
solutionDefinition

Child Elements
warning

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

```
<xsd:element name="warnings">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:warning" maxOccurs="unbounded" minOccurs="0"/>
    </xsd:sequence>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.33 warning

The **warning** element MUST be ignored.

Parent Elements
warnings

Attributes:

hidden: This attribute MUST be ignored.

source: This attribute MUST be ignored.

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

```
<xsd:element name="warning">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="source" type="xsd:string" use="required"/>
    <xsd:attribute name="hidden" type="xsf:xdYesNo" use="optional" default="no"/>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.34 viewsExtension

The **viewsExtension** element contains extended information for the form views in this form template.

Parent Elements
solutionDefinition

Child Elements
viewExtension

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

```
<xsd:element name="viewsExtension">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:viewExtension" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.35 viewExtension

The **viewExtension** element specifies extended information for the **view** element, as specified in section 2.2.1.2.104.

Parent Elements
viewsExtension

Child Elements
xmlToEditExtension

Attributes:

clientOnly: This attribute specifies whether the form view contains features that are not compatible with the protocol server. If this attribute is "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 **name** attribute of the corresponding **view** element.

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

```
<xsd:element name="viewExtension">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:xmlToEditExtension" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="ref" type="xsf:xdTitle" use="required"/>
    <xsd:attribute name="designMode" type="xsd:string" use="optional"/>
    <xsd:attribute name="readOnly" type="xsf:xdYesNo" use="optional"/>
    <xsd:attribute name="clientOnly" type="xsf:xdYesNo" use="optional" default="no"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.36 xmlToEditExtension

The **xmlToEditExtension** element specifies extended information for the **xmlToEdit** element, as specified in section 2.2.1.2.105.

Parent Elements
viewExtension

Attributes:

allowLinkedImages: This attribute specifies whether hyperlink references to images are allowed in a rich text box control. This attribute MUST be "yes" if the corresponding **xmlToEdit** element refers to a rich text box control.

excludeEmbeddedImages: This attribute specifies whether embedded images are excluded in a rich text box control. This attribute MUST be "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 **name** attribute of the corresponding **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>
```

2.2.2.2.37 preview

The **preview** element MUST be ignored.

Parent Elements
solutionDefinition

Attributes:

domain: This attribute MUST be ignored.

sampleData: This attribute MUST be ignored.

userRole: This attribute MUST be ignored.

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

```
<xsd:element name="preview">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="sampleData" type="xsd:string" use="optional"/>
    <xsd:attribute name="domain" type="xsd:string" use="optional"/>
    <xsd:attribute name="userRole" type="xsd:string" use="optional"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.38 autoUpdatePrompt

The **autoUpdatePrompt** element MUST be ignored.

Parent Elements
solutionDefinition

Attributes:

showPrompt: This attribute MUST be ignored.

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

```
<xsd:element name="autoUpdatePrompt">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="showPrompt" type="xsf:xdYesNo" use="optional"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.39 inputScopes

The **inputScopes** element MUST be ignored.

Parent Elements
solutionDefinition

Child Elements
inputScope

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

```
<xsd:element name="inputScopes">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:inputScope" maxOccurs="unbounded" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.40 inputScope

The **inputScope** element MUST be ignored.

Parent Elements
inputScopes

Child Elements
words

Attributes:

caption: This attribute MUST be ignored.

expression: This attribute MUST be ignored.

name: This attribute MUST be ignored.

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

```
<xsd:element name="inputScope">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf2:words" maxOccurs="unbounded" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="caption" type="xsf:xdTitle" use="optional"/>
    <xsd:attribute name="expression" type="xsd:string" use="optional"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.41 words

The **words** element MUST be ignored.

Parent Elements
inputScope

Child Elements
word

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

```
<xsd:element name="words">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="word" maxOccurs="unbounded" minOccurs="1">
        <xsd:complexType>
          <xsd:sequence/>
          <xsd:attribute name="value" type="xsd:string" use="optional" default=""/>
          <xsd:anyAttribute processContents="skip"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.42 word

The **word** element MUST be ignored.

Parent Elements
words

Attributes:

value: This attribute MUST be ignored.

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

```
<xsd:element name="word" maxOccurs="unbounded" minOccurs="1">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="value" type="xsd:string" use="optional" default=""/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.43 managedCode

The **managedCode** element specifies settings for business objects in the form template (.xsn) file. Business objects are loaded when a form template (.xsn) file is published. Platform specifics determine the success of loading business objects.

Parent Elements
solutionDefinition

Attributes:

enabled: This attribute MUST be ignored.

language: This attribute MUST be ignored.

projectPath: This attribute MUST be ignored.

version: This attribute specifies which version of the platform with which the business objects were compiled. The specified value MUST match a supported version of the platform installed on the protocol server.

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

```
<xsd:element name="managedCode">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="projectPath" type="xsd:string" use="optional"/>
    <xsd:attribute name="language" type="xsf2: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>
```

2.2.2.2.44 submit

The **submit** element MUST be ignored.

Parent Elements
solutionDefinition

Child Elements
errorMessage
submitAction
successMessage

Attributes:

caption: This attribute MUST be ignored.

disableMenuItem: This attribute MUST be ignored.

onAfterSubmit: This attribute MUST be ignored.

showSignatureReminder: This attribute MUST be ignored.

showStatusDialog: This attribute MUST be ignored.

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

```
  <xsd:element name="submit">
    <xsd:complexType>
      <xsd:all>
        <xsd:element name="submitAction" minOccurs="0">
          <xsd:complexType>
            <xsd:sequence/>
            <xsd:attribute name="adapter" type="xsf:xdTitle" use="required"/>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="successMessage" type="xsd:string" minOccurs="0"/>
        <xsd:element name="errorMessage" type="xsd:string" minOccurs="0"/>
      </xsd:all>
      <xsd:attribute name="caption" type="xsd:string" use="optional"/>
      <xsd:attribute name="onAfterSubmit" use="optional">
        <xsd:simpleType>
          <xsd:restriction base="xsd:NMTOKEN">
```

```
            <xsd:enumeration value="close"/>
            <xsd:enumeration value="keepOpen"/>
            <xsd:enumeration value="openNew"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="showStatusDialog" type="xsf:xdYesNo" use="optional"/>
      <xsd:attribute name="showSignatureReminder" type="xsf:xdYesNo" use="optional"/>
      <xsd:attribute name="disableMenuItem" type="xsf:xdYesNo" use="optional"/>
    </xsd:complexType>
  </xsd:element>
```

2.2.2.2.45 submitAction

The **submitAction** element MUST be ignored.

Parent Elements
submit

Attributes:

adapter: This attribute MUST be ignored.

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

```
<xsd:element name="submitAction" minOccurs="0">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="adapter" type="xsf:xdTitle" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.46 successMessage

The **successMessage** element MUST be ignored.

Parent Elements
submit

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

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

2.2.2.2.47 errorMessage

The **errorMessage** element MUST be ignored.

Parent Elements
submit

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

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

2.2.2.2.48 featureRestrictionsExtension

The **featureRestrictionsExtension** element MUST be ignored.

Parent Elements
solutionDefinition

Child Elements
exportToPDForXPS

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

```
<xsd:element name="featureRestrictionsExtension">
  <xsd:complexType>
    <xsd:all>
      <xsd:element ref="xsf2:exportToPDForXPS" minOccurs="0"/>
    </xsd:all>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.49 exportToPDForXPS

The **exportToPDForXPS** element MUST be ignored.

Parent Elements
featureRestrictionsExtension

Attributes:

ui: This attribute MUST be ignored.

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

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

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

2.2.2.2.50 list

The **list** element MUST be ignored.

Parent Elements
solutionPropertiesExtension

Attributes:

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="list">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="path" type="xsd:string" use="required"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.51 entity

The **entity** element MUST be ignored.

Parent Elements
solutionPropertiesExtension

Attributes:

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="entity">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="path" type="xsd:string" use="required"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.52 workflowInitAssoc

The **workflowInitAssoc** element MUST be ignored.

Parent Elements
solutionPropertiesExtension

Attributes:

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="workflowInitAssoc">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="path" type="xsd:string" use="required"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.53 groove

The **groove** element MUST be ignored.

Parent Elements
solutionPropertiesExtension

Attributes:

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="groove">
  <xsd:complexType>
    <xsd:sequence/>
    <xsd:attribute name="path" type="xsd:string" use="required"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.2.2.2.54 sharepointListAdapterRWExtension

The **sharepointListAdapterRWExtension** element specifies extended information for the **sharepointListAdapterRW** element, as specified in section 2.2.1.2.132.

Parent Elements

Parent Elements
dataConnections

Child Elements
connectoid

Attributes:

queryFile : This attribute MUST be ignored by the form server.

queryKey : This attribute MUST be ignored by the form server.

queryThisFormOnly : This attribute specifies whether the list data adapter MUST query the list (1) for values applicable only to the current form (1). This attribute MUST be set to "no" for a list main data connection.

ref : This attribute specifies the associated **sharepointListAdapterRW** element that is being extended. The specified value SHOULD match the value specified by the **name** attribute of the corresponding **sharepointListAdapterRW** element.

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

```
<xsd:element name="sharepointListAdapterRWExtension">
  <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:attributeGroup ref="xsf2:queryKeyFile"/>
  </xsd:complexType>
</xsd:element>
```

2.2.3 Form Definition File (XSF3) Extension Specification

The following tables list, in alphabetical order, the types and elements used in the XML schema for the **XSF3** extensions to the form definition (.xsf) file. The types and elements belong to the **XSF3** namespace (http://schemas.microsoft.com/office/infopath/2009/solutionDefinition/extensions).

The **XSF3** XML schema is an extension of the XML schema for the form definition (.xsf) file specified in section 2.2.1.

The **XSF3:solutionDefinition** element, as specified in section 2.2.3.2.1, and the **solutionPropertiesExtension2009** element, as specified in section 2.2.3.2.6, act as the containers all for all **XSF3** extensions.

Type	Specified in Section
xdLineStamp	2.2.3.1.3
xdModeType	2.2.3.1.2
xdParameterType	2.2.3.1.1

Element	Specified in Section
baseUrl	2.2.3.2.2
confirmationMessage	2.2.3.2.13
customValidation	2.2.3.2.17
errorBlank	2.2.3.2.18
signatureLine	2.2.3.2.12
signatureLines	2.2.3.2.11
solutionDefinition	2.2.3.2.1
solutionMode	2.2.3.2.7
solutionPropertiesExtension2009	2.2.3.2.6
suggestedSignerEmailAddress	2.2.3.2.16
suggestedSignerName	2.2.3.2.14
suggestedSignerTitle	2.2.3.2.15
viewExtension	2.2.3.2.9
viewsExtension	2.2.3.2.8
webPartField	2.2.3.2.5
webPartFields	2.2.3.2.4
webPartProperties	2.2.3.2.3
xmlToEditExtension	2.2.3.2.10

2.2.3.1 Form Definition File XSF3 Enumerations

This section specifies the types used by elements and attributes in the **XSF3** namespace (http://schemas.microsoft.com/office/infopath/2009/solutionDefinition/extensions).

2.2.3.1.1 xdParameterType

The **xdParameterType** simple type specifies **enumeration** values for specifying whether the corresponding field (3) can be read from or written to in a Web Part connection.

input : The corresponding field (3) can be read from but not written into.

inputOutput : The corresponding field (3) can be read from or written to.

output : The corresponding field (3) can be written to but not read from.

Referenced By
webPartField.xsf3@parameterType

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

```
<xsd:simpleType name="xdParameterType">
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:enumeration value="input"/>
    <xsd:enumeration value="output"/>
    <xsd:enumeration value="inputOutput"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.3.1.2 xdModeType

The **xdModeType** simple type specifies the intended form-filling scenario for the form template (.xsn) file.

entity : This value specifies that the form template (.xsn) file represents the form (1) used to edit items in an **external list**.

The **bdcAdapter** element, as specified in section 2.2.1.2.129, MUST be present.

groove : This value is associated with a client-only scenario and MUST NOT be present.

list : This value specifies that the form template (.xsn) file represents the form (1) used to edit items in a list (1).

The file attachment control, as specified in section 2.3.1.7 and section 2.4.1.11, MUST NOT be present.

The **sharepointListAdapterRW** element, as specified in section 2.2.1.2.132, MUST be present.

workflowInitAssoc : This value specifies that the form template (.xsn) file represents the form (1) used to edit items in a **workflow (2)**.

The following controls MUST NOT be present:

- Numbered list, as specified in section 2.3.2.7 and section 2.4.1.21.7.
- Bulleted list, as specified in section 2.3.2.7 and section 2.4.1.21.7.
- Plain list, as specified in section 2.3.2.7 and section 2.4.1.21.7.
- Multiple-selection list box, as specified in section 2.3.2.9 and section 2.4.1.21.9.
- Section, as specified in section 2.3.1.14 and section 2.4.1.18.

- Optional section, as specified in section 2.3.1.14 and section 2.4.1.18.

Referenced By
solutionMode.xsf3@mode

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

```
<xsd:simpleType name="xdModeType">
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:enumeration value="entity"/>
    <xsd:enumeration value="groove"/>
    <xsd:enumeration value="list"/>
    <xsd:enumeration value="workflowInitAssoc"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.3.1.3 xdLineStamp

The **xdLineStamp** simple type specifies restrictions for an attribute that MUST be ignored.

line : The use of this value MUST be ignored.

stamp : The use of this value MUST be ignored.

Referenced By
signatureLine.xsf3@signatureType

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

```
<xsd:simpleType name="xdLineStamp">
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:enumeration value="line"/>
    <xsd:enumeration value="stamp"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.2.3.2 Form Definition File XSF3 Elements

This section specifies the types used by elements and attributes in the **XSF3** namespace (http://schemas.microsoft.com/office/infopath/2009/solutionDefinition/extensions).

2.2.3.2.1 solutionDefinition

The **solutionDefinition** element acts as the container for extensions to the properties and content of the form template.

This element MUST be contained by the **extension** element, as specified in section 2.2.1.2.112.

This element also enables extending the form definition (.xsf) file with custom attributes not specified by this protocol. Custom attributes MUST NOT be defined under the **XSF**, **XSF2** or **XSF3** namespaces (http://schemas.microsoft.com/office/infopath/2003/solutionDefinition, http://schemas.microsoft.com/office/infopath/2006/solutionDefinition/extensions, http://schemas.microsoft.com/office/infopath/2009/solutionDefinition/extensions respectively). Any custom attribute MUST be ignored.

Child Elements
baseUrl
customValidation
viewsExtension
webPartProperties

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="xsf3:webPartProperties" minOccurs="0"/>
      <xsd:element ref="xsf3:viewsExtension" minOccurs="0"/>
      <xsd:element ref="xsf3:customValidation" minOccurs="0"/>
      <xsd:element ref="xsf3:baseUrl" minOccurs="0" maxOccurs="1"/>
    </xsd:all>
  </xsd:complexType>
</xsd:element>
```

2.2.3.2.2 baseUrl

The **baseUrl** element specifies the location of the form template.

Parent Elements
solutionDefinition

Attributes:

queryServerInfo : This attribute MUST be ignored.

relativeUrlBase : This attribute specifies the location where the form template is published. It MUST be an absolute URL that points either to the form template or to the folder where the form template is hosted.

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

```
<xsd:element name="baseUrl">
  <xsd:complexType>
    <xsd:attribute name="relativeUrlBase" type="xsd:anyURI" use="optional"/>
```

```
      <xsd:attribute name="queryServerInfo" type="xsf:xdYesNo" use="optional"/>
      <xsd:anyAttribute processContents="skip"/>
    </xsd:complexType>
  </xsd:element>
```

2.2.3.2.3 webPartProperties

The **webPartProperties** element specifies a collection of fields (3) that are promoted from the form file and made available to Web Part connections as properties that can be set or queried.

Parent Elements
solutionDefinition

Child Elements
webPartFields

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

```
<xsd:element name="webPartProperties">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf3:webPartFields" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.3.2.4 webPartFields

The **webPartFields** element specifies a collection of fields (3) that are promoted from the form file and made available to Web Part connections as properties that can be set or queried.

Parent Elements
webPartProperties

Child Elements
webPartField

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

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

```
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf3:webPartField" maxOccurs="unbounded" minOccurs="0"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
```

2.2.3.2.5 webPartField

The **webPartField** element specifies a field (3) that is promoted from the form file and made available to Web Part connections as a property that can be read from or written into. Each promoted field (3) for Web Part connections MUST be specified by an instance of this element.

Parent Elements
webPartFields

Attributes:

name : This attribute specifies the friendly name of the field (3) used to configure the Web Part connection.

node : This attribute specifies the XPath expression that evaluates to the corresponding field (3) in the form file.

parameterType : This attribute specifies how the field (3) specified by the **node** attribute is shared in a Web Part connection. This parameter has the following possible values:

- **input:** The field (3) MUST be only written into and MUST NOT be read from.
- **inputOutput:** The field (3) MUST be able to be read from and written into.
- **output:** The field MUST be only read from and MUST NOT be written into.

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

```
  <xsd:element name="webPartField">
    <xsd:complexType>
      <xsd:sequence/>
      <xsd:attribute name="name" type="xsd:string" use="required"/>
      <xsd:attribute name="node" type="xsd:string" use="required"/>
      <xsd:attribute name="parameterType" type="xsf3:xdParameterType" use="required"/>
      <xsd:anyAttribute processContents="skip"/>
    </xsd:complexType>
  </xsd:element>
```

2.2.3.2.6 solutionPropertiesExtension2009

The **solutionPropertiesExtension2009** element specifies that the form template (.xsn) file is intended for a specific form-filling scenario. The details of the scenario are specified in the **solutionMode** element.

Child Elements
solutionMode

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

```
<xsd:element name="solutionPropertiesExtension2009">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf3:solutionMode" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.3.2.7 solutionMode

The **solutionMode** element specifies the form-filling scenario associated with the form template.

Parent Elements
solutionPropertiesExtension2009

Attributes:

autogenerated : This attribute MUST be ignored by the form server.

isListEditForm : This attribute specifies whether the form template is used to create, edit, or view a list item on the form server. The value of "yes" MUST only be set when the **mode** attribute is set to "entity" or "list". If this attribute is not present, its value MUST be interpreted as "no".

mode : This attribute specifies the **xd:modeType** associated with the form template, as specified in section 2.2.3.1.2.

originalCtid : This attribute MUST be ignored by the form server.

originalPublishUrl : This attribute MUST be ignored by the form server.

originalPublishUrlFriendlyName : This attribute 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="solutionMode">
  <xsd:complexType>
    <xsd:attribute name="autogenerated" type="xsf:xdYesNo" use="optional"/>
    <xsd:attribute name="isListEditForm" type="xsf:xdYesNo" use="optional"/>
    <xsd:attribute name="mode" type="xsf3:xdModeType" use="required"/>
    <xsd:attribute name="originalCtid" type="xsd:string" use="optional"/>
    <xsd:attribute name="originalPublishUrl" type="xsd:anyURI" use="optional"/>
    <xsd:attribute name="originalPublishUrlFriendlyName" type="xsd:string" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

2.2.3.2.8 viewsExtension

The **viewsExtension** element contains extended information for the form views in this form template.

Parent Elements
solutionDefinition

Child Elements
viewExtension

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

```
<xsd:element name="viewsExtension">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf3:viewExtension" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.3.2.9 viewExtension

The **viewExtension** element specifies extended information for the **view** element, as specified in section 2.2.1.2.104.

Parent Elements
viewsExtension

Child Elements
signatureLines
xmlToEditExtension

Attributes:

ref : This attribute specifies the name of the corresponding form view and MUST match the name attribute of the corresponding **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="xsf3:xmlToEditExtension" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="xsf3:signatureLines" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="ref" type="xsf:xdTitle" use="required"/>
      <xsd:anyAttribute processContents="skip"/>
    </xsd:complexType>
  </xsd:element>
```

2.2.3.2.10 xmlToEditExtension

The **xmlToEditExtension** element specifies extended information for the **xmlToEdit** element, as specified in section 2.2.1.2.105.

Parent Elements
viewExtension

Attributes:

excludeHyperlink : This attribute specifies whether hyperlink references are allowed in a rich text box control. If this attribute is present, it MUST be set to "yes" and the **excludeTables** attribute MUST also be present.

excludeTables : This attribute specifies whether HTML tables, as specified in [HTML], are allowed in a rich text box control. If this attribute is present, it MUST be set to "yes" and the **excludeHyperlink** attribute MUST also be present.

ref : This attribute specifies the name of the corresponding control and MUST match the **name** attribute of the corresponding **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="excludeTables" type="xsf:xdYesNo" use="optional" default="no"/>
      <xsd:attribute name="excludeHyperlink" type="xsf:xdYesNo" use="optional" default="no"/>
      <xsd:anyAttribute processContents="skip"/>
    </xsd:complexType>
  </xsd:element>
```

2.2.3.2.11 signatureLines

The **signatureLines** element MUST be ignored.

Parent Elements
viewExtension

Child Elements
signatureLine

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

```
<xsd:element name="signatureLines">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf3:signatureLine" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
  <!-- signature line names must be unique -->
  <xsd:unique name="signature_line_names_unique">
    <xsd:selector xpath="./xsf3:signatureLine"/>
    <xsd:field xpath="@name"/>
  </xsd:unique>
</xsd:element>
```

2.2.3.2.12 signatureLine

The **signatureLine** element MUST be ignored.

Parent Elements
signatureLines

Child Elements
confirmationMessage
suggestedSignerEmailAddress
suggestedSignerName
suggestedSignerTitle

Attributes:

name : This attribute MUST be ignored.

showDate : This attribute MUST be ignored.

signatureType : This attribute MUST be ignored.

signedDataBlock : This attribute MUST be ignored.

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

```
<xsd:element name="signatureLine">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="confirmationMessage" type="xsf:xdSignedDataBlockMessage"
minOccurs="0"/>
      <xsd:element name="suggestedSignerName" type="xsf3:xdSignatureLinePropertyType"
minOccurs="0"/>
      <xsd:element name="suggestedSignerTitle" type="xsf3:xdSignatureLinePropertyType"
minOccurs="0"/>
      <xsd:element name="suggestedSignerEmailAddress" type="xsf3:xdSignatureLinePropertyType"
minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:NMTOKEN" use="required"/>
    <xsd:attribute name="signedDataBlock" type="xsd:string" use="required"/>
    <xsd:attribute name="showDate" type="xsf:xdYesNo" use="required"/>
    <xsd:attribute name="signatureType" type="xsf3:xdLineStamp" use="required"/>
  </xsd:complexType>
</xsd:element>
```

2.2.3.2.13 confirmationMessage

The **confirmationMessage** element MUST be ignored.

Parent Elements
signatureLine

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

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

2.2.3.2.14 suggestedSignerName

The **suggestedSignerName** element MUST be ignored.

Parent Elements
signatureLine

Attributes:

isCalculation : This attribute MUST be ignored.

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

```
<xsd:element name="suggestedSignerName" type="xsf3:xdSignatureLinePropertyType"
minOccurs="0"/>
<xsd:complexType name="xdSignatureLinePropertyType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="isCalculation" type="xsf:xdYesNo" use="required"/>
```

```
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

2.2.3.2.15 suggestedSignerTitle

The **suggestedSignerTitle** element MUST be ignored.

Parent Elements
signatureLine

Attributes:

isCalculation : This attribute MUST be ignored.

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

```
<xsd:element name="suggestedSignerTitle" type="xsf3:xdSignatureLinePropertyType"
minOccurs="0"/>
<xsd:complexType name="xdSignatureLinePropertyType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="isCalculation" type="xsf:xdYesNo" use="required"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

2.2.3.2.16 suggestedSignerEmailAddress

The **suggestedSignerEmailAddress** element MUST be ignored.

Parent Elements
signatureLine

Attributes:

isCalculation : This attribute MUST be ignored.

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

```
<xsd:element name="suggestedSignerEmailAddress" type="xsf3:xdSignatureLinePropertyType"
minOccurs="0"/>
<xsd:complexType name="xdSignatureLinePropertyType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="isCalculation" type="xsf:xdYesNo" use="required"/>
    </xsd:extension>
  </xsd:simpleContent>
```

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

2.2.3.2.17 customValidation

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

Parent Elements
solutionDefinition

Child Elements
errorBlank

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="xsf3:errorBlank" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

2.2.3.2.18 errorBlank

The **errorBlank** element specifies a custom validation that determines if the specified set of XML nodes in the form file are required to contain values.

Parent Elements
customValidation

Attributes:

expression : This attribute specifies an XPath expression to validate the XML nodes returned by evaluating the **match** attribute value.

expressionContext : This attribute specifies the XML node that the validation error is reported on, if the custom validation evaluates to "true()".

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

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

```
<xsd:element name="errorBlank">
  <xsd:complexType>
    <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="required"/>
    <xsd:anyAttribute processContents="skip"/>
  </xsd:complexType>
</xsd:element>
```

2.3 XML Schema Files (XSD) Specification

The XML schema documents in the form template (.xsn) file MUST conform 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 specified in section 2.2.1.2.42.

In this section, the following terms are used as specified in [XMLSCHEMA1]:

- constraining facet
- namespace prefix
- XSD
- XSD data type
- XSD schema

In section 2.3.1, the "xs" and "xsd" namespace prefixes in all XSD fragments MUST be associated with the "http://www.w3.org/2001/XMLSchema" namespace.

2.3.1 Control Representation

Each control that is bound to a field (3) 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.

Control	Section	Description
Button control	2.3.1.1	Specifies the representation of a button control in the XML schema document.
Check box control	2.3.1.2	Specifies the representation of a check box control in the XML schema document.
Choice group control and choice section control	2.3.2.1	Specifies the representation of a choice group control and choice section control
Combo box control	2.3.2.2	Specifies the representation of a combo box control in the XML schema document.
Contact selector control	2.3.1.3	Specifies the representation of a contact selector control

Control	Section	Description
		in the XML schema document.
Date and time picker control	2.3.2.3	Specifies the representation of a date and time picker control in the XML schema document.
Date picker control	2.3.1.4	Specifies the representation of a date picker control in the XML schema document.
Drop-down list control	2.3.1.5	Specifies the representation of a drop-down control in the XML schema document.
Embedded picture control	2.3.2.5	Specifies the representation of an embedded picture control in the XML schema document.
External item picker control	2.3.2.4	Specifies the representation of an external item picker control in the XML schema document.
Expression box control	2.3.1.6	Specifies the representation of an expression box control in the XML schema document.
File attachment control	2.3.1.7	Specifies the representation of a file attachment control in the XML schema document.
Hyperlink control	2.3.1.8	Specifies the representation of a hyperlink control in the XML schema document.
Hyperlink input control	2.3.2.6	Specifies the representation of a hyperlink input control in the XML schema document.
Linked picture control	2.3.2.8	Specifies the representation of a linked picture control in the XML schema document.
List box control	2.3.1.9	Specifies the representation of a list box control in the XML schema document.
List controls (bulleted list control, numbered list control and plain list control)	2.3.2.7	Specifies the representation of a list control (bulleted list control, numbered list control and plain list control) in the XML schema document.
Multiple-selection list box control	2.3.2.9	Specifies the representation of a multiple-selection list box control in the XML schema document.
Option button control	2.3.1.10	Specifies the representation of an option button control in the XML schema document.
Picture button control	2.3.2.10	Specifies the representation of a picture button control in the XML schema document.
Repeating section control	2.3.1.11	Specifies the representation of a repeating section control in the XML schema document.
Repeating table control	2.3.1.12	Specifies the representation of a repeating table control in the XML schema document.
Rich text box control	2.3.1.13	Specifies the representation of a rich text box control in the XML schema document.
Section control and optional	2.3.1.14	Specifies the representation of a section control in the

Control	Section	Description
section control		XML schema document.
SharePoint file attachment control	2.3.2.11	Specifies the representation of a SharePoint file attachment control in the XML schema document.
Table control	2.3.1.15	Specifies the representation of a table control in the XML schema document.
Text box control	2.3.1.16	Specifies the representation of a text box control in the XML schema document.

2.3.1.1 Button Control

A button control MUST be unbound. It has no XSD representation.

2.3.1.2 Check Box Control

A check box control that is not required to contain data and is bound to an XML element, with data type set to "boolean" and for which no constraining facets have been set, MUST have the following XSD definition, assuming that the element name is "field1":

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

A check box control that is required to contain data and is bound to an XML element, with data type set to "boolean" and for which no constraining facets have been set, MUST have the following XSD definition, assuming that the element name is "field1":

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

A check box control SHOULD be bound to a field (3) with one of the following XSD data types for which any valid constraining facets 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) MUST have the following complex type complex type XSD definition, assuming that the element name is "group1":

```
<xsd:element name="group1">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="pc:Person" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

The XSD element **Person** MUST have the following complex type XSD definition:

```
<xs:element name="Person">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="pc:DisplayName" minOccurs="0"/>
            <xs:element ref="pc:AccountId" minOccurs="0"/>
            <xs:element ref="pc:AccountType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="DisplayName" type="xs:string"/>
<xs:element name="AccountId" type="xs:string"/>
<xs:element name="AccountType" type="xs:string"/>
```

The XSD definitions assume that the "pc" namespace prefix is associated with the "http://schemas.microsoft.com/office/infopath/2007/PartnerControls" namespace.

2.3.1.4 Date Picker Control

A date picker control that is not required to contain data and is bound to an XML element, with data type set to "date" and for which no constraining facets have been set, MUST have the following XSD definition, assuming that the element name is "field1":

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

A date picker control that is required to contain data and is bound to an XML element, with data type set to "date" and for which no constraining facets have been set, MUST have the following XSD definition, assuming that the element name is "field1":

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

A date picker control SHOULD be bound to a field (3) with one of the following XSD data types for which any valid constraining facets 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 and is bound to an XML element, with data type set to "string" and for which no constraining facets have been set, MUST have the following XSD definition, assuming that the element name is "field1":

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

A drop-down list control that is not required to contain data and is bound to an XML element, with data type set to "string" and for which a **xsd:minLength** constraining facet has been set, MUST have the following XSD definition, assuming that the element name is "field1":

```
<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 (3) with one of the following XSD data types for which any valid constraining facets 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 (3) to retrieve the value that it displays. If bound, an expression box control MUST NOT change the data in the field (3) to which it is bound, though the data in that field (3) 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 and is bound to an XML element with data type set to "xsd:base64Binary", and for which no constraining facets have been set, MUST have the following XSD definition, assuming that the element name is "field1":

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

A file attachment control that is required to contain data and is bound to an XML element with data type set to "xsd:base64Binary", and for which an **xsd:minLength** constraining facet has been set, MUST have the following XSD definition, assuming that the element name is "field1":

```
<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 (3) with the **base64Binary** XSD data type, for which any valid constraining facets MAY also be set.

2.3.1.8 Hyperlink Control

A hyperlink control MAY be bound to up to two fields (3); one field (3) to retrieve the target of the hyperlink and another field (3) to retrieve the value of the hyperlink's display text. If bound, a hyperlink control MUST NOT change the data in either field (3) to which it is bound, though the data in those fields (3) 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 and is bound to an XML element with data type set to "string" and for which no constraining facets have been set, MUST have the following XSD definition, assuming that the element name is "field1":

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

A list box control that is required to contain data and is bound to an XML element with data type set to "string", and for which an **xsd:minLength** constraining facet has been set, MUST have the following XSD definition, assuming that the element name is "field1":

```
<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 (3) with one of the following XSD data types, for which any valid constraining facets 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 and is bound to an XML element with data type set to "string", and for which no constraining facets have been set, MUST have the following XSD definition, assuming that the element name is "field1":

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

An option button control that is required to contain data and is bound to an XML element with data type set to "string", and for which an **xsd:minLength** constraining facet has been set, MUST have the following XSD definition, assuming that the element name is "field1":

```
<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 (3) with one of the following XSD data types, for which any valid constraining facets MAY also be set:

- **string**
- **integer**
- **double**
- **boolean**
- **anyURI**
- **date**
- **time**
- **dateTime**

2.3.1.11 Repeating Section Control

A repeating section control that is bound to a **repeating group** that contains no other bound controls MUST have the following complex type XSD 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>
```

The previous definition assumes that the element name is "group1" and the repeating group name is "group2".

A repeating section control that is bound to a repeating group that contains a textbox control that is not required to contain data and is bound to an XML element with data type set to "string", and for which no constraining facets have been set, MUST have the following complex type XSD 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"/>
```

The previous definition assumes the element name for the repeating section control is "group1", the repeating group name is "group2", and the element name for the textbox control is "my:field1".

2.3.1.12 Repeating Table Control

A repeating table control that is bound to a repeating group that contains three textbox controls that are not required to contain data, and are bound to XML elements with data type set to "string", and for which no constraining facets have been set, MUST have the following complex type XSD 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"/>
```

The previous definition assumes that the element name for the repeating table control is "group1", the repeating group name is "group2", and the element names for the textbox controls are "field1", "field2", and "field3".

2.3.1.13 Rich Text Box Control

A rich text box control that is bound to an XML element MUST have the following complex type XSD definition, assuming that the element name is "field1":

```
<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 or optional section control that is bound to a group (1) that contains no other bound controls MUST have the following complex type XSD definition, assuming that the element name is "group1":

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

A section or optional section control that is bound to a group (1) that contains a textbox control that is not required to contain data, and is bound to an XML element with data type set to "string", and for which no constraining facets have been set, MUST have the following complex type XSD 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"/>
```

The previous definition assumes the element name for the repeating section control is "group1" and the element name for the textbox control is "my:field1".

2.3.1.15 Table Control

A table control MUST be unbound. It has no XSD representation.

2.3.1.16 Text Box Control

A text box control that is not required to contain data and is bound to an XML element with data type set to "string", and for which no constraining facets have been set, MUST have the following XSD definition, assuming that the element name is "field1":

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

A text box control that is required to contain data and is bound to an XML element with data type set to "string", and for which an **xsd:minLength** constraining facet has been set, MUST have the following XSD definition, assuming that the element name is "field1":

```
<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, for which any valid constraining facets MAY also be set:

- **string**
- **integer**
- **double**
- **boolean**
- **anyURI**
- **date**
- **time**
- **dateTime**

2.3.2 Controls Introduced in Version 2 of the Structure Specification

2.3.2.1 Choice Group Control and Choice Section Control

A choice group control that is bound to a group (1) that contains two choice section controls that are bound to XML elements that are not required to be present and that contain no bound controls, MUST have the following complex type XSD definition:

```
<xsd:element name="group1">
    <xsd:complexType>
        <xsd:choice minOccurs="0">
            <xsd:element ref="my:group2" minOccurs="0"/>
            <xsd:element ref="my:group3" minOccurs="0"/>
        </xsd:choice>
    </xsd:complexType>
</xsd:element>
<xsd:element name="group2">
    <xsd:complexType>
        <xsd:sequence/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="group3">
    <xsd:complexType>
        <xsd:sequence/>
    </xsd:complexType>
</xsd:element>
```

The previous definition assumes that the element name for the choice group control is "group1" and the element names for the choice section controls are "group2" and "group3".

A choice group control that is bound to a group (1) that contains two choice section controls that are bound to XML elements that are not required to be present, the first of which contains a text box control that is bound to an XML element with data type set to "string", and the second of which contains a check box control that is bound to an XML element with data type set to "boolean", MUST have the following complex type XSD definition:

```
<xsd:element name="group1">
    <xsd:complexType>
        <xsd:choice minOccurs="0">
            <xsd:element ref="my:group2" minOccurs="0"/>
            <xsd:element ref="my:group3" minOccurs="0"/>
        </xsd:choice>
    </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"/>
<xsd:element name="group3">
    <xsd:complexType>
        <xsd:sequence>
```

```
            <xsd:element ref="my:field2" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="field2" nillable="true" type="xsd:boolean"/>
```

The previous definition assumes the element name for the choice group control is "group1" and the element names for the choice section controls are "group2" and "group3". The text box control and checkbox control contained by the choice section controls are assumed to be named "field1" and "field2" respectively.

2.3.2.2 Combo Box Control

A combo box control that is not required to contain data and is bound to an XML element with data type set to "string", and for which no constraining facets have been set, MUST have the following XSD definition, assuming that the element name is "field1":

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

A combo box control that is required to contain data and is bound to an XML element with data type set to "string", and for which an **xsd:minLength** constraining facet has been set, MUST have the following XSD definition, assuming that the element name is "field1":

```
<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 combo box control SHOULD be bound to a field (3) with one of the following XSD data types, for which any valid constraining facets MAY also be set:

- **string**
- **integer**
- **double**
- **boolean**
- **anyURI**
- **date**
- **time**
- **dateTime**

2.3.2.3 Date and Time Picker Control

A date and time picker control consists of a date picker control, as specified in 2.3.1.4, and a text box control, as specified in 2.3.1.16, bound to the same XML element.

2.3.2.4 External Item Picker Control

An external item picker control that is bound to a group (1) MUST have the following complex type XSD definition, assuming that the element name is "group1":

```
<xsd:element name="group1">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="pc:BDCAssociatedEntity"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

The XSD element **BDCAssociatedEntity** MUST have the following complex type XSD schema definition:

```
<xs:element name="BDCAssociatedEntity">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="pc:BDCEntity" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute ref="pc:EntityNamespace"/>
        <xs:attribute ref="pc:EntityName"/>
        <xs:attribute ref="pc:SystemInstanceName"/>
        <xs:attribute ref="pc:AssociationName"/>
    </xs:complexType>
</xs:element>
<xs:attribute name="EntityNamespace" type="xs:string"/>
<xs:attribute name="EntityName" type="xs:string"/>
<xs:attribute name="SystemInstanceName" type="xs:string"/>
<xs:attribute name="AssociationName" type="xs:string"/>
<xs:element name="BDCEntity">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="pc:EntityDisplayName" minOccurs="0"/>
            <xs:element ref="pc:EntityInstanceReference" minOccurs="0"/>
            <xs:element ref="pc:EntityId1" minOccurs="0"/>
            <xs:element ref="pc:EntityId2" minOccurs="0"/>
            <xs:element ref="pc:EntityId3" minOccurs="0"/>
            <xs:element ref="pc:EntityId4" minOccurs="0"/>
            <xs:element ref="pc:EntityId5" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="EntityDisplayName" type="xs:string"/>
<xs:element name="EntityInstanceReference" type="xs:string"/>
<xs:element name="EntityId1" type="xs:string"/>
<xs:element name="EntityId2" type="xs:string"/>
<xs:element name="EntityId3" type="xs:string"/>
<xs:element name="EntityId4" type="xs:string"/>
```

```
<xs:element name="EntityId5" type="xs:string"/>
```

The previous definition assumes that the "pc" namespace prefix is associated with the "http://schemas.microsoft.com/office/infopath/2007/PartnerControls" namespace.

2.3.2.5 Embedded Picture Control

An embedded picture control that is not required to contain data and is bound to an XML element with data type set to "xsd:base64Binary", and for which no constraining facets have been set, MUST have the following XSD definition, assuming that the element name is "field1":

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

An embedded picture control that is required to contain data and is bound to an XML element, with data type set to "xsd:base64Binary", and for which an **xsd:minLength** constraining facet has been set, MUST have the following XSD definition, assuming that the element name is "field1":

```
<xsd:element name="field1" type="my:requiredBase64Binary"/>
<xsd:simpleType name="requiredBase64Binary">
    <xsd:restriction base="xsd:base64Binary">
        <xsd:minLength value="1"/>
    </xsd:restriction>
</xsd:simpleType>
```

An embedded picture control SHOULD be bound to a field (3) with the **base64Binary** XSD data type, for which any valid constraining facets MAY also be set.

2.3.2.6 Hyperlink Input Control

A hyperlink input control that is not required to contain data and is bound, using the **xd:binding** attribute, as specified in section 2.4.2.6, to an XML element with data type set to "xsd:anyURI", and for which no constraining facets have been set, MUST have the following XSD definition, assuming that the element name is "field1":

```
<xsd:element name="field1">
```

A hyperlink input control that is required to contain data and is bound, using the **xd:binding** attribute, to an XML element with data type set to "xsd:anyURI", and for which an **xsd:minLength** constraining facet has been set, MUST have the following XSD definition, assuming that the element name is "field1":

```
<xsd:element name="field1" type="my:requiredAnyURI"/>
<xsd:simpleType name="requiredAnyURI">
    <xsd:restriction base="xsd:anyURI">
        <xsd:minLength value="1"/>
    </xsd:restriction>
</xsd:simpleType>
```

A hyperlink input control that is not required to contain data that is bound to two XML elements as follows:

- Bound, using the **xd:binding** attribute to an XML element with data type set to "xsd:anyURI", and for which no constraining facets have been set and
- Bound, using the **xd:binding_secondary** attribute, as specified in section 2.4.2.37.2, to another XML element with data type set to "xsd:string", for which no constraining facets have been set

MUST have the following XSD definition, assuming that the element names are "field1" and "field2":

```
<xsd:attribute name="field2" type="xsd:string" />
<xsd:element name="field1">
    <xsd:complexType>
        <xsd:simpleContent>
            <xsd:extension base="xsd:anyURI">
                <xsd:attribute ref="my:field2" />
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:element>
```

A hyperlink input control that is required to contain data and is bound to two XML elements as follows:

- Bound, using the **xd:binding** attribute to an XML element with data type set to "xsd:anyURI", and for which an **xsd:minLength** constraining facet has been set and
- Bound, using the **xd:binding_secondary** attribute, to another XML element with data type set to "xsd:string", and for which an **xsd:minLength** constraining facet has been set

MUST have the following XSD definition, assuming that the element names are "field1" and "field2":

```
<xsd:attribute name="field2" type="my:requiredString" />
<xsd:element name="field1">
    <xsd:complexType>
        <xsd:simpleContent>
            <xsd:extension base="my:requiredAnyURI">
                <xsd:attribute ref="my:field2" />
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:element>
<xsd:simpleType name="requiredString">
    <xsd:restriction base="xsd:string">
        <xsd:minLength value="1" />
    </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="requiredAnyURI">
    <xsd:restriction base="xsd:anyURI">
        <xsd:minLength value="1" />
    </xsd:restriction>
</xsd:simpleType>
```

A hyperlink input control SHOULD be bound, using the **xd:binding** attribute, to a field (3) with one of the following XSD data types, for which any valid constraining facets MAY also be set:

- **anyURI**
- **string**

A hyperlink input control SHOULD be bound, using the **xd:binding_secondary** attribute, to a field (3) with one of the following XSD data types, for which any valid constraining facets MAY also be set:

- **anyURI**
- **string**
- **integer**
- **double**
- **boolean**
- **anyURI**
- **date**
- **time**
- **dateTime**
- **base64**

2.3.2.7 List Controls (Bulleted List Control, Numbered List Control and Plain List Control)

A list control that is not required to contain data and is bound to a repeating XML element with data type set to "string", and for which no constraining facets have been set, MUST have the following complex type XSD definition, assuming that the repeating element name is "field1":

```
<xsd:element name="group1">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="my:field1" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="field1" type="xsd:string" />
```

A list control that is required to contain data and is bound to a repeating XML element with data type set to "string", and for which an **xsd:minLength** constraining facet has been set, MUST have the following complex type XSD definition, assuming that the repeating element name is "field1":

```
<xsd:element name="group1">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="my:field1" minOccurs="0" maxOccurs="unbounded" />
```

```
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<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 control SHOULD be bound to a field (3) with one of the following XSD data types, for which any valid constraining facets MAY also be set:

- **string**
- **XHTML**

2.3.2.8 Linked Picture Control

A linked picture control that is not required to contain data and is bound, using the **xd:binding** attribute, as specified in section 2.4.2.6, to an XML element with data type set to "xsd:anyURI", and for which no constraining facets have been set, MUST have the following XSD definition, assuming that the element name is "field1":

```
<xsd:element name="field1">
```

A linked picture control that is required to contain data and is bound, using the **xd:binding** attribute, to an XML element with data type set to "xsd:anyURI" and for which an **xsd:minLength** constraining facets has been set, MUST have the following XSD definition, assuming that the element name is "field1":

```
<xsd:element name="field1" type="my:requiredAnyURI"/>
<xsd:simpleType name="requiredAnyURI">
    <xsd:restriction base="xsd:anyURI">
        <xsd:minLength value="1"/>
    </xsd:restriction>
</xsd:simpleType>
```

A linked picture control that is not required to contain data and is bound to two XML elements as follows:

- Bound, using the **xd:binding** attribute, to an XML element with data type set to "xsd:anyURI", and for which no constraining facets have been set and
- Bound, using the **xd:binding_secondary** attribute, as specified in section 2.4.2.37.2, to another XML element with data type set to "xsd:string", for which no constraining facets have been set

MUST have the following XSD definition, assuming that the element names are "field1" and "field2":

```
<xsd:attribute name="field2" type="xsd:string" />
<xsd:element name="field1">
    <xsd:complexType>
```

```
        <xsd:simpleContent>
            <xsd:extension base="xsd:anyURI">
                <xsd:attribute ref="my:field2" />
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:element>
```

A linked picture control that is required to contain data and is bound to two XML elements as follows:

- Bound, using the **xd:binding** attribute, to an XML element with data type set to "xsd:anyURI", and for which an **xsd:minLength** constraining facet has been set and
- Bound, using the **xd:binding_secondary** attribute, to another XML element with data type set to "xsd:string", and for which an **xsd:minLength** constraining facet has been set

MUST have the following XSD definition, assuming that the element names are "field1" and "field2":

```
<xsd:attribute name="field2" type="my:requiredString" />
<xsd:element name="field1">
    <xsd:complexType>
        <xsd:simpleContent>
            <xsd:extension base="my:requiredAnyURI">
                <xsd:attribute ref="my:field2" />
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:element>
<xsd:simpleType name="requiredString">
    <xsd:restriction base="xsd:string">
        <xsd:minLength value="1" />
    </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="requiredAnyURI">
    <xsd:restriction base="xsd:anyURI">
        <xsd:minLength value="1" />
    </xsd:restriction>
</xsd:simpleType>
```

A linked picture control SHOULD be bound, using the **xd:binding** attribute, to a field (3) with one of the following XSD data types, for which any valid constraining facets MAY also be set:

- **anyURI**
- **string**

A linked picture control SHOULD be bound, using the **xd:binding_secondary** attribute, to a field (3) with one of the following XSD data types, for which any valid constraining facets MAY also be set:

- **anyURI**
- **string**

- **integer**
- **double**
- **boolean**
- **date**
- **time**
- **dateTime**
- **base64**

2.3.2.9 Multiple-Selection List Box Control

A multiple-selection list box control that is not required to contain data and is bound to a repeating XML element with data type set to "string", and for which no constraining facets have been set, MUST have the following complex type XSD definition, assuming that the repeating element name is "field1":

```
<xsd:element name="group1">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="my:field1" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="field1" type="xsd:string" />
```

A multiple-selection list box control that is required to contain data and is bound to a repeating XML element with data type set to "string", and for which an **xsd:minLength** constraining facet has been set, MUST have the following complex type XSD definition, assuming that the repeating element name is "field1":

```
<xsd:element name="group1">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="my:field1" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="field1" type="my:requiredString" />
<xsd:simpleType name="requiredString">
    <xsd:restriction base="xsd:string">
        <xsd:minLength value="1" />
    </xsd:restriction>
</xsd:simpleType>
```

2.3.2.10 Picture Button Control

A picture button control MUST be unbound. It has no XSD representation.

2.3.2.11 SharePoint File Attachment Control

A SharePoint file attachment control that is bound to a group (1) MUST have the following complex type XSD definition, assuming that the element name is "Attachments":

```
<xsd:element name="Attachments">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="attachmentURL" type="xsd:anyURI" nillable="true"
                         minOccurs="0" maxOccurs="unbounded"></xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

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:** Specifies a valid XSLT transformation of the form data into HTML.
- **Section** 2.4.2 **Control-specific Attributes:** Specifies the HTML representation of certain control properties and behaviors.
- **Section** 2.4.3 **XSL Function Extensions:** Specifies extensions to XSLT used in transforming the form data to HTML.

2.4.1 View Representation

Each control MUST have the representation specified in the following sections. Simple or complex XSD values, as specified in [XMLSCHEMA1], MUST be rendered using controls.

The following table lists the sections that specify the different constructs used to represent the XSLT file.

View Construct	Section	Description
View Syntax	2.4.1.1	Specifies the syntax used to represent the XSLT file.
XSL Root Template	2.4.1.2	Specifies the starting element that contains the XSLT file.
XSL Root Template Style Sheets	2.4.1.3	Specifies the representation for the style sheet.
Control Data Formatting	2.4.1.4	Specifies the representation of data formatting for multiple controls.
Button Control	2.4.1.5	Specifies the representation of a button control.

View Construct	Section	Description
Check Box Control	2.4.1.6	Specifies the representation of a check box control.
Choice Group Control and Choice Section Control	2.4.1.21.1	Specifies the representation of a choice group control and a choice section control.
Combo Box Control	2.4.1.21.2	Specifies the representation of a combo box control.
Contact Selector Control	2.4.1.7	Specifies the representation of a contact selector control.
Date and Time Picker Control	2.4.1.21.3	Specifies the representation of a date and time picker control.
Date Picker Control	2.4.1.8	Specifies the representation of a date picker control.
Drop-Down List Control	2.4.1.9	Specifies the representation of a drop-down list control.
Embedded Picture Control	2.4.1.21.5	Specifies the representation of an embedded picture control.
External Item Picker Control	2.4.1.21.4	Specifies the representation of an external item picker control.
Expression Box Control	2.4.1.10	Specifies the representation of an expression box control.
File Attachment Control	2.4.1.11	Specifies the representation of a file attachment control.
Hyperlink Control	2.4.1.12	Specifies the representation of a hyperlink control.
Hyperlink Input Control	2.4.1.21.6	Specifies the representation of a hyperlink input control.
Linked Picture Control	2.4.1.21.8	Specifies the representation of a linked picture control.
List Box Control	2.4.1.13	Specifies the representation of a list box control.
List Controls (Bulleted List Control, Numbered List Control and Plain List Control)	2.4.1.21.7	Specifies the representation of a list controls (bulleted list control, numbered list control, and plain list control).
Multiple-Selection List Box Control	2.4.1.21.9	Specifies the representation of a multiple-selection list box control.
Option Button Control	2.4.1.14	Specifies the representation of an option button control.
Picture Button Control	2.4.1.21.10	Specifies the representation of a picture button control.
Repeating Section Control	2.4.1.15	Specifies the representation of a repeating section

View Construct	Section	Description
		control.
Repeating Table Control	2.4.1.16	Specifies the representation of a table control.
Rich Text Box Control	2.4.1.17	Specifies the representation of a rich text box control.
Section Control and Optional Section Control	2.4.1.18	Specifies the representation of a section control.
SharePoint File Attachment Control	2.4.1.21.11	Specifies the representation of a SharePoint file attachment control.
Table Control	2.4.1.19	Specifies the representation of a table control.
Text Box Control	2.4.1.20	Specifies the representation of a text box control.
Ignored Controls	2.4.1.22	Specifies a list of controls that are ignored.
Invalid Controls	2.4.1.23	Specifies a list of controls that are invalid.
Invalid Constructs	2.4.1.24	Specifies a list of invalid constructs for the XSLT.

2.4.1.1 View Syntax

The formal grammar of a view XSL file is given in this specification using an **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

```
SYMBOL ::= expression
```

Each rule in the grammar defines one symbol, in the following form:

- Symbols are written with upper case 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).
- #x*N* -where *N* is a hexadecimal integer, the expression matches the character whose number, or code point, in [ISO-10646] is *N*. The number of leading zeros in the #x*N* form is insignificant.
- [a-zA-Z], [#x*N*-#x*N*]: Matches any character with a value in the range(s) indicated (inclusive).
- [abc], [#x*N*#x*N*#x*N*]: Matches any character with a value among the characters enumerated. Enumerations and ranges can be mixed in one set of brackets.
- [^a-z], [^#x*N*-#x*N*]: Matches any character with a value outside the range indicated.
- [^abc], [^#x*N*#x*N*#x*N*]: 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 is given inside the single quotes.

To match more complex patterns these symbols MUST be combined as follows, where **A** and **B** represent simple expressions:

- (expression): Expression is treated as a unit and MUST be combined as specified in this list.
- **semicolon-delimited list**((expression)(, expression)*): Matches a semicolon-delimited list of expressions.
- **A**?: Matches **A** or nothing; optional **A**.
- **A B**: Matches **A** followed by **B**. This operator has higher precedence than alternation; thus **A B** | **C D** is identical to (**A B**) | (**C D**).
- **A** | **B**: Matches **A** or **B**.
- **A** – **B**: Matches any string that matches **A** but does not match **B**.
- **A**+: Matches one or more occurrences of **A**. Concatenation has higher precedence than alternation; thus **A**+ | **B**+ is identical to (**A**+) | (**B**+).
- **A***: Matches zero or more occurrences of **A**. Concatenation has higher precedence than alternation; thus **A*** | **B*** is identical to (**A***) | (**B***).

text: The text that does not match any production specified earlier MUST be interpreted as a literal. Additionally, any construct that has the same semantics in the target language ([HTML], [CSS-LEVEL1] , [W3C-XML] [XMLSCHEMA1] [XPATH] and [W3C-XSLT]) can substitute the literal text.

The order and value of element attributes in the EBNF rules 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"/>
```

This is true because the **class** attribute is specified in HTML and the syntax of the attribute is case-insensitive. Also the number of white spaces or tabs between **xdTextBox** and **xdBehavior_Formatting** is not important as long as there is at least one. The syntax for the values of **class** attributes MUST be as specified in [HTML] section 7.5.2.

The following productions MUST be used for the controls representation:

```
ISO_646_DIGIT::= [#x0030-#x0039]
LATIN_CHARACTER::= [#x0041-#x005A] | [#x0061- #x007a]
SINGLE_CHARACTER  ::= LATIN_ CHARACTER | ISO_646_DIGIT
BUTTON_POSTBACKMODEL ::= always | auto
POSTBACKMODEL ::= never | BUTTON_POSTBACKMODEL
```

The semantics of the postback model values MUST be as specified in section 2.4.2.29.

CONTROL_ID: (LATIN_CHARACTER) (LATIN_CHARACTER|ISO_646_DIGIT|_)*. The value of **CONTROL_ID** MUST be a valid value for type **xsf: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 section 2.4.2.21, section 2.4.2.20, and section 2.4.2.2.

ALIGN: "left" or "right".

VALIGN: "middle" or "baseline" or "bottom" or "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.

FONT_COLOR: Values MUST be as specified in [HTML] section 15.2.2.

FONT_FACE: Values MUST be as specified in [HTML] section 15.2.2.

XML_TO_EDIT_NAME: Nmtoken, as specified in [W3C-XML] and MUST match the name of a corresponding **xmlToEdit** entry, as specified in section 2.4.2.36, in the XSF file.

TAB_INDEX<3>: "-1" or as specified in [HTML] section 17.11.

HEIGHT: Value pairs MUST be as specified in [HTML] section 13.7.1.

MIN_HEIGHT: Values MUST be as specified in [CSS-LEVEL2] section 10.7.

WIDTH: Value pairs MUST be as specified in [HTML] section 13.7.1.

COLSPAN: Value pairs MUST be as specified in [HTML] section 11.2.6.

ROWSPAN: Value pairs MUST be as specified in [HTML] section 11.2.6.

STYLE_SIZE: Value pairs MUST be as specified 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] section 5.5.23.

STYLE_HEIGHT: Value pairs MUST be as specified in [CSS-LEVEL1] section 5.5.24.

CSS1_STYLE: Values MUST be as specified in [CSS-LEVEL1].

STYLE_DISPLAY_NONE: DISPLAY: none, as specified in [CSS-LEVEL1] section 5.6.1.

STYLE_MARGIN: Value pairs MUST be as specified in [CSS-LEVEL1] sections 5.5.1, 5.5.2, 5.5.3, 5.5.4, and 5.5.5.

STYLE_PADDING: Value pairs MUST be as specified in [CSS-LEVEL1] sections 5.5.6, 5.5.7, 5.5.8, 5.5.9, and 5.5.10.

STYLE_TEXT_DECORATION: Value pairs MUST be as specified in [CSS-LEVEL1] section 5.4.3.

STYLE_BACKGROUND_COLOR: Value pairs MUST be as specified in [CSS-LEVEL1] section 5.3.2.

STYLE_BORDER: Value pairs MUST be as specified in [CSS-LEVEL1] sections 5.5.11, 5.5.12, 5.5.13, 5.5.14, 5.5.15, 5.5.16, 5.5.17, 5.5.18, 5.5.19, 5.5.20, 5.5.21, and 5.5.22.

STYLE_BORDER_STYLE: Value pairs MUST be as specified in [CSS-LEVEL2] section 8.5.3.

STYLE_BORDER_COLLAPSE: Value pairs MUST be as specified in [CSS-LEVEL2] section 17.6.

STYLE_FONT: Value pairs MUST be as specified in [CSS-LEVEL1] section 5.2.

STYLE_FONT_STYLE: Value pairs MUST be as specified in [CSS-LEVEL1] section 5.2.3.

STYLE_FONT_FAMILY: Value pairs MUST be as specified in [CSS-LEVEL1] section 5.2.2.

STYLE_FONT_STYLE: Value pairs MUST be as specified in [CSS-LEVEL1] section 5.2.3.

STYLE_COLOR: Value pairs MUST be as specified in [CSS-LEVEL1] section 5.3.1.

STYLE_FONT_WEIGHT: Value pairs MUST be as specified in [CSS-LEVEL1] section 5.2.5.

STYLE_TEXT_ALIGN: Value pairs MUST be as specified in [CSS-LEVEL1] section 5.4.6

STYLE_WRAP: WHITE-SPACE: normal or WHITE-SPACE: nowrap; WORD-WRAP: normal

STYLE_OVERFLOW: Value pairs MUST be as specified in [CSS-LEVEL2] section 11.1.1.

STYLE_VERTICAL_ALIGN: VERTICAL-ALIGN: ("sub" or "super").

STYLE_DIRECTION: (**DIRECTION**: ltr) or (**DIRECTION**: rtl).

STYLE_DISABLE_CHILD_XML_TO_EDIT:

```
msos-(xOptional|xCollection)-XML_TO_EDIT_NAME-editing:disabled;
```

STYLE_CAPTION: caption: **NON_EMPTY_STRING**(;valid: false).

STYLE_XD_BACKGROUND_COLOR:= **xdBackgroundColor**: Value MUST be as specified in [CSS-LEVEL1] section 5.3.2.

LEAF_CONTROL_CONDITIONAL_FORMATTING_CAPTION: Semicolon-delimited list of ((**STYLE_DISPLAY_NONE** | (**STYLE_FONT**?, **STYLE_COLOR**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_TEXT_DECORATION**?)), **STYLE_CAPTION**)

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 the **dataObject** element, as specified in section 2.2.1.2.17.

LEAF_XPATH: MUST be an extended location XPath expression, as specified in [XPATH], and MUST use only the child and attribute XPath axes, as specified in [XPATH] section 2.2. The generated **node-set** MUST have only 1 element. The XPath expression MUST be a relative or absolute XPath. To alter the context of the XPath evaluation, the **xdXDocument:GetDOM** function, as specified in section 2.4.3.9.2, MUST be used before the first XPath step.

GROUP_XPATH: MUST be an extended location XPath expression and MUST use only the child XPath axes, as specified in [XPATH] section 2.2. There are no restrictions for the number of elements in the generated **node-set**. The XPath 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 expression and MUST use only the child XPath axes, as specified in [XPATH] section 2.2. There are no restrictions for the number of elements in the generated **node-set**. The XPath expression MUST be a relative XPath.

RELATIVE_GROUP_XPATH: MUST be an extended location XPath expression and MUST use only the child XPath axes, as specified in [XPATH] section 2.2. There are no restrictions for the number of elements in the generated node-set. The XPath expression MUST be a relative XPath.

RELATIVE_LEAF_XPATH: MUST be a location XPath expression and MUST use only the child and attribute XPath axes, as specified in [XPATH] section 2.2. The XPath expression MUST be a relative XPath.

REPEATING_LEAF_XPATH: MUST be an extended location XPath expression, as specified in [XPATH] and MUST use only the child and attribute XPath axes, as specified in [XPATH] section 2.2. There are no restrictions for the number of elements in the generated **node-set**. The XPath expression MUST be a relative or absolute XPath. To alter the context of the XPath evaluation, the **xdXDocument:GetDOM** function MUST be used before the first XPath step.

PREDICATE_XPATH: MUST be an expression XPath expression. The predicate expression MUST NOT use the following functions:

- **position()**
- **last()**
- The [*n*] syntax where *n* is a positive integer.

The expression MUST NOT contain user-defined variable references, as specified in [XPATH] section 2.4.

BOOLEAN_XPATH_EXPRESSION: MUST be an XPath expression that yields an object that MUST be a **Boolean** basic type. **Boolean** is specified in [XPATH] section 3.4. To extend the syntax of XPath expression, the XSL function extensions specified in section 2.4.3 MUST be used. The XPath expression MUST be less than 100 in depth. It MUST NOT use **position** and **last** functions, as 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 that yields an object that has a **String** basic type. **String** is specified in [XPATH] section 3.6. To extend the syntax of XPath

expressions, the XSL function extensions specified in section 2.4.3 MUST be used. The XPath expression MUST be less than 100 in depth. It MUST NOT use **position** and **last** functions, as specified in [XPATH] section 4.1. It MUST NOT use XPath predicates. XPath predicates are specified in [XPATH] section 2.4.

HIDDEN_FORMATTING_CAPTION: (xd:caption_(ISO_646_DIGIT)+="NON_EMPTY_STRING")+. The first instance of the attribute MUST be named "xd:caption_0" and subsequent instances MUST be named with consecutive integers. The value of this attribute MUST be ignored by the form server. See section 2.4.2.37.11 for more details.

CHECK_FOR_GETDOM_BEGIN:

```
(<xsl:if test="function-available('xdXDocument:GetDOM')">)?.
```

CHECK_FOR_GETDOM_END:

```
(</xsl:if>)?.
```

CHECK_FOR_GETDOM_BEGIN and **CHECK_FOR_GETDOM_END** symbols always appear in pairs in the EBNF rules in the following sections. Subscripts are used to mark the pairs.

If the yield of **CHECK_FOR_GETDOM_BEGIN** in one production is empty, the yield of the pairing **CHECK_FOR_GETDOM_END** MUST be empty.

If the yield of **CHECK_FOR_GETDOM_END** in one production is empty, the yield of the pairing **CHECK_FOR_GETDOM_BEGIN** MUST be empty.

FONT: **ANY_STRING** without a comma.

FONT_ITALIC: Specifies if the text is shown in italic or not. The following table lists the possible values and explanations.

Value	Description
"0"	not italic
"1"	Italic

FONT_SIZE: All integers and all real numbers, ending in .5, between 1 and 2000, inclusive.

FONT_STRIKETHROUGH: Specifies if the text is shown with a strike though line or not. The following table lists the possible values and explanations.

Value	Description
"0"	no strike through
"1"	strike through

FONT_UNDERLINE: Specifies if the text shown is underlined or not. The following table lists the possible values and explanations.

Value	Description
"0"	not underlined
"1"	Underlined

FONT_WEIGHT: Specifies if the text shown is bold or not. The following table lists the possible values and explanations.

Value	Description
"400"	not bold
"700"	Bold

CHARACTER_SET: ANY_STRING without a comma.

2.4.1.2 XSL Root Template

The starting element for the EBNF notation is **XSL_STYLE_SHEET**.

```
XSL_STYLE_SHEET ::=
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    (XML_NAMESPACE)*
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    (xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance")?
    (xmlns:xd="http://schemas.microsoft.com/office/infopath/2003")?
    (xmlns:msxsl="urn:schemas-microsoft-com:xslt")?
    (xmlns:x="urn:schemas-microsoft-com:office:excel")?
    (xmlns:xdExtension="http://schemas.microsoft.com/office/infopath/2003/xslt/extension")?
    (xmlns:xdXDocument="http://schemas.microsoft.com/office/infopath/2003/xslt/xDocument")?
    (xmlns:xdSolution="http://schemas.microsoft.com/office/infopath/2003/xslt/solution")?
    (xmlns:xdFormatting="http://schemas.microsoft.com/office/infopath/2003/xslt/formatting")?
    (xmlns:xdImage="http://schemas.microsoft.com/office/infopath/2003/xslt/xImage")?
    (xmlns:xdUtil="http://schemas.microsoft.com/office/infopath/2003/xslt/Util")?
    (xmlns:xdMath="http://schemas.microsoft.com/office/infopath/2003/xslt/Math")?
    (xmlns:xdDate="http://schemas.microsoft.com/office/infopath/2003/xslt/Date")?
    (xmlns:sig="http://www.w3.org/2000/09/xmldsig#")?
(xmlns:xdSignatureProperties="http://schemas.microsoft.com/office/infopath/2003/SignatureProp
erties")?
    (xmlns:ipApp="http://schemas.microsoft.com/office/infopath/2006/XPathExtension/ipApp")?
(xmlns:xdEnvironment="http://schemas.microsoft.com/office/infopath/2006/xslt/environment")?
    (xmlns:xdUser="http://schemas.microsoft.com/office/infopath/2006/xslt/User")?
    (xmlns:ma="http://schemas.microsoft.com/office/2009/metadata/properties/metaAttributes")?
    (xmlns:dfs="http://schemas.microsoft.com/office/infopath/2003/dataFormSolution")?
    (xmlns:d="http://schemas.microsoft.com/office/infopath/2009/WSSList/dataFields")?
    (xmlns:pc="http://schemas.microsoft.com/office/infopath/2007/PartnerControls")?
    (xmlns:q="http://schemas.microsoft.com/office/infopath/2009/WSSList/queryFields")?
    (xmlns:dms="http://schemas.microsoft.com/office/2009/documentManagement/types")?
    (xmlns:xdServerInfo="http://schemas.microsoft.com/office/infopath/2009/xslt/ServerInfo")?
    (xmlns:xsf="http://schemas.microsoft.com/office/infopath/2003/solutionDefinition")?

(xmlns:xsf2="http://schemas.microsoft.com/office/infopath/2006/solutionDefinition/extensions"
)?
```

```
(xmlns:xsf3="http://schemas.microsoft.com/office/infopath/2009/solutionDefinition/extensions"
)?
    (xmlns:xsd="http://www.w3.org/2001/XMLSchema")?
    (xmlns:xhtml="http://www.w3.org/1999/xhtml")? >
    <xsl:output method="html" indent="no"/>
    <xsl:template match="GROUP_XPATH">
        <html (dir="HTML_DIR")?
(xmlns:xsf="http://schemas.microsoft.com/office/infopath/2003/solutionDefinition")?
(xmlns:xsf2="http://schemas.microsoft.com/office/infopath/2006/solutionDefinition/extensions"
)?
(xmlns:xsf3="http://schemas.microsoft.com/office/infopath/2009/solutionDefinition/extensions"
)?
(xmlns:xsd="http://www.w3.org/2001/XMLSchema")? (xmlns:xhtml="http://www.w3.org/1999/xhtml")?
>
            <head>
               (HTML_COMMENTS)?
               <meta http-equiv="Content-Type" content="text/html"></meta>
               (CONTROL_STYLE)?
               TABLE_STYLE
               LANGUAGE_STYLE
               (THEME_STYLE)?
               (PRETTYFORMS_TABLE_STYLE)?
            </head>
            <body (style="CSS1_STYLE")? (background="IMAGE_FILE")?
                (scroll="auto")?>MAIN_BODY</body>
        </html>
    </xsl:template>
    (SECTION_BODY | REPEATING_SECTION_BODY)*
</xsl:stylesheet>

MAIN_BODY ::= XML_HTML_4_1_WITH_CONTROLS |
<span>
    <xsl:attribute name="style">
        (<xsl:if test="BOOLEAN_XPATH_EXPRESSION">STYLE_DISABLE_CHILD_XML_TO_EDIT</xsl:if>)+
    </xsl:attribute>
    XML_HTML_4_1_WITH_CONTROLS
</span>
```

XML_HTML_4_1_WITH_CONTROLS: MUST be an HTML 4.1 fragment, as specified in [HTML], valid under the **BODY** element that is also a valid XML 1.0 fragment, as specified in [W3C-XML]. If an element inside the fragment contains the **xd:xctname** attribute, it MUST conform to one of the control productions specified for controls in section 2.4.1.5 to section 2.4.1.21. If the fragment contains an XSL element with the syntax of **SECTION_CALL**, it MUST be located only in the locations where a <**DIV**/> element, as specified in [HTML] section 7.5.4, could also be placed.

SECTION_CALL: SIMPLE_SECTION_CALL or **OPTIONAL_SECTION_CALL** or **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: 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.

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:17px;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:17px;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:17px;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:17px;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:17px;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 {TEXT-TRANSFORM: none; FONT-STYLE: normal; FONT-FAMILY: Calibri; COLOR: black;
FONT-SIZE: 10pt; FONT-WEIGHT: normal}
    SELECT {FONT-SIZE: 10pt; FONT-FAMILY: Verdana}
    .optionalPlaceholder {PADDING-LEFT: 20px; FONT-WEIGHT: normal; FONT-SIZE: 9pt; BEHAVIOR:
url(#default#xOptional); COLOR: #333333; FONT-STYLE: normal; FONT-FAMILY: Verdana; TEXT-
DECORATION: none}
    .langFont {FONT-FAMILY: Verdana}
    .defaultInDocUI {FONT-SIZE: 9pt; FONT-FAMILY: Verdana}
    .optionalPlaceholder {PADDING-RIGHT: 20px}
</style>
|
 <style languageStyle="languageStyle">body, select {font-family:CSS_FONT_FAMILY;font-
size:CSS_FONT_SIZE} table{ TEXT-TRANSFORM: none; FONT-STYLE: normal; COLOR: black; FONT-
WEIGHT: normal 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: MUST have one of the values in the following table.

Theme Style	Settings
THEME_STANDARD	<style themestyle="urn:office.microsoft.com:themeOffice"> TABLE { BORDER-BOTTOM: medium none; BORDER-LEFT: medium none; BORDER-COLLAPSE: collapse; BORDER-TOP: medium none; BORDER-RIGHT: medium none } TD { BORDER-BOTTOM-COLOR: #d8d8d8; BORDER-TOP-COLOR: #d8d8d8; BORDER-RIGHT-COLOR: #d8d8d8; BORDER-LEFT-COLOR: #d8d8d8 } TH { BORDER-BOTTOM-COLOR: #000000; BACKGROUND-COLOR: #f2f2f2; BORDER-TOP-COLOR: #000000; COLOR: black; BORDER-RIGHT-COLOR: #000000; BORDER-LEFT-COLOR: #000000 } .xdTableHeader { BACKGROUND-COLOR: #f2f2f2; COLOR: black } .light1 { BACKGROUND-COLOR: #ffffff } .dark1 { BACKGROUND-COLOR: #000000 } .light2 { BACKGROUND-COLOR: #f6f6f6 } .dark2 { BACKGROUND-COLOR: #182738 } .accent1 { BACKGROUND-COLOR: #0072bc } .accent2 { BACKGROUND-COLOR: #ec008c } .accent3 { BACKGROUND-COLOR: #00adee } .accent4 { BACKGROUND-COLOR: #fd9f08 } .accent5 { BACKGROUND-COLOR: #36b000 } .accent6 { BACKGROUND-COLOR: #fae032 }</style>
THEME_AZURE	<style themestyle="urn:office.microsoft.com:themeAzure"> TABLE { BORDER-BOTTOM: medium none; BORDER-LEFT: medium none; BORDER-COLLAPSE: collapse; BORDER-TOP: medium none; BORDER-RIGHT: medium none } TD { BORDER-BOTTOM-COLOR: #d8d8d8; BORDER-TOP-COLOR: #d8d8d8; BORDER-RIGHT-COLOR: #d8d8d8; BORDER-LEFT-COLOR:

Theme Style	Settings
	#d8d8d8 } TH { BORDER-BOTTOM-COLOR: #000000; BACKGROUND-COLOR: #f2f2f2; BORDER-TOP-COLOR: #000000; COLOR: black; BORDER-RIGHT-COLOR: #000000; BORDER-LEFT-COLOR: #000000 } .xdTableHeader { BACKGROUND-COLOR: #f2f2f2; COLOR: black } .light1 { BACKGROUND-COLOR: #ffffff } .dark1 { BACKGROUND-COLOR: #000000 } .light2 { BACKGROUND-COLOR: #e7f7f6 } .dark2 { BACKGROUND-COLOR: #3e5354 } .accent1 { BACKGROUND-COLOR: #859e9d } .accent2 { BACKGROUND-COLOR: #9eb5b4 } .accent3 { BACKGROUND-COLOR: #5dd1c6 } .accent4 { BACKGROUND-COLOR: #7dc3d9 } .accent5 { BACKGROUND-COLOR: #8fdbd4 } .accent6 { BACKGROUND-COLOR: #a1bac7} </style>
THEME_BERRY	<style themestyle="urn:office.microsoft.com:themeBerry"> TABLE { BORDER-BOTTOM: medium none; BORDER-LEFT: medium none; BORDER-COLLAPSE: collapse; BORDER-TOP: medium none; BORDER-RIGHT: medium none } TD { BORDER-BOTTOM-COLOR: #d8d8d8; BORDER-TOP-COLOR: #d8d8d8; BORDER-RIGHT-COLOR: #d8d8d8; BORDER-LEFT-COLOR: #d8d8d8 } TH { BORDER-BOTTOM-COLOR: #232c5b; BACKGROUND-COLOR: #f2f2f2; BORDER-TOP-COLOR: #232c5b; COLOR: black; BORDER-RIGHT-COLOR: #232c5b; BORDER-LEFT-COLOR: #232c5b } .xdTableHeader { BACKGROUND-COLOR: #f2f2f2; COLOR: black } .light1 { BACKGROUND-COLOR: #ffffff } .dark1 { BACKGROUND-COLOR: #232c5b } .light2 { BACKGROUND-COLOR: #f9f093 } .dark2 { BACKGROUND-COLOR: #603240 } .accent1 { BACKGROUND-COLOR: #ef3372 } .accent2 { BACKGROUND-COLOR: #11aed9 } .accent3 { BACKGROUND-COLOR: #e6cf22 } .accent4 { BACKGROUND-COLOR: #d01010 } .accent5 { BACKGROUND-COLOR: #a7d527 } .accent6 { BACKGROUND-COLOR: #8f2525} </style>
THEME_BITTERSWEET	<style themestyle="urn:office.microsoft.com:themeBittersweet"> TABLE { BORDER-BOTTOM: medium none; BORDER-LEFT: medium none; BORDER-COLLAPSE: collapse; BORDER-TOP: medium none; BORDER-RIGHT: medium none } TD { BORDER-BOTTOM-COLOR: #d8d8d8; BORDER-TOP-COLOR: #d8d8d8; BORDER-RIGHT-COLOR: #d8d8d8; BORDER-LEFT-COLOR: #d8d8d8 } TH { BORDER-BOTTOM-COLOR: #000000; BACKGROUND-COLOR: #f2f2f2; BORDER-TOP-COLOR: #000000; COLOR: black; BORDER-RIGHT-COLOR: #000000; BORDER-LEFT-COLOR: #000000 } .xdTableHeader { BACKGROUND-COLOR: #f2f2f2; COLOR: black } .light1 { BACKGROUND-COLOR: #ffffff } .dark1 { BACKGROUND-COLOR: #000000 } .light2 { BACKGROUND-COLOR: #edb59c } .dark2 { BACKGROUND-COLOR: #c6d199 } .accent1 { BACKGROUND-COLOR: #e2b524 } .accent2 { BACKGROUND-COLOR: #d16c3f } .accent3 { BACKGROUND-COLOR: #7c684d } .accent4 { BACKGROUND-COLOR: #b5a653 } .accent5 { BACKGROUND-COLOR: #d16c3f } .accent6 { BACKGROUND-COLOR: #848058 } </style>
THEME_CAY	<style themestyle="urn:office.microsoft.com:themeCay"> TABLE { BORDER-BOTTOM: medium none; BORDER-LEFT: medium none; BORDER-COLLAPSE: collapse; BORDER-TOP: medium none; BORDER-RIGHT: medium none } TD { BORDER-BOTTOM-COLOR: #d8d8d8; BORDER-TOP-COLOR: #d8d8d8; BORDER-RIGHT-COLOR: #d8d8d8; BORDER-LEFT-COLOR: #d8d8d8 } TH { BORDER-BOTTOM-COLOR: #000000; BACKGROUND-COLOR: #f2f2f2; BORDER-TOP-COLOR: #000000; COLOR: black; BORDER-RIGHT-COLOR: #000000; BORDER-LEFT-COLOR: #000000 } .xdTableHeader { BACKGROUND-COLOR: #f2f2f2; COLOR: black } .light1 { BACKGROUND-COLOR: #ffffff } .dark1 { BACKGROUND-COLOR: #000000 } .light2 { BACKGROUND-COLOR: #f9faf4 } .dark2 { BACKGROUND-COLOR: #42495a } .accent1 { BACKGROUND-COLOR: #3dc8d6 } .accent2 { BACKGROUND-

Theme Style	Settings
	COLOR: #5dc22a } .accent3 { BACKGROUND-COLOR: #e20cc3 } .accent4 { BACKGROUND-COLOR: #f5b709 } .accent5 { BACKGROUND-COLOR: #731997 } .accent6 { BACKGROUND-COLOR: #a5d458 } </style>
THEME_CLASSIC	<style themestyle="urn:office.microsoft.com:themeClassic"> TABLE { BORDER-BOTTOM: medium none; BORDER-LEFT: medium none; BORDER-COLLAPSE: collapse; BORDER-TOP: medium none; BORDER-RIGHT: medium none } TD { BORDER-BOTTOM-COLOR: #d8d8d8; BORDER-TOP-COLOR: #d8d8d8; BORDER-RIGHT-COLOR: #d8d8d8; BORDER-LEFT-COLOR: #d8d8d8 } TH { BORDER-BOTTOM-COLOR: #000000; BACKGROUND-COLOR: #f2f2f2; BORDER-TOP-COLOR: #000000; COLOR: black; BORDER-RIGHT-COLOR: #000000; BORDER-LEFT-COLOR: #000000 } .xdTableHeader { BACKGROUND-COLOR: #f2f2f2; COLOR: black } .light1 { BACKGROUND-COLOR: #ffffff } .dark1 { BACKGROUND-COLOR: #000000 } .light2 { BACKGROUND-COLOR: #f1f5fd } .dark2 { BACKGROUND-COLOR: #002c84 } .accent1 { BACKGROUND-COLOR: #f6aa00 } .accent2 { BACKGROUND-COLOR: #f41e1e } .accent3 { BACKGROUND-COLOR: #568804 } .accent4 { BACKGROUND-COLOR: #0a8faa } .accent5 { BACKGROUND-COLOR: #aa54a2 } .accent6 { BACKGROUND-COLOR: #f47c46 } </style>
THEME_CONSTRUCT	<style themestyle="urn:office.microsoft.com:themeConstruct"> TABLE { BORDER-BOTTOM: medium none; BORDER-LEFT: medium none; BORDER-COLLAPSE: collapse; BORDER-TOP: medium none; BORDER-RIGHT: medium none } TD { BORDER-BOTTOM-COLOR: #d8d8d8; BORDER-TOP-COLOR: #d8d8d8; BORDER-RIGHT-COLOR: #d8d8d8; BORDER-LEFT-COLOR: #d8d8d8 } TH { BORDER-BOTTOM-COLOR: #000000; BACKGROUND-COLOR: #f2f2f2; BORDER-TOP-COLOR: #000000; COLOR: black; BORDER-RIGHT-COLOR: #000000; BORDER-LEFT-COLOR: #000000 } .xdTableHeader { BACKGROUND-COLOR: #f2f2f2; COLOR: black } .light1 { BACKGROUND-COLOR: #ffffff } .dark1 { BACKGROUND-COLOR: #000000 } .light2 { BACKGROUND-COLOR: #f2f3eb } .dark2 { BACKGROUND-COLOR: #666776 } .accent1 { BACKGROUND-COLOR: #87abc9 } .accent2 { BACKGROUND-COLOR: #99957e } .accent3 { BACKGROUND-COLOR: #baa32b } .accent4 { BACKGROUND-COLOR: #f5ac25 } .accent5 { BACKGROUND-COLOR: #91aa9d } .accent6 { BACKGROUND-COLOR: #7c95} </style>
THEME_CONVENTION	<style themestyle="urn:office.microsoft.com:themeConvention"> TABLE { BORDER-BOTTOM: medium none; BORDER-LEFT: medium none; BORDER-COLLAPSE: collapse; BORDER-TOP: medium none; BORDER-RIGHT: medium none } TD { BORDER-BOTTOM-COLOR: #d8d8d8; BORDER-TOP-COLOR: #d8d8d8; BORDER-RIGHT-COLOR: #d8d8d8; BORDER-LEFT-COLOR: #d8d8d8 } TH { BORDER-BOTTOM-COLOR: #3f4746; BACKGROUND-COLOR: #f2f2f2; BORDER-TOP-COLOR: #3f4746; COLOR: black; BORDER-RIGHT-COLOR: #3f4746; BORDER-LEFT-COLOR: #3f4746 } .xdTableHeader { BACKGROUND-COLOR: #f2f2f2; COLOR: black } .light1 { BACKGROUND-COLOR: #ffffff } .dark1 { BACKGROUND-COLOR: #3f4746 } .light2 { BACKGROUND-COLOR: #f0f1ef } .dark2 { BACKGROUND-COLOR: #0c647c } .accent1 { BACKGROUND-COLOR: #b5b919 } .accent2 { BACKGROUND-COLOR: #63acd7 } .accent3 { BACKGROUND-COLOR: #d9570f } .accent4 { BACKGROUND-COLOR: #7d9809 } .accent5 { BACKGROUND-COLOR: #92cac1 } .accent6 { BACKGROUND-COLOR: #e4af01 } </style>

Theme Style	Settings
THEME_FELT	<style themestyle="urn:office.microsoft.com:themeFelt"> TABLE { BORDER-BOTTOM: medium none; BORDER-LEFT: medium none; BORDER-COLLAPSE: collapse; BORDER-TOP: medium none; BORDER-RIGHT: medium none } TD { BORDER-BOTTOM-COLOR: #d8d8d8; BORDER-TOP-COLOR: #d8d8d8; BORDER-RIGHT-COLOR: #d8d8d8; BORDER-LEFT-COLOR: #d8d8d8 } TH { BORDER-BOTTOM-COLOR: #000000; BACKGROUND-COLOR: #f2f2f2; BORDER-TOP-COLOR: #000000; COLOR: black; BORDER-RIGHT-COLOR: #000000; BORDER-LEFT-COLOR: #000000 } .xdTableHeader { BACKGROUND-COLOR: #f2f2f2; COLOR: black } .light1 { BACKGROUND-COLOR: #ffffff } .dark1 { BACKGROUND-COLOR: #000000 } .light2 { BACKGROUND-COLOR: #e8eeee } .dark2 { BACKGROUND-COLOR: #919649 } .accent1 { BACKGROUND-COLOR: #ffad1c } .accent2 { BACKGROUND-COLOR: #a5a5a5 } .accent3 { BACKGROUND-COLOR: #7c684d } .accent4 { BACKGROUND-COLOR: #b5a653 } .accent5 { BACKGROUND-COLOR: #d16c3f } .accent6 { BACKGROUND-COLOR: #848058 } </style>
THEME_GRAHAM	<style themestyle="urn:office.microsoft.com:themeGraham"> TABLE { BORDER-BOTTOM: medium none; BORDER-LEFT: medium none; BORDER-COLLAPSE: collapse; BORDER-TOP: medium none; BORDER-RIGHT: medium none } TD { BORDER-BOTTOM-COLOR: #ffe970; BORDER-TOP-COLOR: #ffe970; BORDER-RIGHT-COLOR: #ffe970; BORDER-LEFT-COLOR: #ffe970 } TH { BORDER-BOTTOM-COLOR: #4e2d02; BACKGROUND-COLOR: #fef4ba; BORDER-TOP-COLOR: #4e2d02; COLOR: black; BORDER-RIGHT-COLOR: #4e2d02; BORDER-LEFT-COLOR: #4e2d02 } .xdTableHeader { BACKGROUND-COLOR: #fef4ba; COLOR: black } .light1 { BACKGROUND-COLOR: #fffceb } .dark1 { BACKGROUND-COLOR: #4e2d02 } .light2 { BACKGROUND-COLOR: #fff8e3 } .dark2 { BACKGROUND-COLOR: #573e25 } .accent1 { BACKGROUND-COLOR: #d27800 } .accent2 { BACKGROUND-COLOR: #ae7c4b } .accent3 { BACKGROUND-COLOR: #ff0000 } .accent4 { BACKGROUND-COLOR: #febc16 } .accent5 { BACKGROUND-COLOR: #f9a02d } .accent6 { BACKGROUND-COLOR: #f5d000} </style>
THEME_GRAPELLO	<style themestyle="urn:office.microsoft.com:themeGrapello"> TABLE { BORDER-BOTTOM: medium none; BORDER-LEFT: medium none; BORDER-COLLAPSE: collapse; BORDER-TOP: medium none; BORDER-RIGHT: medium none } TD { BORDER-BOTTOM-COLOR: #d8d8d8; BORDER-TOP-COLOR: #d8d8d8; BORDER-RIGHT-COLOR: #d8d8d8; BORDER-LEFT-COLOR: #d8d8d8 } TH { BORDER-BOTTOM-COLOR: #000000; BACKGROUND-COLOR: #f2f2f2; BORDER-TOP-COLOR: #000000; COLOR: black; BORDER-RIGHT-COLOR: #000000; BORDER-LEFT-COLOR: #000000 } .xdTableHeader { BACKGROUND-COLOR: #f2f2f2; COLOR: black } .light1 { BACKGROUND-COLOR: #ffffff } .dark1 { BACKGROUND-COLOR: #000000 } .light2 { BACKGROUND-COLOR: #f7f6f4 } .dark2 { BACKGROUND-COLOR: #3f3f3f } .accent1 { BACKGROUND-COLOR: #903cd6 } .accent2 { BACKGROUND-COLOR: #9d86b8 } .accent3 { BACKGROUND-COLOR: #c0b9b5 } .accent4 { BACKGROUND-COLOR: #d6b8f0 } .accent5 { BACKGROUND-COLOR: #f7e797 } .accent6 { BACKGROUND-COLOR: #f0d23c } </style>
THEME_LAMINATE	<style themestyle="urn:office.microsoft.com:themeLaminate"> TABLE { BORDER-BOTTOM: medium none; BORDER-LEFT: medium none;

Theme Style	Settings
	BORDER-COLLAPSE: collapse; BORDER-TOP: medium none; BORDER-RIGHT: medium none } TD { BORDER-BOTTOM-COLOR: #d8d8d8; BORDER-TOP-COLOR: #d8d8d8; BORDER-RIGHT-COLOR: #d8d8d8; BORDER-LEFT-COLOR: #d8d8d8 } TH { BORDER-BOTTOM-COLOR: #000000; BACKGROUND-COLOR: #f2f2f2; BORDER-TOP-COLOR: #000000; COLOR: black; BORDER-RIGHT-COLOR: #000000; BORDER-LEFT-COLOR: #000000 } .xdTableHeader { BACKGROUND-COLOR: #f2f2f2; COLOR: black } .light1 { BACKGROUND-COLOR: #ffffff } .dark1 { BACKGROUND-COLOR: #000000 } .light2 { BACKGROUND-COLOR: #d9dfdd } .dark2 { BACKGROUND-COLOR: #664d4a } .accent1 { BACKGROUND-COLOR: #bb7384 } .accent2 { BACKGROUND-COLOR: #93a299 } .accent3 { BACKGROUND-COLOR: #b5ae53 } .accent4 { BACKGROUND-COLOR: #848058 } .accent5 { BACKGROUND-COLOR: #e8b54d } .accent6 { BACKGROUND-COLOR: #786c71 } </style>
THEME_MISSION	<style themestyle="urn:office.microsoft.com:themeModule"> TABLE { BORDER-BOTTOM: medium none; BORDER-LEFT: medium none; BORDER-COLLAPSE: collapse; BORDER-TOP: medium none; BORDER-RIGHT: medium none } TD { BORDER-BOTTOM-COLOR: #d8d8d8; BORDER-TOP-COLOR: #d8d8d8; BORDER-RIGHT-COLOR: #d8d8d8; BORDER-LEFT-COLOR: #d8d8d8 } TH { BORDER-BOTTOM-COLOR: #000000; BACKGROUND-COLOR: #f2f2f2; BORDER-TOP-COLOR: #000000; COLOR: black; BORDER-RIGHT-COLOR: #000000; BORDER-LEFT-COLOR: #000000 } .xdTableHeader { BACKGROUND-COLOR: #f2f2f2; COLOR: black } .light1 { BACKGROUND-COLOR: #ffffff } .dark1 { BACKGROUND-COLOR: #000000 } .light2 { BACKGROUND-COLOR: #f0e5d3 } .dark2 { BACKGROUND-COLOR: #3c2902 } .accent1 { BACKGROUND-COLOR: #4c6c22 } .accent2 { BACKGROUND-COLOR: #f3ba32 } .accent3 { BACKGROUND-COLOR: #97371e } .accent4 { BACKGROUND-COLOR: #e84a30 } .accent5 { BACKGROUND-COLOR: #d2b712 } .accent6 { BACKGROUND-COLOR: #6c4444 } </style>
THEME_MODERNROSE	<style themestyle="urn:office.microsoft.com:themeModernrose"> TABLE { BORDER-BOTTOM: medium none; BORDER-LEFT: medium none; BORDER-COLLAPSE: collapse; BORDER-TOP: medium none; BORDER-RIGHT: medium none } TD { BORDER-BOTTOM-COLOR: #d8d8d8; BORDER-TOP-COLOR: #d8d8d8; BORDER-RIGHT-COLOR: #d8d8d8; BORDER-LEFT-COLOR: #d8d8d8 } TH { BORDER-BOTTOM-COLOR: #000000; BACKGROUND-COLOR: #f2f2f2; BORDER-TOP-COLOR: #000000; COLOR: black; BORDER-RIGHT-COLOR: #000000; BORDER-LEFT-COLOR: #000000 } .xdTableHeader { BACKGROUND-COLOR: #f2f2f2; COLOR: black } .light1 { BACKGROUND-COLOR: #ffffff } .dark1 { BACKGROUND-COLOR: #000000 } .light2 { BACKGROUND-COLOR: #f8f9fb } .dark2 { BACKGROUND-COLOR: #424c60 } .accent1 { BACKGROUND-COLOR: #ff006c } .accent2 { BACKGROUND-COLOR: #014aed } .accent3 { BACKGROUND-COLOR: #00d05e } .accent4 { BACKGROUND-COLOR: #e9aa0d } .accent5 { BACKGROUND-COLOR: #3db2ff } .accent6 { BACKGROUND-COLOR: #7030a0} </style>
THEME_MUNICIPAL	<style themestyle="urn:office.microsoft.com:themeMunicipal"> TABLE { BORDER-BOTTOM: medium none; BORDER-LEFT: medium none; BORDER-COLLAPSE: collapse; BORDER-TOP: medium none; BORDER-RIGHT: medium none } TD { BORDER-BOTTOM-COLOR: #d8d8d8; BORDER-TOP-COLOR: #d8d8d8; BORDER-RIGHT-COLOR: #d8d8d8; BORDER-LEFT-COLOR: #d8d8d8 } TH { BORDER-BOTTOM-COLOR: #000000; BACKGROUND-

Theme Style	Settings
	COLOR: #f2f2f2; BORDER-TOP-COLOR: #000000; COLOR: black; BORDER-RIGHT-COLOR: #000000; BORDER-LEFT-COLOR: #000000 } .xdTableHeader { BACKGROUND-COLOR: #f2f2f2; COLOR: black } .light1 { BACKGROUND-COLOR: #ffffff } .dark1 { BACKGROUND-COLOR: #000000 } .light2 { BACKGROUND-COLOR: #f0efea } .dark2 { BACKGROUND-COLOR: #5e5a5a } .accent1 { BACKGROUND-COLOR: #d34817 } .accent2 { BACKGROUND-COLOR: #9b2d1f } .accent3 { BACKGROUND-COLOR: #a28e6a } .accent4 { BACKGROUND-COLOR: #9b694b } .accent5 { BACKGROUND-COLOR: #918485 } .accent6 { BACKGROUND-COLOR: #855d5d } </style>
THEME_PINNATE	<style themestyle="urn:office.microsoft.com:themePinnate"> TABLE { BORDER-BOTTOM: medium none; BORDER-LEFT: medium none; BORDER-COLLAPSE: collapse; BORDER-TOP: medium none; BORDER-RIGHT: medium none } TD { BORDER-BOTTOM-COLOR: #d8d8d8; BORDER-TOP-COLOR: #d8d8d8; BORDER-RIGHT-COLOR: #d8d8d8; BORDER-LEFT-COLOR: #d8d8d8 } TH { BORDER-BOTTOM-COLOR: #000000; BACKGROUND-COLOR: #f2f2f2; BORDER-TOP-COLOR: #000000; COLOR: black; BORDER-RIGHT-COLOR: #000000; BORDER-LEFT-COLOR: #000000 } .xdTableHeader { BACKGROUND-COLOR: #f2f2f2; COLOR: black } .light1 { BACKGROUND-COLOR: #ffffff } .dark1 { BACKGROUND-COLOR: #000000 } .light2 { BACKGROUND-COLOR: #f9f8f6 } .dark2 { BACKGROUND-COLOR: #292309 } .accent1 { BACKGROUND-COLOR: #95a201 } .accent2 { BACKGROUND-COLOR: #4a5100 } .accent3 { BACKGROUND-COLOR: #e65f3b } .accent4 { BACKGROUND-COLOR: #315c6a } .accent5 { BACKGROUND-COLOR: #78aec0 } .accent6 { BACKGROUND-COLOR: #eeb415 } </style>
THEME_RICASSO	<style themestyle="urn:office.microsoft.com:themeRicasso"> TABLE { BORDER-BOTTOM: medium none; BORDER-LEFT: medium none; BORDER-COLLAPSE: collapse; BORDER-TOP: medium none; BORDER-RIGHT: medium none } TD { BORDER-BOTTOM-COLOR: #d8d8d8; BORDER-TOP-COLOR: #d8d8d8; BORDER-RIGHT-COLOR: #d8d8d8; BORDER-LEFT-COLOR: #d8d8d8 } TH { BORDER-BOTTOM-COLOR: #000000; BACKGROUND-COLOR: #f2f2f2; BORDER-TOP-COLOR: #000000; COLOR: black; BORDER-RIGHT-COLOR: #000000; BORDER-LEFT-COLOR: #000000 } .xdTableHeader { BACKGROUND-COLOR: #f2f2f2; COLOR: black } .light1 { BACKGROUND-COLOR: #ffffff } .dark1 { BACKGROUND-COLOR: #000000 } .light2 { BACKGROUND-COLOR: #fbfbfb } .dark2 { BACKGROUND-COLOR: #525252 } .accent1 { BACKGROUND-COLOR: #000000 } .accent2 { BACKGROUND-COLOR: #b2b2b2 } .accent3 { BACKGROUND-COLOR: #969696 } .accent4 { BACKGROUND-COLOR: #c00000 } .accent5 { BACKGROUND-COLOR: #6d6d6d } .accent6 { BACKGROUND-COLOR: #5a5a5a } </style>
THEME_SUMMER	<style themestyle="urn:office.microsoft.com:themeSummer"> TABLE { BORDER-BOTTOM: medium none; BORDER-LEFT: medium none; BORDER-COLLAPSE: collapse; BORDER-TOP: medium none; BORDER-RIGHT: medium none } TD { BORDER-BOTTOM-COLOR: #d8d8d8; BORDER-TOP-COLOR: #d8d8d8; BORDER-RIGHT-COLOR: #d8d8d8; BORDER-LEFT-COLOR: #d8d8d8 } TH { BORDER-BOTTOM-COLOR: #000000; BACKGROUND-COLOR: #f2f2f2; BORDER-TOP-COLOR: #000000; COLOR: black; BORDER-RIGHT-COLOR: #000000; BORDER-LEFT-COLOR: #000000 } .xdTableHeader { BACKGROUND-COLOR: #f2f2f2; COLOR: black } .light1 { BACKGROUND-COLOR: #ffffff } .dark1 { BACKGROUND-COLOR: #000000 } .light2 { BACKGROUND-COLOR: #fbfbf9 } .dark2 { BACKGROUND-COLOR: #2b4b4d } .accent1 { BACKGROUND-COLOR: #6c9a7f } .accent2 { BACKGROUND-

Theme Style	Settings
	COLOR: #bb523d } .accent3 { BACKGROUND-COLOR: #c89d11 } .accent4 { BACKGROUND-COLOR: #fccf10 } .accent5 { BACKGROUND-COLOR: #568ea1 } .accent6 { BACKGROUND-COLOR: #decf28} </style>
THEME_VANTAGE	<style themestyle="urn:office.microsoft.com:themeVantage"> TABLE { BORDER-BOTTOM: medium none; BORDER-LEFT: medium none; BORDER-COLLAPSE: collapse; BORDER-TOP: medium none; BORDER-RIGHT: medium none } TD { BORDER-BOTTOM-COLOR: #d8d8d8; BORDER-TOP-COLOR: #d8d8d8; BORDER-RIGHT-COLOR: #d8d8d8; BORDER-LEFT-COLOR: #d8d8d8 } TH { BORDER-BOTTOM-COLOR: #000000; BACKGROUND-COLOR: #f2f2f2; BORDER-TOP-COLOR: #000000; COLOR: black; BORDER-RIGHT-COLOR: #000000; BORDER-LEFT-COLOR: #000000 } .xdTableHeader { BACKGROUND-COLOR: #f2f2f2; COLOR: black } .light1 { BACKGROUND-COLOR: #ffffff } .dark1 { BACKGROUND-COLOR: #000000 } .light2 { BACKGROUND-COLOR: #dff1fd } .dark2 { BACKGROUND-COLOR: #4c5b72 } .accent1 { BACKGROUND-COLOR: #ffb219 } .accent2 { BACKGROUND-COLOR: #ea157a } .accent3 { BACKGROUND-COLOR: #92d050 } .accent4 { BACKGROUND-COLOR: #00addc } .accent5 { BACKGROUND-COLOR: #738ac8 } .accent6 { BACKGROUND-COLOR: #1ab39f } </style>
THEME_VIEWPOINT	<style themestyle="urn:office.microsoft.com:themeViewpoint"> TABLE { BORDER-BOTTOM: medium none; BORDER-LEFT: medium none; BORDER-COLLAPSE: collapse; BORDER-TOP: medium none; BORDER-RIGHT: medium none } TD { BORDER-BOTTOM-COLOR: #d8d8d8; BORDER-TOP-COLOR: #d8d8d8; BORDER-RIGHT-COLOR: #d8d8d8; BORDER-LEFT-COLOR: #d8d8d8 } TH { BORDER-BOTTOM-COLOR: #000000; BACKGROUND-COLOR: #f2f2f2; BORDER-TOP-COLOR: #000000; COLOR: black; BORDER-RIGHT-COLOR: #000000; BORDER-LEFT-COLOR: #000000 } .xdTableHeader { BACKGROUND-COLOR: #f2f2f2; COLOR: black } .light1 { BACKGROUND-COLOR: #ffffff } .dark1 { BACKGROUND-COLOR: #000000 } .light2 { BACKGROUND-COLOR: #f2f8ee } .dark2 { BACKGROUND-COLOR: #43282f } .accent1 { BACKGROUND-COLOR: #96a528 } .accent2 { BACKGROUND-COLOR: #e4ad01 } .accent3 { BACKGROUND-COLOR: #ad7d4c } .accent4 { BACKGROUND-COLOR: #c76d00 } .accent5 { BACKGROUND-COLOR: #bfb24c } .accent6 { BACKGROUND-COLOR: #9c9768 } </style>
THEME_YOSHI	<style themestyle="urn:office.microsoft.com:themeYoshi"> TABLE { BORDER-BOTTOM: medium none; BORDER-LEFT: medium none; BORDER-COLLAPSE: collapse; BORDER-TOP: medium none; BORDER-RIGHT: medium none } TD { BORDER-BOTTOM-COLOR: #d8d8d8; BORDER-TOP-COLOR: #d8d8d8; BORDER-RIGHT-COLOR: #d8d8d8; BORDER-LEFT-COLOR: #d8d8d8 } TH { BORDER-BOTTOM-COLOR: #000000; BACKGROUND-COLOR: #f2f2f2; BORDER-TOP-COLOR: #000000; COLOR: black; BORDER-RIGHT-COLOR: #000000; BORDER-LEFT-COLOR: #000000 } .xdTableHeader { BACKGROUND-COLOR: #f2f2f2; COLOR: black } .light1 { BACKGROUND-COLOR: #ffffff } .dark1 { BACKGROUND-COLOR: #000000 } .light2 { BACKGROUND-COLOR: #ffffff } .dark2 { BACKGROUND-COLOR: #000000 } .accent1 { BACKGROUND-COLOR: #ea7b20 } .accent2 { BACKGROUND-COLOR: #a6192c } .accent3 { BACKGROUND-COLOR: #e5d00d } .accent4 { BACKGROUND-COLOR: #f599a8 } .accent5 { BACKGROUND-COLOR: #368030 } .accent6 { BACKGROUND-COLOR: #683912 } </style>

PRETTYFORMS_TABLE_STYLE: MUST have one of the values in the following table.

Table Style	Settings
TABLE_STYLE_PROFESSIONAL	<style tablestyle="Professional"> TR.xdTitleRow { min-height: 83px; } TD.xdTitleCell { PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 14px; PADDING-TOP: 32px; BORDER-LEFT: PROFESSIONAL_STYLE_COLOR3 1pt solid; BORDER-RIGHT: PROFESSIONAL_STYLE_COLOR3 1pt solid; BORDER-TOP: PROFESSIONAL_STYLE_COLOR3 1pt solid; background-color: PROFESSIONAL_STYLE_COLOR7; valign: bottom; text-align: center } TR.xdTitleRowWithHeading { min-height: 69px; } TD.xdTitleCellWithHeading { PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 0px; PADDING-TOP: 32px; BORDER-LEFT: PROFESSIONAL_STYLE_COLOR3 1pt solid; BORDER-RIGHT: PROFESSIONAL_STYLE_COLOR3 1pt solid; BORDER-TOP: PROFESSIONAL_STYLE_COLOR3 1pt solid; background-color: PROFESSIONAL_STYLE_COLOR7; valign: bottom; text-align: center } TR.xdTitleRowWithSubHeading { min-height: 75px; } TD.xdTitleCellWithSubHeading { PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 6px; PADDING-TOP: 32px; BORDER-LEFT: PROFESSIONAL_STYLE_COLOR3 1pt solid; BORDER-RIGHT: PROFESSIONAL_STYLE_COLOR3 1pt solid; BORDER-TOP: PROFESSIONAL_STYLE_COLOR3 1pt solid; background-color: PROFESSIONAL_STYLE_COLOR7; valign: bottom; text-align: center } TR.xdTitleRowWithOffsetBody { min-height: 72px; } TD.xdTitleCellWithOffsetBody { PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 2px; PADDING-TOP: 32px; BORDER-LEFT: PROFESSIONAL_STYLE_COLOR3 1pt solid; BORDER-RIGHT: PROFESSIONAL_STYLE_COLOR3 1pt solid; BORDER-TOP: PROFESSIONAL_STYLE_COLOR3 1pt solid; background-color: PROFESSIONAL_STYLE_COLOR7; valign: bottom; text-align: left } TR.xdTitleHeadingRow { min-height: 37px; } TD.xdTitleHeadingCell { PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 14px; PADDING-TOP: 0px; BORDER-LEFT: PROFESSIONAL_STYLE_COLOR3 1pt solid; BORDER-RIGHT:

Table Style	Settings

	PROFESSIONAL_STYLE_COLOR3 1pt solid; background-color: PROFESSIONAL_STYLE_COLOR7; valign: top; text-align: center
	}
	TR.xdTitleSubheadingRow {
	min-height: 70px;
	}
	TD.xdTitleSubheadingCell {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 16px; PADDING-TOP: 8px; BORDER-LEFT: PROFESSIONAL_STYLE_COLOR3 1pt solid; BORDER-RIGHT: PROFESSIONAL_STYLE_COLOR3 1pt solid; background-color: PROFESSIONAL_STYLE_COLOR7; valign: top
	}
	TD.xdVerticalFill {
	BORDER-LEFT: PROFESSIONAL_STYLE_COLOR3 1pt solid; BORDER-BOTTOM: PROFESSIONAL_STYLE_COLOR3 1pt solid; BORDER-TOP: PROFESSIONAL_STYLE_COLOR3 1pt solid; BORDER-LEFT: PROFESSIONAL_STYLE_COLOR3 1pt solid; background-color: PROFESSIONAL_STYLE_COLOR1
	}
	TD.xdTableContentCellWithVerticalOffset {
	BORDER-RIGHT-COLOR: PROFESSIONAL_STYLE_COLOR3; BORDER-RIGHT-WIDTH:1pt; BORDER-RIGHT-STYLE: solid; BORDER-LEFT-COLOR: PROFESSIONAL_STYLE_COLOR3; BORDER-LEFT-WIDTH:1pt; BORDER-LEFT-STYLE: solid; BORDER-BOTTOM-COLOR: PROFESSIONAL_STYLE_COLOR3; BORDER-BOTTOM-WIDTH:1pt; BORDER-BOTTOM-STYLE: solid; background-color: PROFESSIONAL_STYLE_COLOR7; PADDING-RIGHT: 0px; PADDING-LEFT: 95px; PADDING-BOTTOM: 2px; PADDING-TOP: 32px; valign: bottom; text-align: left
	}
	TR.xdTableContentRow {
	min-height: 140px;
	}
	TD.xdTableContentCell {
	padding: 0px 0px 0px 0px; BORDER-LEFT: PROFESSIONAL_STYLE_COLOR3 1pt solid; BORDER-RIGHT: PROFESSIONAL_STYLE_COLOR3 1pt solid; BORDER-BOTTOM: PROFESSIONAL_STYLE_COLOR3 1pt solid; background-color: PROFESSIONAL_STYLE_COLOR7; valign: top
	}
	TD.xdTableContentCellWithVerticalFill {
	padding: 0px 0px 0px 0px; BORDER-LEFT: PROFESSIONAL_STYLE_COLOR3 1pt solid; BORDER-RIGHT: PROFESSIONAL_STYLE_COLOR3 1pt solid; BORDER-BOTTOM: PROFESSIONAL_STYLE_COLOR3 1pt solid; background-color: PROFESSIONAL_STYLE_COLOR7; valign: top
	}
	TD.xdTableStyleOneCol {
	padding: 4px 22px 4px 22px;
	}
	TR.xdContentRowOneCol {
	min-height: 45px; valign: center
	}
	TR.xdHeadingRow {
	min-height: 27px;
	}

Table Style	Settings

	TD.xdHeadingCell {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 2px; PADDING-TOP: 2px; border-top-color: PROFESSIONAL_STYLE_COLOR4; border-top-width: 1pt; border-top-style: solid; border-bottom-color: PROFESSIONAL_STYLE_COLOR4; border-bottom-width: 1pt; border-bottom-style: solid; background-color: PROFESSIONAL_STYLE_COLOR5; valign: bottom; text-align: center
	}
	TR.xdSubheadingRow {
	min-height: 28px;
	}
	TD.xdSubheadingCell {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 4px; PADDING-TOP: 4px; border-bottom-color: PROFESSIONAL_STYLE_COLOR4; border-bottom-width: 1pt; border-bottom-style: solid; valign: bottom; text-align: center
	}
	TR.xdHeadingRowEmphasis {
	min-height: 27px;
	}
	TD.xdHeadingCellEmphasis {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 2px; PADDING-TOP: 2px; border-top-color: PROFESSIONAL_STYLE_COLOR4; border-top-width: 1pt; border-top-style: solid; border-bottom-color: PROFESSIONAL_STYLE_COLOR4; border-bottom-width: 1pt; border-bottom-style: solid; background-color: PROFESSIONAL_STYLE_COLOR5; valign: bottom; text-align: center
	}
	TR.xdSubheadingRowEmphasis {
	min-height: 28px;
	}
	TD.xdSubheadingCellEmphasis {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 4px; PADDING-TOP: 4px; border-bottom-color: PROFESSIONAL_STYLE_COLOR4; border-bottom-width: 1pt; border-bottom-style: solid; valign: bottom; text-align: center
	}
	TR.xdTableLabelControlStackedRow {
	min-height: 45px;
	}
	TD.xdTableLabelControlStackedCellLabel {
	padding: 4px 5px 4px 22px;
	}
	TD.xdTableLabelControlStackedCellComponent {
	padding: 4px 22px 4px 5px;
	}
	TR.xdTableRow {
	min-height: 30px;
	}
	TD.xdTableCellLabel {
	padding-top: 4px; padding-bottom: 4px; padding-right: 5px; padding-left: 22px;

Table Style	Settings

	}
	TD.xdTableCellComponent {
	padding-top: 4px; padding-bottom: 4px; padding-right: 22px; padding-left: 5px;
	}
	TD.xdTableMiddleCell {
	padding-top: 4px; padding-bottom: 4px; padding-right: 5px; padding-left: 5px;
	}
	TR.xdTableEmphasisRow {
	min-height: 30px;
	}
	TD.xdTableEmphasisCellLabel {
	padding-top: 4px; padding-bottom: 4px; padding-right: 5px; padding-left: 22px; background-color: PROFESSIONAL_STYLE_COLOR6;
	}
	TD.xdTableEmphasisCellComponent {
	padding-top: 4px; padding-bottom: 4px; padding-right: 22px; padding-left: 5px; background-color: PROFESSIONAL_STYLE_COLOR6;
	}
	TD.xdTableMiddleCellEmphasis {
	padding-top: 4px; padding-bottom: 4px; padding-right: 5px; padding-left: 5px; background-color: PROFESSIONAL_STYLE_COLOR6;
	}
	TR.xdTableOffsetRow {
	min-height: 30px;
	}
	TD.xdTableOffsetCellLabel {
	PADDING-RIGHT: 5px; PADDING-LEFT: 22px; PADDING-BOTTOM: 4px; PADDING-TOP: 4px; background-color: PROFESSIONAL_STYLE_COLOR6;
	}
	TD.xdTableOffsetCellComponent {
	background-color: PROFESSIONAL_STYLE_COLOR6; PADDING-RIGHT: 22px; PADDING-LEFT: 5px; PADDING-BOTTOM: 4px; PADDING-TOP: 4px;
	}
	P {
	margin-top: 0px; color: PROFESSIONAL_STYLE_COLOR1; font-size: 11pt
	}
	h1 {
	color: PROFESSIONAL_STYLE_COLOR1; margin-top:0px;margin-bottom:0px;font-size: 24pt;font-weight: normal;
	}
	h2 {
	color: PROFESSIONAL_STYLE_COLOR1; margin-top:0px;margin-bottom:0px;font-size:16pt;font-weight: bold
	}
	h3 {
	color: PROFESSIONAL_STYLE_COLOR1; margin-top:0px;margin-bottom:0px; font-size:12pt; font-weight: normal; text-transform: uppercase
	}

Table Style	Settings

	h4 {
	color: PROFESSIONAL_STYLE_COLOR2;margin-top:0px;margin-bottom:0px;font-size:10pt; font-style: italic; font-weight: normal
	}
	h5 {
	color: PROFESSIONAL_STYLE_COLOR1;margin-top:0px;margin-bottom:0px; font-size:10pt;font-style: italic; font-weight: bold
	}
	h6 {
	color: PROFESSIONAL_STYLE_COLOR2;margin-top:0px;margin-bottom:0px; font-size:10pt;font-weight: normal
	}
	Body {
	color: black;
	}
	</style>

TABLE_STYLE_INDUSTRIAL	<style tablestyle="Industrial"> TR.xdTitleRow {
	min-height: 68px;}
	TD.xdTitleCell {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 2px; PADDING-TOP: 32px; BORDER-LEFT: INDUSTRIAL_STYLE_COLOR5 1pt solid; BORDER-RIGHT: INDUSTRIAL_STYLE_COLOR5 1pt solid; BORDER-TOP: INDUSTRIAL_STYLE_COLOR5 1pt solid; background-color: INDUSTRIAL_STYLE_COLOR6; valign:bottom
	}
	TR.xdTitleRowWithHeading {
	min-height: 62px;
	}
	TD.xdTitleCellWithHeading {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 0px; PADDING-TOP: 32px; BORDER-LEFT: INDUSTRIAL_STYLE_COLOR5 1pt solid; BORDER-RIGHT: INDUSTRIAL_STYLE_COLOR5 1pt solid; BORDER-TOP: INDUSTRIAL_STYLE_COLOR5 1pt solid; background-color: INDUSTRIAL_STYLE_COLOR6; valign:bottom
	}
	TR.xdTitleRowWithSubHeading {
	min-height: 68px;
	}
	TD.xdTitleCellWithSubHeading {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 6px; PADDING-TOP: 32px; BORDER-LEFT: INDUSTRIAL_STYLE_COLOR5 1pt solid; BORDER-RIGHT: INDUSTRIAL_STYLE_COLOR5 1pt solid; BORDER-TOP: INDUSTRIAL_STYLE_COLOR5 1pt solid; background-color: INDUSTRIAL_STYLE_COLOR6; valign:bottom
	}
	TR.xdTitleRowWithOffsetBody {
	min-height: 68px;
	}
	TD.xdTitleCellWithOffsetBody {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 2px; PADDING-TOP: 32px; BORDER-LEFT:

Table Style	Settings

	INDUSTRIAL_STYLE_COLOR5 1pt solid; BORDER-RIGHT: INDUSTRIAL_STYLE_COLOR5 1pt solid; BORDER-TOP: INDUSTRIAL_STYLE_COLOR5 1pt solid; background-color: INDUSTRIAL_STYLE_COLOR6; valign:bottom
	}
	TR.xdTitleHeadingRow {
	min-height: 31px;}
	TD.xdTitleHeadingCell {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 6px; PADDING-TOP: 2px; BORDER-LEFT: INDUSTRIAL_STYLE_COLOR5 1pt solid; BORDER-RIGHT: INDUSTRIAL_STYLE_COLOR5 1pt solid; border-bottom-color: INDUSTRIAL_STYLE_COLOR2; border-bottom-width: 1pt; border-bottom-style: solid; background-color: INDUSTRIAL_STYLE_COLOR6; valign: top
	}
	TR.xdTitleSubheadingRow {
	min-height: 70px;
	}
	TD.xdTitleSubheadingCell {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 16px; PADDING-TOP: 8px; BORDER-LEFT: INDUSTRIAL_STYLE_COLOR5 1pt solid; BORDER-RIGHT: INDUSTRIAL_STYLE_COLOR5 1pt solid; background-color: INDUSTRIAL_STYLE_COLOR6; valign:top
	}
	TD.xdVerticalFill {
	BORDER-LEFT: INDUSTRIAL_STYLE_COLOR5 1pt solid; BORDER-BOTTOM: INDUSTRIAL_STYLE_COLOR5 1pt solid; BORDER-TOP: INDUSTRIAL_STYLE_COLOR5 1pt solid; background-color: INDUSTRIAL_STYLE_COLOR3
	}
	TD.xdTableContentCellWithVerticalOffset {
	BORDER-LEFT: INDUSTRIAL_STYLE_COLOR5 1pt solid; BORDER-RIGHT: INDUSTRIAL_STYLE_COLOR5 1pt solid; BORDER-BOTTOM: INDUSTRIAL_STYLE_COLOR5 1pt solid; background-color: INDUSTRIAL_STYLE_COLOR6; PADDING-RIGHT: 0px; PADDING-LEFT: 95px; PADDING-BOTTOM: 2px; PADDING-TOP: 12px; valign: bottom
	}
	TR.xdTableContentRow {
	min-height: 140px;
	}
	TD.xdTableContentCell {
	padding:0px 0px 0px 0px; BORDER-LEFT: INDUSTRIAL_STYLE_COLOR5 1pt solid; BORDER-RIGHT: INDUSTRIAL_STYLE_COLOR5 1pt solid; BORDER-BOTTOM: INDUSTRIAL_STYLE_COLOR5 1pt solid; background-color: INDUSTRIAL_STYLE_COLOR6; valign: top
	}
	TD.xdTableContentCellWithVerticalFill {
	padding:0px 0px 0px 0px; BORDER-LEFT: INDUSTRIAL_STYLE_COLOR5 1pt solid; BORDER-RIGHT: INDUSTRIAL_STYLE_COLOR5 1pt solid; BORDER-BOTTOM: INDUSTRIAL_STYLE_COLOR5 1pt solid; background-color: INDUSTRIAL_STYLE_COLOR6; valign: top
	}
	TD.xdTableStyleOneCol {

Table Style	Settings

	padding: 4px 22px 4px 22px;
	}
	TR.xdContentRowOneCol {
	min-height:45px; valign:center
	}
	TR.xdHeadingRow {
	min-height: 43px;
	}
	TD.xdHeadingCell {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 6px; PADDING-TOP: 12px; border-bottom-color: INDUSTRIAL_STYLE_COLOR3; border-bottom-width: 2.25pt; border-bottom-style: solid; valign: bottom
	}
	TR.xdSubheadingRow {
	min-height: 28px;
	}
	TD.xdSubheadingCell {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 4px; PADDING-TOP: 4px; border-bottom-color: INDUSTRIAL_STYLE_COLOR2; border-bottom-width: 1pt; border-bottom-style: solid; background-color: INDUSTRIAL_STYLE_COLOR5; valign: bottom
	}
	TR.xdHeadingRowEmphasis {
	min-height: 43px;
	}
	TD.xdHeadingCellEmphasis {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 6px; PADDING-TOP: 12px; border-bottom-color: INDUSTRIAL_STYLE_COLOR3; border-bottom-width: 2.25pt; border-bottom-style: solid; valign: bottom
	}
	TR.xdSubheadingRowEmphasis {
	min-height: 28px;
	}
	TD.xdSubheadingCellEmphasis {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 4px; PADDING-TOP: 4px; border-bottom-color: INDUSTRIAL_STYLE_COLOR2; border-bottom-width: 1pt; border-bottom-style: solid; background-color: INDUSTRIAL_STYLE_COLOR5; valign: bottom
	}
	TR.xdTableLabelControlStackedRow {
	min-height: 45px;
	}
	TD.xdTableLabelControlStackedCellLabel {
	padding: 4px 5px 4px 22px;
	}
	TD.xdTableLabelControlStackedCellComponent {
	padding: 4x 22px 4px 5px;
	}
	TR.xdTableRow {
	min-height: 30px;
	}
	TD.xdTableCellLabel {
	padding-top: 4px; padding-bottom: 4px; padding-right:

Table Style	Settings

	5px; padding-left: 22px;
	}
	TD.xdTableCellComponent {
	padding-top: 4px; padding-bottom: 4px; padding-right: 22px; padding-left: 5px;
	}
	TD.xdTableMiddleCell {
	padding-top: 4px; padding-bottom: 4px; padding-right: 5px; padding-left: 5px;
	}
	TR.xdTableEmphasisRow {
	min-height: 30px;
	}
	TD.xdTableEmphasisCellLabel {
	padding-top: 4px; padding-bottom: 4px; padding-right: 5px; padding-left: 22px; background-color: INDUSTRIAL_STYLE_COLOR5;
	}
	TD.xdTableEmphasisCellComponent {
	padding-top: 4px; padding-bottom: 4px; padding-right: 22px; padding-left: 5px; background-color: INDUSTRIAL_STYLE_COLOR5;}
	TD.xdTableMiddleCellEmphasis {
	padding-top: 4px; padding-bottom: 4px; padding-right: 5px; padding-left: 5px; background-color: INDUSTRIAL_STYLE_COLOR5;}
	TR.xdTableOffsetRow {
	min-height: 30px;
	}
	TD.xdTableOffsetCellLabel {
	PADDING-RIGHT: 5px; PADDING-LEFT: 22px; PADDING-BOTTOM: 4px; PADDING-TOP: 4px; background-color: INDUSTRIAL_STYLE_COLOR5;}
	TD.xdTableOffsetCellComponent {
	background-color: INDUSTRIAL_STYLE_COLOR5; PADDING-RIGHT: 22px; PADDING-LEFT: 5px; PADDING-BOTTOM: 4px; PADDING-TOP: 4px;
	}
	P {
	margin-top:0px; color:INDUSTRIAL_STYLE_COLOR1; font-size:11pt
	}
	h1 {
	color:INDUSTRIAL_STYLE_COLOR1;margin-top:0px;margin-bottom:0px;font-size: 20pt;font-weight: normal
	}
	h2 {
	color:INDUSTRIAL_STYLE_COLOR2;margin-top:0px;margin-bottom:0px;font-size:14pt;font-weight: bold
	}
	h3 {
	color:INDUSTRIAL_STYLE_COLOR1;margin-top:0px;margin-bottom:0px; font-size:11pt;font-weight: normal; text-transform: uppercase
	}
	h4 {
	color:INDUSTRIAL_STYLE_COLOR4;margin-top:0px;margin-

Table Style	Settings

	bottom:0px; font-size:10pt;font-weight: normal
	}
	h5 {
	color:INDUSTRIAL_STYLE_COLOR4;margin-top:0px;margin-bottom:0px;font-size:10pt;font-weight: bold
	}
	h6 {
	color:INDUSTRIAL_STYLE_COLOR4;margin-top:0px;margin-bottom:0px; font-size:10pt;font-weight: normal
	}
	Body {
	color: black;
	}
	</style>

TABLE_STYLE_PLAYGROUND	<style tablestyle="Playground"> TR.xdTitleRow {
	min-height: 61px;
	}
	TD.xdTitleCell {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 6px; PADDING-TOP: 18px; BORDER-LEFT: PLAYGROUND_STYLE_COLOR9 1.5pt solid; BORDER-RIGHT: PLAYGROUND_STYLE_COLOR9 1.5pt solid; BORDER-TOP: PLAYGROUND_STYLE_COLOR9 1.5pt solid; background-color: PLAYGROUND_STYLE_COLOR6; valign: bottom; text-align: right
	}
	TR.xdTitleRowWithHeading {
	min-height: 61px;} TD.xdTitleCellWithHeading{PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 2px; PADDING-TOP: 22px; BORDER-LEFT: PLAYGROUND_STYLE_COLOR9 1.5pt solid; BORDER-RIGHT: PLAYGROUND_STYLE_COLOR9 1.5pt solid; BORDER-TOP: PLAYGROUND_STYLE_COLOR9 1.5pt solid; background-color: PLAYGROUND_STYLE_COLOR6; valign: bottom; text-align: right
	}
	TR.xdTitleRowWithSubHeading {
	min-height: 61px;
	}
	TD.xdTitleCellWithSubHeading {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 6px; PADDING-TOP: 22px; BORDER-LEFT: PLAYGROUND_STYLE_COLOR9 1.5pt solid; BORDER-RIGHT: PLAYGROUND_STYLE_COLOR9 1.5pt solid; BORDER-TOP: PLAYGROUND_STYLE_COLOR9 1.5pt solid; background-color: PLAYGROUND_STYLE_COLOR6; valign: bottom; text-align: right
	}
	TR.xdTitleRowWithOffsetBody {
	min-height: 61px;
	}
	TD.xdTitleCellWithOffsetBody {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 6px; PADDING-TOP: 18px; BORDER-LEFT: PLAYGROUND_STYLE_COLOR9 1.5pt solid; BORDER-RIGHT: PLAYGROUND_STYLE_COLOR9 1.5pt solid; BORDER-TOP: PLAYGROUND_STYLE_COLOR9 1.5pt solid; background-color:

Table Style	Settings

	PLAYGROUND_STYLE_COLOR6; valign: bottom; text-align: left
	}
	TR.xdTitleHeadingRow {
	min-height: 45px;
	}
	TD.xdTitleHeadingCell {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 18px; PADDING-TOP: 0px; BORDER-LEFT: PLAYGROUND_STYLE_COLOR9 1.5pt solid; BORDER-RIGHT: PLAYGROUND_STYLE_COLOR9 1.5pt solid; background-color: PLAYGROUND_STYLE_COLOR6; valign: top; text-align: right
	}
	TR.xdTitleSubheadingRow {
	min-height: 76px;
	}
	TD.xdTitleSubheadingCell {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 14px; PADDING-TOP: 6px; BORDER-LEFT: PLAYGROUND_STYLE_COLOR9 1.5pt solid; BORDER-RIGHT: PLAYGROUND_STYLE_COLOR9 1.5pt solid; background-color: PLAYGROUND_STYLE_COLOR6; valign: top
	}
	TD.xdVerticalFill {
	BORDER-LEFT: PLAYGROUND_STYLE_COLOR9 1.5pt solid; BORDER-BOTTOM: PLAYGROUND_STYLE_COLOR9 1.5pt solid; BORDER-TOP: PLAYGROUND_STYLE_COLOR9 1.5pt solid; border-right-color: PLAYGROUND_STYLE_COLOR1; border-right-width: 1.5pt; border-right-style: solid; background-color: PLAYGROUND_STYLE_COLOR5
	}
	TD.xdTableContentCellWithVerticalOffset {
	BORDER-LEFT: PLAYGROUND_STYLE_COLOR9 1.5pt solid; BORDER-RIGHT: PLAYGROUND_STYLE_COLOR9 1.5pt solid; BORDER-BOTTOM: PLAYGROUND_STYLE_COLOR9 1.5pt solid; background-color: PLAYGROUND_STYLE_COLOR6; PADDING-RIGHT: 0px; PADDING-LEFT: 95px; PADDING-BOTTOM: 6px; PADDING-TOP: 18px; valign: bottom; text-align: left
	}
	TR.xdTableContentRow {
	min-height: 140px;
	}
	TD.xdTableContentCell {
	padding: 0px 0px 0px 0px ; BORDER-LEFT: PLAYGROUND_STYLE_COLOR9 1.5pt solid; BORDER-RIGHT: PLAYGROUND_STYLE_COLOR9 1.5pt solid; BORDER-BOTTOM: PLAYGROUND_STYLE_COLOR9 1.5pt solid; background-color: PLAYGROUND_STYLE_COLOR6; valign: top
	}
	TD.xdTableContentCellWithVerticalFill {
	padding: 0px 0px 0px 0px ; BORDER-RIGHT: PLAYGROUND_STYLE_COLOR9 1.5pt solid; BORDER-BOTTOM: PLAYGROUND_STYLE_COLOR9 1.5pt solid; background-color: PLAYGROUND_STYLE_COLOR6; valign: top
	}
	TD.xdTableStyleOneCol {
	padding: 4px 22px 4px 22px;
	}

Table Style	Settings

	TR.xdContentRowOneCol {
	min-height:45px; valign:center
	}
	TR.xdHeadingRow {
	min-height: 37px;
	}
	TD.xdHeadingCell {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 4px; PADDING-TOP: 12px; border-top-color: PLAYGROUND_STYLE_COLOR3; border-top-width: 1.5pt; border-top-style: dashed; border-bottom-color: PLAYGROUND_STYLE_COLOR5; border-bottom-width: 1pt; border-bottom-style: solid; background-color: PLAYGROUND_STYLE_COLOR7; valign: bottom
	}
	TR.xdSubheadingRow {
	min-height: 29 px;
	}
	TD.xdSubheadingCell {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 4px; PADDING-TOP: 4px; border-bottom-color: PLAYGROUND_STYLE_COLOR8; border-bottom-width: 2.25pt; border-bottom-style: solid; background-color: PLAYGROUND_STYLE_COLOR5; valign: bottom
	}
	TR.xdHeadingRowEmphasis {
	min-height: 37px;
	}
	TD.xdHeadingCellEmphasis {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 4px; PADDING-TOP: 12px; border-top-color: PLAYGROUND_STYLE_COLOR3; border-top-width: 1.5pt; border-top-style: dashed; border-bottom-color: PLAYGROUND_STYLE_COLOR5; border-bottom-width: 1pt; border-bottom-style: solid; background-color: PLAYGROUND_STYLE_COLOR7; valign: bottom
	}
	TR.xdSubheadingRowEmphasis {
	min-height: 29 px;
	}
	TD.xdSubheadingCellEmphasis {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 4px; PADDING-TOP: 4px; border-bottom-color: PLAYGROUND_STYLE_COLOR8; border-bottom-width: 2.25pt; border-bottom-style: solid; background-color: PLAYGROUND_STYLE_COLOR5; valign: bottom
	}
	TR.xdTableLabelControlStackedRow {
	min-height: 45px;
	}
	TD.xdTableLabelControlStackedCellLabel {
	padding: 4px 5px 4px 22px;
	}
	TD.xdTableLabelControlStackedCellComponent {
	padding: 4px 22px 4px 5px;
	}
	TR.xdTableRow {

Table Style	Settings

	min-height: 30px;
	}
	TD.xdTableCellLabel {
	padding-top: 4px; padding-bottom: 4px; padding-right: 5px; padding-left: 22px;
	}
	TD.xdTableCellComponent {
	padding-top: 4px; padding-bottom: 4px; padding-right: 22px; padding-left: 5px;
	}
	TD.xdTableMiddleCell {
	padding-top: 4px; padding-bottom: 4px; padding-right: 5px; padding-left: 5px;
	}
	TR.xdTableEmphasisRow {
	min-height: 30px;
	}
	TD.xdTableEmphasisCellLabel {
	padding-top: 4px; padding-bottom: 4px; padding-right: 5px; padding-left: 22px; background-color: PLAYGROUND_STYLE_COLOR4; BORDER-TOP: PLAYGROUND_STYLE_COLOR7 1pt solid; BORDER-BOTTOM: PLAYGROUND_STYLE_COLOR7 1pt solid;
	}
	TD.xdTableEmphasisCellComponent {
	padding-top: 4px; padding-bottom: 4px; padding-right: 22px; padding-left: 5px; background-color: PLAYGROUND_STYLE_COLOR4; BORDER-TOP: PLAYGROUND_STYLE_COLOR7 1pt solid; BORDER-BOTTOM: PLAYGROUND_STYLE_COLOR7 1pt solid;
	}
	TD.xdTableMiddleCellEmphasis {
	padding-top: 4px; padding-bottom: 4px; padding-right: 5px; padding-left: 5px; background-color: PLAYGROUND_STYLE_COLOR4; BORDER-TOP: PLAYGROUND_STYLE_COLOR7 1pt solid; BORDER-BOTTOM: PLAYGROUND_STYLE_COLOR7 1pt solid;
	}
	TR.xdTableOffsetRow {
	min-height: 30px;
	}
	TD.xdTableOffsetCellLabel {
	BORDER-TOP: PLAYGROUND_STYLE_COLOR7 1pt solid; BORDER-BOTTOM: PLAYGROUND_STYLE_COLOR7 1pt solid; PADDING-RIGHT: 5px; PADDING-LEFT: 22px; PADDING-BOTTOM: 4px; PADDING-TOP: 4px; background-color: PLAYGROUND_STYLE_COLOR4;
	}
	TD.xdTableOffsetCellComponent {
	BORDER-TOP: PLAYGROUND_STYLE_COLOR7 1pt solid; BORDER-BOTTOM: PLAYGROUND_STYLE_COLOR7 1pt solid; background-color: PLAYGROUND_STYLE_COLOR4; PADDING-RIGHT: 22px; PADDING-LEFT: 5px; PADDING-BOTTOM: 4px; PADDING-TOP: 4px;
	}
	P {
	margin-top:0px; color:PLAYGROUND_STYLE_COLOR1; font-size:11pt

Table Style	Settings

	}
	h1 {
	color:PLAYGROUND_STYLE_COLOR1;margin-top:0px;margin-bottom:0px;font-size: 22pt;font-weight: bold
	}
	h2 {
	color:PLAYGROUND_STYLE_COLOR2;margin-top:0px;margin-bottom:0px;font-size:16pt;font-weight: normal
	}
	h3 {
	color:PLAYGROUND_STYLE_COLOR7;margin-top:0px;margin-bottom:0px; font-size:12pt;font-weight: bold; text-transform: uppercase
	}
	h4 {
	color:PLAYGROUND_STYLE_COLOR1;margin-top:0px;margin-bottom:0px; font-size:11pt;font-weight: bold
	}
	h5 {
	color:PLAYGROUND_STYLE_COLOR1;margin-top:0px;margin-bottom:0px;font-size:11pt; font-weight: normal
	}
	h6 {
	color:PLAYGROUND_STYLE_COLOR1;margin-top:0px;margin-bottom:0px; font-size:11pt;font-weight: normal
	}
	Body {
	color:black;
	}
	</style>

TABLE_STYLE_MODERN	<style tablestyle="Modern"> TR.xdTitleRow {
	min-height: 62px;
	}
	TD.xdTitleCell {
	PADDING-RIGHT: 0px; PADDING-LEFT: 0px; PADDING-BOTTOM: 6px; PADDING-TOP: 18px; background-color: MODERN_STYLE_COLOR3; valign:bottom
	}
	TR.xdTitleRowWithHeading {
	min-height: 61px;
	}
	TD.xdTitleCellWithHeading {
	PADDING-RIGHT: 0px; PADDING-LEFT: 0px; PADDING-BOTTOM: 6px; PADDING-TOP: 18px; background-color: MODERN_STYLE_COLOR3; valign:bottom
	}
	TR.xdTitleRowWithSubHeading {
	min-height: 59px;
	}
	TD.xdTitleCellWithSubHeading {
	PADDING-RIGHT: 0px; PADDING-LEFT: 0px; PADDING-BOTTOM: 4px; PADDING-TOP: 18px; background-color: MODERN_STYLE_COLOR3; valign:bottom
	}
	TR.xdTitleRowWithOffsetBody {

Table Style	Settings

	min-height: 62px;
	}
	TD.xdTitleCellWithOffsetBody {
	PADDING-RIGHT: 0px; PADDING-LEFT: 0px; PADDING-BOTTOM: 6px; PADDING-TOP: 18px; background-color: MODERN_STYLE_COLOR3; valign:bottom
	}
	TR.xdTitleHeadingRow {
	min-height: 37px;
	}
	TD.xdTitleHeadingCell {
	PADDING-RIGHT: 0px; PADDING-LEFT: 0px; PADDING-BOTTOM: 12px; PADDING-TOP: 0px; background-color: MODERN_STYLE_COLOR3; valign: top
	}
	TR.xdTitleSubheadingRow {
	min-height: 67px;
	}
	TD.xdTitleSubheadingCell {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 12px; PADDING-TOP: 0px; background-color: MODERN_STYLE_COLOR3; valign: top
	}
	TD.xdVerticalFill {
	background-color: MODERN_STYLE_COLOR3
	}
	TD.xdTableContentCellWithVerticalOffset {
	background-color: MODERN_STYLE_COLOR3; PADDING-RIGHT: 10px; PADDING-LEFT: 85px; PADDING-BOTTOM: 0px; PADDING-TOP: 12px;
	}
	TR.xdTableContentRow {
	min-height: 140px;
	}
	TD.xdTableContentCell {
	padding: 0px 0px 0px 0px; background-color: MODERN_STYLE_COLOR3
	}
	TD.xdTableContentCellWithVerticalFill {
	padding: 0px 0px 0px 0px; background-color: MODERN_STYLE_COLOR3
	}
	TD.xdTableStyleOneCol {
	padding: 6px 22px 6px 22px;
	}
	TR.xdContentRowOneCol {
	min-height:45px; valign:center
	}
	TR.xdHeadingRow {
	min-height: 38px;
	}
	TD.xdHeadingCell {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 6px; PADDING-TOP: 6px; border-bottom-color: MODERN_STYLE_COLOR4; border-bottom-width: 1.5pt; border-bottom-style: solid; valign: top
	}

Table Style	Settings

	TR.xdSubheadingRow {
	min-height: 30px;
	}
	TD.xdSubheadingCell {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 6px; PADDING-TOP: 6px; border-bottom-color: MODERN_STYLE_COLOR5; border-bottom-width: 1pt; border-bottom-style: solid;
	}
	TR.xdHeadingRowEmphasis {
	min-height: 38px;
	}
	TD.xdHeadingCellEmphasis {
	PADDING-RIGHT: 22px; PADDING-LEFT: 16px; PADDING-BOTTOM: 6px; PADDING-TOP: 6px; border-bottom-color: MODERN_STYLE_COLOR4; border-bottom-width: 1.5pt; border-bottom-style: solid; border-left-color: MODERN_STYLE_COLOR4; border-left-width: 6pt; border-left-style: solid; valign: top
	}
	TR.xdSubheadingRowEmphasis {
	min-height: 30px;
	}
	TD.xdSubheadingCellEmphasis {
	PADDING-RIGHT: 22px; PADDING-LEFT: 16px; PADDING-BOTTOM: 6px; PADDING-TOP: 6px; border-bottom-color: MODERN_STYLE_COLOR5; border-bottom-width: 1pt; border-bottom-style: solid; border-left-color: MODERN_STYLE_COLOR4; border-left-width: 6pt; border-left-style: solid;
	}
	TR.xdTableLabelControlStackedRow {
	min-height: 45px;
	}
	TD.xdTableLabelControlStackedCellLabel {
	padding: 4px 5px 4px 22px;
	}
	TD.xdTableLabelControlStackedCellComponent {
	padding: 4px 22px 4px 5px;
	}
	TR.xdTableRow {
	min-height: 30px;
	}
	TD.xdTableCellLabel {
	padding-top: 4px; padding-bottom: 4px; padding-right: 5px; padding-left: 22px;
	}
	TD.xdTableCellComponent {
	padding-top: 4px; padding-bottom: 4px; padding-right: 22px; padding-left: 5px;
	}
	TD.xdTableMiddleCell {
	padding-top: 4px; padding-bottom: 4px; padding-right: 5px; padding-left: 5px;
	}
	TR.xdTableEmphasisRow {
	min-height: 30px;

Table Style	Settings

	}
	TD.xdTableEmphasisCellLabel {
	padding-top: 4px; padding-bottom: 4px; padding-right: 5px; padding-left: 16px; border-left-color: MODERN_STYLE_COLOR4; border-left-width: 6pt; border-left-style: solid;
	}
	TD.xdTableEmphasisCellComponent {
	padding-top: 4px; padding-bottom: 4px; padding-right: 22px; padding-left: 5px;
	}
	TD.xdTableMiddleCellEmphasis {
	padding-top: 4px; padding-bottom: 4px; padding-right: 5px; padding-left: 5px;
	}
	TR.xdTableOffsetRow {
	min-height: 30px;
	}
	TD.xdTableOffsetCellLabel {
	border-left-color: MODERN_STYLE_COLOR4; border-left-width: 6pt; border-left-style: solid; PADDING-RIGHT: 5px; PADDING-LEFT: 16px; PADDING-BOTTOM: 4px; PADDING-TOP: 4px;
	}
	TD.xdTableOffsetCellComponent {
	PADDING-RIGHT: 22px; PADDING-LEFT: 5px; PADDING-BOTTOM: 4px; PADDING-TOP: 4px;
	}
	P {
	margin-top:0px; color:MODERN_STYLE_COLOR2; font-size:11pt
	}
	h1 {
	color:MODERN_STYLE_COLOR1;margin-top:0px;margin-bottom:0px;font-size: 22pt;font-weight: normal
	}
	h2 {
	color:MODERN_STYLE_COLOR2;margin-top:0px;margin-bottom:0px;font-size:15pt;font-weight: normal
	}
	h3 {
	color:MODERN_STYLE_COLOR1;margin-top:0px;margin-bottom:0px; font-size:12pt;font-weight: bold
	}
	h4 {
	color:MODERN_STYLE_COLOR2;margin-top:0px;margin-bottom:0px; font-size:11pt;font-weight: normal
	}
	h5 {
	color:MODERN_STYLE_COLOR2;margin-top:0px;margin-bottom:0px;font-size:11pt; font-weight: bold
	}
	h6 {
	color:MODERN_STYLE_COLOR2;margin-top:0px;margin-bottom:0px; font-size:10pt;font-weight: normal
	}
	Body {

Table Style	Settings
	color: black ; } </style>

TABLE_STYLE_SHAREPOINT	<style tablestyle="Sharepoint"> TR.xdTitleRow {
	min-height: 58px; }
	TD.xdTitleCell {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM:
	6px; PADDING-TOP: 18px; BORDER-LEFT:
	SHAREPOINT_STYLE_COLOR3 1pt solid; BORDER-RIGHT:
	SHAREPOINT_STYLE_COLOR3 1pt solid; BORDER-TOP:
	SHAREPOINT_STYLE_COLOR3 1pt solid; background-color:
	SHAREPOINT_STYLE_COLOR9
	}
	TR.xdTitleRowWithHeading {
	min-height: 58px;
	}
	TD.xdTitleCellWithHeading {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM:
	4px; PADDING-TOP: 18px; BORDER-LEFT:
	SHAREPOINT_STYLE_COLOR3 1pt solid; BORDER-RIGHT:
	SHAREPOINT_STYLE_COLOR3 1pt solid; BORDER-TOP:
	SHAREPOINT_STYLE_COLOR3 1pt solid; background-color:
	SHAREPOINT_STYLE_COLOR9
	}
	TR.xdTitleRowWithSubHeading {
	min-height: 58px;
	}
	TD.xdTitleCellWithSubHeading {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM:
	4px; PADDING-TOP: 18px; BORDER-LEFT:
	SHAREPOINT_STYLE_COLOR3 1pt solid; BORDER-RIGHT:
	SHAREPOINT_STYLE_COLOR3 1pt solid; BORDER-TOP:
	SHAREPOINT_STYLE_COLOR3 1pt solid; background-color:
	SHAREPOINT_STYLE_COLOR9
	}
	TR.xdTitleRowWithOffsetBody {
	min-height: 58px;
	}
	TD.xdTitleCellWithOffsetBody {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM:
	6px; PADDING-TOP: 18px; BORDER-LEFT:
	SHAREPOINT_STYLE_COLOR3 1pt solid; BORDER-RIGHT:
	SHAREPOINT_STYLE_COLOR3 1pt solid; BORDER-TOP:
	SHAREPOINT_STYLE_COLOR3 1pt solid; background-color:
	SHAREPOINT_STYLE_COLOR9
	}
	TR.xdTitleHeadingRow {
	min-height: 38px;
	}
	TD.xdTitleHeadingCell {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM:
	12px; PADDING-TOP: 0px; BORDER-LEFT:
	SHAREPOINT_STYLE_COLOR3 1pt solid; BORDER-RIGHT:
	SHAREPOINT_STYLE_COLOR3 1pt solid; background-color:
	SHAREPOINT_STYLE_COLOR9; valign: top
	}

Table Style	Settings

	TR.xdTitleSubheadingRow {
	min-height: 67px;} TD.xdTitleSubheadingCell{PADDING-
	RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 18px;
	PADDING-TOP: 8px; BORDER-LEFT: SHAREPOINT_STYLE_COLOR3
	1pt solid; BORDER-RIGHT: SHAREPOINT_STYLE_COLOR3 1pt
	solid; border-top-color: SHAREPOINT_STYLE_COLOR5;
	border-top-width: 1pt; border-top-style: solid;
	background-color: SHAREPOINT_STYLE_COLOR9
	}
	TD.xdVerticalFill {
	BORDER-LEFT: SHAREPOINT_STYLE_COLOR3 1pt solid; BORDER-
	BOTTOM: SHAREPOINT_STYLE_COLOR3 1pt solid; BORDER-TOP:
	SHAREPOINT_STYLE_COLOR3 1pt solid; background-color:
	SHAREPOINT_STYLE_COLOR6
	}
	TD.xdTableContentCellWithVerticalOffset {
	BORDER-LEFT: SHAREPOINT_STYLE_COLOR3 1pt solid; BORDER-
	RIGHT: SHAREPOINT_STYLE_COLOR3 1pt solid; BORDER-BOTTOM:
	SHAREPOINT_STYLE_COLOR3 1pt solid; background-color:
	SHAREPOINT_STYLE_COLOR9; PADDING-RIGHT: 10px; PADDING-
	LEFT: 85px; PADDING-BOTTOM: 0px; PADDING-TOP: 12px;
	}
	TR.xdTableContentRow {
	min-height: 140px;
	}
	TD.xdTableContentCell {
	padding: 0px 0px 0px 0px; BORDER-LEFT:
	SHAREPOINT_STYLE_COLOR3 1pt solid; BORDER-RIGHT:
	SHAREPOINT_STYLE_COLOR3 1pt solid; BORDER-BOTTOM:
	SHAREPOINT_STYLE_COLOR3 1pt solid; background-color:
	SHAREPOINT_STYLE_COLOR9
	}
	TD.xdTableContentCellWithVerticalFill {
	padding: 0px 0px 0px 0px; BORDER-LEFT:
	SHAREPOINT_STYLE_COLOR3 1pt solid; BORDER-RIGHT:
	SHAREPOINT_STYLE_COLOR3 1pt solid; BORDER-BOTTOM:
	SHAREPOINT_STYLE_COLOR3 1pt solid; background-color:
	SHAREPOINT_STYLE_COLOR9
	}
	TD.xdTableStyleOneCol {
	padding: 4px 22px 4px 22px;BORDER-TOP:
	SHAREPOINT_STYLE_COLOR3 1pt solid; BORDER-BOTTOM:
	SHAREPOINT_STYLE_COLOR3 1pt solid;
	}
	TR.xdContentRowOneCol {
	min-height:45px; valign:center
	}
	TR.xdHeadingRow {
	min-height: 36px;
	}
	TD.xdHeadingCell {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM:
	4px; PADDING-TOP: 6px; border-bottom-
	color:SHAREPOINT_STYLE_COLOR6; border-bottom-width:
	1.5pt; border-bottom-style: solid;
	}
	TR.xdSubheadingRow {
	min-height: 27px;

Table Style	Settings

	}
	TD.xdSubheadingCell {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 4px; PADDING-TOP: 4px; border-bottom-color: SHAREPOINT_STYLE_COLOR4; border-bottom-width: 1pt; border-bottom-style: solid;
	}
	TR.xdHeadingRowEmphasis {
	min-height: 36px;
	}
	TD.xdHeadingCellEmphasis {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 4px; PADDING-TOP: 6px; Border-bottom-color:SHAREPOINT_STYLE_COLOR6; border-bottom-width: 1.5pt; border-bottom-style: solid;
	}
	TR.xdSubheadingRowEmphasis {
	min-height: 27px;
	}
	TD.xdSubheadingCellEmphasis {
	PADDING-RIGHT: 22px; PADDING-LEFT: 22px; PADDING-BOTTOM: 4px; PADDING-TOP: 4px; border-bottom-color: SHAREPOINT_STYLE_COLOR4; border-bottom-width: 1pt; border-bottom-style: solid;
	}
	TR.xdTableLabelControlStackedRow {
	min-height: 45px;
	}
	TD.xdTableLabelControlStackedCellLabel {
	padding: 4px 5px 4px 22px;BORDER-TOP: SHAREPOINT_STYLE_COLOR3 1pt solid; BORDER-BOTTOM: SHAREPOINT_STYLE_COLOR3 1pt solid;}
	TD.xdTableLabelControlStackedCellComponent {
	padding: 4px 22px 4px 5px;BORDER-TOP: SHAREPOINT_STYLE_COLOR3 1pt solid; BORDER-BOTTOM: SHAREPOINT_STYLE_COLOR3 1pt solid;}
	TR.xdTableRow {
	min-height: 30px;
	}
	TD.xdTableCellLabel {
	padding-top: 4px; padding-bottom: 4px; padding-right: 5px; padding-left: 22px;
	}
	TD.xdTableCellComponent {
	padding-top: 4px; padding-bottom: 4px; padding-right: 22px; padding-left: 5px;
	}
	TD.xdTableMiddleCell {
	padding-top: 4px; padding-bottom: 4px; padding-right: 5px; padding-left: 5px;
	}
	TR.xdTableEmphasisRow {
	min-height: 30px;
	}
	TD.xdTableEmphasisCellLabel {
	padding-top: 4px; padding-bottom: 4px; padding-right: 5px; padding-left: 22px; background-color:

Table Style	Settings

	SHAREPOINT_STYLE_COLOR7; BORDER-TOP: SHAREPOINT_STYLE_COLOR3 1pt solid; BORDER-BOTTOM: SHAREPOINT_STYLE_COLOR3 1pt solid;
	}
	TD.xdTableEmphasisCellComponent {
	padding-top: 4px; padding-bottom: 4px; padding-right: 22px; padding-left: 5px; background-color: SHAREPOINT_STYLE_COLOR7; BORDER-TOP: SHAREPOINT_STYLE_COLOR3 1pt solid; BORDER-BOTTOM: SHAREPOINT_STYLE_COLOR3 1pt solid;
	}
	TD.xdTableMiddleCellEmphasis {
	padding-top: 4px; padding-bottom: 4px; padding-right: 5px; padding-left: 5px; background-color: SHAREPOINT_STYLE_COLOR7; BORDER-TOP: SHAREPOINT_STYLE_COLOR3 1pt solid; BORDER-BOTTOM: SHAREPOINT_STYLE_COLOR3 1pt solid;
	}
	TR.xdTableOffsetRow {
	min-height: 30px;
	}
	TD.xdTableOffsetCellLabel {
	BORDER-TOP: SHAREPOINT_STYLE_COLOR3 1pt solid; BORDER-BOTTOM: SHAREPOINT_STYLE_COLOR3 1pt solid; PADDING-RIGHT: 5px; PADDING-LEFT: 22px; PADDING-BOTTOM: 4px; PADDING-TOP: 4px;
	}
	TD.xdTableOffsetCellComponent {
	BORDER-TOP: SHAREPOINT_STYLE_COLOR3 1pt solid; BORDER-BOTTOM: SHAREPOINT_STYLE_COLOR3 1pt solid; background-color: SHAREPOINT_STYLE_COLOR7; PADDING-RIGHT: 22px; PADDING-LEFT: 5px; PADDING-BOTTOM: 4px; PADDING-TOP: 4px;
	}
	P {
	margin-top:0px; color:SHAREPOINT_STYLE_COLOR1; font-size:10pt
	}
	h1 {
	color:SHAREPOINT_STYLE_COLOR1;margin-top:0px;margin-bottom:0px;font-size: 22pt;font-weight: normal
	}
	h2 {
	color:SHAREPOINT_STYLE_COLOR2;margin-top:0px;margin-bottom:0px;font-size:15pt;font-weight: normal
	}
	h3 {
	color:SHAREPOINT_STYLE_COLOR1;margin-top:0px;margin-bottom:0px; font-size:12pt;font-weight: bold
	}
	h4 {
	color:SHAREPOINT_STYLE_COLOR1;margin-top:0px;margin-bottom:0px; font-size:10pt;font-weight: normal
	}
	h5 {
	color:SHAREPOINT_STYLE_COLOR1;margin-top:0px;margin-bottom:0px;font-size:10pt; font-weight: bold
	}

Table Style	Settings

	h6 {
	color:SHAREPOINT_STYLE_COLOR1;margin-top:0px;margin-bottom:0px; font-size:10pt;font-weight: normal
	}
	Body {
	color: black;
	}
	</style>

PROFESSIONAL_STYLE_COLOR1, **PROFESSIONAL_STYLE_COLOR2**, **PROFESSIONAL_STYLE_COLOR3**, **PROFESSIONAL_STYLE_COLOR4**, **PROFESSIONAL_STYLE_COLOR5**, **PROFESSIONAL_STYLE_COLOR6**, **PROFESSIONAL_STYLE_COLOR7**, **INDUSTRIAL_STYLE_COLOR1**, **INDUSTRIAL_STYLE_COLOR2**, **INDUSTRIAL_STYLE_COLOR3**, **INDUSTRIAL_STYLE_COLOR4**, **INDUSTRIAL_STYLE_COLOR5**, **INDUSTRIAL_STYLE_COLOR6**, **PLAYGROUND_STYLE_COLOR1**, **PLAYGROUND_STYLE_COLOR2**, **PLAYGROUND_STYLE_COLOR3**, **PLAYGROUND_STYLE_COLOR4**, **PLAYGROUND_STYLE_COLOR5**, **PLAYGROUND_STYLE_COLOR6**, **PLAYGROUND_STYLE_COLOR7**, **PLAYGROUND_STYLE_COLOR8**, **PLAYGROUND_STYLE_COLOR9**, **MODERN_STYLE_COLOR1**, **MODERN_STYLE_COLOR2**, **MODERN_STYLE_COLOR3**, **MODERN_STYLE_COLOR4**, **MODERN_STYLE_COLOR5**, **SHAREPOINT_STYLE_COLOR1**, **SHAREPOINT_STYLE_COLOR2**, **SHAREPOINT_STYLE_COLOR3**, **SHAREPOINT_STYLE_COLOR4**, **SHAREPOINT_STYLE_COLOR5**, **SHAREPOINT_STYLE_COLOR6**, **SHAREPOINT_STYLE_COLOR7**, **SHAREPOINT_STYLE_COLOR8**, **SHAREPOINT_STYLE_COLOR9**: These are different color values that change depending on the **THEME_STYLE** being used and MUST be represented in units as specified in [CSS-LEVEL1] section 6.3.

2.4.1.4 Control Data Formatting

This section specifies the rules that MUST be used for formatting data in controls.

DATA_FMT_LOCALE_VAL: MUST be a language code identifier (LCID) value, as specified in [MS-LCID].

DATA_FMT_LOCALE: locale:DATA_FMT_LOCALE_VAL.

DATA_FMT_NUM_DIGITS: [0-9] or auto. See numDigits in the **datafmt** control attribute, as specified in section 2.4.2.11.

DATA_FMT_GROUPING: "-1" or [0-9] or "32" See **grouping** in the **datafmt** control attribute, as specified in section 2.4.2.11.

DATA_FMT_DECIMAL_SEP: Period (".") or comma (",") or **space_char**. See **decimalSep** in the **datafmt** control attribute, as specified in section 2.4.2.11.

DATA_FMT_THOUSAND_SEP: Period (".") or comma (",") or **space_char**. See **thousandSep** in the **datafmt** control attribute, as specified in section 2.4.2.11.

DATA_FMT_NEG_ORDER: See **negativeOrder** in the **datafmt** control attribute, as specified in section 2.4.2.11.

DATA_FMT_POS_ORDER: See **positiveOrder** in the **datafmt** control attribute, as specified in section 2.4.2.11.

DATA_FMT_CUR_LOCALE: currencyLocale:DATA_FMT_LOCALE_VAL.

DATA_FMT_DATE_FORMAT_CUSTOM: See **dateFormat** in the **datafmt** control attribute, as specified in section 2.4.2.11.

DATA_FMT_DATE_FORMAT: MUST be one of the following values:

- "Short Date"
- "Long Date"
- "Year Month"
- "none"
- **"DATA_FMT_DATE_FORMAT_CUSTOM"**

DATA_FMT_ALT_CAL: MUST be zero ("0") or "1". See **useAltCalendar** in the **datafmt** control attribute, as specified in section 2.4.2.11.

DATA_FMT_EN_STR: MUST be zero ("0") or "1". See **englishStringOnly** in the **datafmt** control attribut**e**, as specified in section 2.4.2.11.

DATA_FMT_TIME_FORMAT_CUSTOM: See **timeFormat** in the **datafmt** control attribute, as specified in section 2.4.2.11.

DATA_FMT_TIME_FORMAT: MUST be one of the following values:

- "Short Time"
- "Long Time"
- "none"
- "**DATA_FMT_TIME_FORMAT_CUSTOM**"

DATA_FMT_NOSECONDS: MUST be zero ("0") or "1". See **noSeconds** in the **datafmt** control attribute, as specified in section 2.4.2.11.

DATA_FMT_CAT_STRING: "string", "plainMutiline"

DATA_FMT_CAT_PERCENTAGE: "percentage","**semicolon-delimited list of** (**DATA_FMT_LOCALE**?, **DATA_FMT_NUM_DIGITS**, **DATA_FMT_GROUPING**?, **DATA_FMT_DECIMAL_SEP**?, **DATA_FMT_THOUSAND_SEP**?, **DATA_FMT_NEG_ORDER**)"

DATA_FMT_CAT_NUMBER: "number","**semicolon-delimited list of** (**DATA_FMT_LOCALE**?, **DATA_FMT_NUM_DIGITS**, **DATA_FMT_GROUPING**?, **DATA_FMT_DECIMAL_SEP**?, **DATA_FMT_THOUSAND_SEP**?, **DATA_FMT_NEG_ORDER**)"

DATA_FMT_CAT_DATETIME: "datetime","**semicolon delimited list of** (**DATA_FMT_LOCALE**?, **DATA_FMT_DATE_FORMAT**, **DATA_FMT_ALT_CAL**?, **DATA_FMT_EN_STR**?, **DATA_FMT_TIME_FORMAT**, **DATA_FMT_NOSECONDS**?)"

DATA_FMT_CAT_DATE: "date","**semicolon delimited list of** (**DATA_FMT_LOCALE**?, **DATA_FMT_DATE_FORMAT**, **DATA_FMT_ALT_CAL**?, **DATA_FMT_EN_STR**?)"

DATA_FMT_CAT_TIME: "time","**semicolon delimited list of** (**DATA_FMT_LOCALE**?, **DATA_FMT_TIME_FORMAT**, **DATA_FMT_NOSECONDS**?)"

DATA_FMT_CTRL_DATE_PICKER: "**DATA_FMT_CAT_DATE"** or "**DATA_FMT_CAT_DATETIME**".

DATA_FMT_CTRL_EXPBOX: MUST be one of the following values:

- "**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**

DATA_FMT_CTRL_MSLB: MUST be one of the following values:

- "**DATA_FMT_CAT_DATE**"
- "**DATA_FMT_CAT_DATETIME**"
- "**DATA_FMT_CAT_NUMBER**"
- "**DATA_FMT_CAT_TIME**"

DATA_FMT2: See the **datafmt2** control attribute, as specified in section 2.4.2.37.4.

2.4.1.5 Button Control

The button control is an unbound control that will run actions (submit, query, new, and refresh), rules (1), or custom code when clicked. The following table describes the symbols for a button control.

Symbol	Description
BUTTON_RULES_AND_CUSTOM_CODE	The button runs rules (1) and custom code when clicked.
BUTTON_RULES_AND_CUSTOM_CODE_WITH_CONDITIONAL_FORMATTING	The button runs rules (1) and custom code when

Symbol	Description
	clicked, and supports conditional formatting.
BUTTON_RULES_AND_CUSTOM_CODE_WITH_DYNAMIC_DISPLAY_NAME_AND_CONDITIONAL_FORMATTING	The button runs 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.
BUTTON_ACTION	The button runs actions (submit, query, new, and refresh)

Symbol	Description
	when clicked.
BUTTON_ACTION_WITH_CONDITIONAL_FORMATTING	The button runs actions (submit, query, new, and refresh) when clicked, and supports conditional formatting.
BUTTON_ACTION_WITH_DYNAMIC_DISPLAY_NAME_AND_CONDITIONAL_FORMATTING	The button runs actions (submit, query, new, and refresh) when clicked, renders a dynamic display name, and supports conditional formatting.

BUTTON_ACTION_TYPE: MUST be one of the following values:

- "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**?, **STYLE_CAPTION**)

BUTTON_CONDITIONAL_FORMATTING:

```
(<xsl:attribute name="style">BUTTON_STYLE?<xsl:choose>
        (<xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_DISPLAY_NONE;
STYLE_CAPTION</xsl:when>|
        <xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_CAPTION</xsl:when>|
        <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;
STYLE_CAPTION</xsl:when>|
        <xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_CAPTION</xsl:when>|
        <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;
STYLE_CAPTION</xsl:when>|
        <xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_CAPTION</xsl:when>|
        <xsl:when
test="BOOLEAN_XPATH_EXPRESSION">BUTTON_CONDITIONAL_FORMATTING_STYLE</xsl:when>)*
    </xsl:choose>
</xsl:attribute>
(<xsl:choose>
    <xsl:when test="not(xdEnvironment:IsBrowser())"/>
    (<xsl:when test="BOOLEAN_XPATH_EXPRESSION"/>|
    <xsl:when test="BOOLEAN_XPATH_EXPRESSION">
```

```
        <xsl:attribute name="disabled">true</xsl:attribute>
    </xsl:when>)+
</xsl:choose>)?
```

BUTTON_RULES_AND_CUSTOM_CODE:

```
<input class="langFont" title="ANY_STRING" type="button" (value="NON_EMPTY_STRING")?
xd:xctname="Button" xd:CtrlId="CONTROL_ID" (xd:auxDom="AUX_DOM_SOURCE_NAME")?
(tabIndex="TAB_INDEX")? (style="BUTTON_STYLE")? (xd:postbackModel="BUTTON_POSTBACKMODEL")?
(accessKey="SINGLE_CHARACTER")? (size="SIZE")?/>
```

BUTTON_RULES_AND_CUSTOM_CODE_WITH_CONDITIONAL_FORMATTING:

```
<input class="langFont" title="ANY_STRING" type="button" (value="NON_EMPTY_STRING")?
xd:xctname="Button" xd:CtrlId="CONTROL_ID" (xd:auxDom="AUX_DOM_SOURCE_NAME")?
(tabIndex="TAB_INDEX")? (style="BUTTON_STYLE")? (xd:postbackModel="BUTTON_POSTBACKMODEL")?
(accessKey="SINGLE_CHARACTER")? (size="SIZE")?>
    BUTTON_CONDITIONAL_FORMATTING
</input>
```

BUTTON_RULES_AND_CUSTOM_CODE_WITH_DYNAMIC_DISPLAY_NAME_AND_CONDITIONAL_FORMATTING:

```
<input class="langFont" title="ANY_STRING" type="button" xd:xctname="Button"
xd:CtrlId="CONTROL_ID" (xd:auxDom="AUX_DOM_SOURCE_NAME")? (tabIndex="TAB_INDEX")?
(style="BUTTON_STYLE")? (xd:postbackModel="BUTTON_POSTBACKMODEL")?
(accessKey="SINGLE_CHARACTER")? (size="SIZE")?>
    BUTTON_CONDITIONAL_FORMATTING
    CHECK_FOR_GETDOM_BEGIN1
    <xsl:attribute name="value">
        <xsl:value-of select="STRING_XPATH_EXPRESSION"/>
    </xsl:attribute>
    CHECK_FOR_GETDOM_END1
</input>
```

BUTTON_UPDATE_FORM_WITH_CONDITIONAL_FORMATTING:

```
<input class="langFont" title="ANY_STRING" type="button" value="NON_EMPTY_STRING"
xd:xctname="Button" xd:CtrlId="CONTROL_ID" xd:action="updateForm"
(xd:auxDom="AUX_DOM_SOURCE_NAME")? (tabIndex="TAB_INDEX")? (style="BUTTON_ACTION_STYLE")?
(xd:postbackModel="BUTTON_POSTBACKMODEL")? (accessKey="SINGLE_CHARACTER")? (size="SIZE")?>
    BUTTON_UPDATE_FORM_CONDITIONAL_FORMATTING
</input>
```

BUTTON_UPDATE_FORM_WITH_DYNAMIC_DISPLAY_NAME_AND_CONDITIONAL_FORMATTING:

```
<input class="langFont" title="ANY_STRING" type="button" xd:xctname="Button"
xd:CtrlId="CONTROL_ID" xd:action="updateForm" (xd:auxDom="AUX_DOM_SOURCE_NAME")?
(tabIndex="TAB_INDEX")? (style="BUTTON_ACTION_STYLE")?
(xd:postbackModel="BUTTON_POSTBACKMODEL")? (accessKey="SINGLE_CHARACTER")? (size="SIZE")?>
    BUTTON_UPDATE_FORM_CONDITIONAL_FORMATTING
    CHECK_FOR_GETDOM_BEGIN1
    <xsl:attribute name="value">
        <xsl:value-of select="STRING_XPATH_EXPRESSION"/>
    </xsl:attribute>
    CHECK_FOR_GETDOM_END1
</input>
```

BUTTON_ACTION:

```
<input class="langFont" title="ANY_STRING" style="BUTTON_ACTION_STYLE" type="button"
(value="NON_EMPTY_STRING")? xd:xctname="Button" xd:CtrlId="CONTROL_ID"
(xd:action="BUTTON_ACTION_TYPE")? (xd:auxDom="AUX_DOM_SOURCE_NAME")? (tabIndex="TAB_INDEX")?
(xd:postbackModel="BUTTON_POSTBACKMODEL")? (accessKey="SINGLE_CHARACTER")? (size="SIZE")?/>
```

BUTTON_ACTION_WITH_CONDITIONAL_FORMATTING:

```
<input class="langFont" title="ANY_STRING" type="button" (value="NON_EMPTY_STRING")?
xd:xctname="Button" xd:CtrlId="CONTROL_ID" (xd:action="BUTTON_ACTION_TYPE")?
(xd:auxDom="AUX_DOM_SOURCE_NAME")? (tabIndex="TAB_INDEX")? (style="BUTTON_ACTION_STYLE")?
(xd:postbackModel="BUTTON_POSTBACKMODEL")? (accessKey="SINGLE_CHARACTER")? (size="SIZE")?>
    BUTTON_ACTION_CONDITIONAL_FORMATTING
</input>
```

BUTTON_ACTION_WITH_DYNAMIC_DISPLAY_NAME_AND_CONDITIONAL_FORMATTING:

```
<input class="langFont" title="ANY_STRING" type="button" xd:xctname="Button"
xd:CtrlId="CONTROL_ID" xd:action="BUTTON_ACTION_TYPE" (xd:auxDom="AUX_DOM_SOURCE_NAME")?
(tabIndex="TAB_INDEX")? (style="BUTTON_ACTION_STYLE")?
(xd:postbackModel="BUTTON_POSTBACKMODEL")? (accessKey="SINGLE_CHARACTER")? (size="SIZE")?>
    BUTTON_ACTION_CONDITIONAL_FORMATTING
    CHECK_FOR_GETDOM_BEGIN1
    <xsl:attribute name="value">
        <xsl:value-of select="STRING_XPATH_EXPRESSION"/>
    </xsl:attribute>
    CHECK_FOR_GETDOM_END1
</input>
```

The following table lists control-specific attributes used by the button control.

Attributes	Section
xd:action	2.4.2.1
xd:auxDom	2.4.2.4
xd:CtrlId	2.4.2.10
xd:postbackModel	2.4.2.29

Attributes	Section
xd:xctname	2.4.2.35

The **xdEnvironment:IsBrowser** XSL function extension, as specified in section 2.4.3.3.1, is used by the button control.

2.4.1.6 Check Box Control

A check box control is a bi-state leaf control that has a value when it is selected and a different value when it is cleared. The following table describes the symbols for a check box control.

Symbol	Description
SIMPLE_CHECK_BOX	A control that has two states, **checked** and **unchecked**. The **unchecked** state tends to be represented as a blank white square, and the **checked** state has a mark, which is commonly a check mark, contained in the white square.
CHECK_BOX_WITH_CONDITIONAL_FORMATTING	Similar to **SIMPLE_CHECK_BOX**, with the addition that the control can be disabled conditionally. A disabled checkbox does not allow the user to directly toggle the control between its two states.

SIMPLE_CHECK_BOX:

```
<input class="xdBehavior_Boolean" title="ANY_STRING1" type="checkbox"
(accessKey="SINGLE_CHARACTER")? xd:binding="LEAF_XPATH1" xd:boundProp="xd:value"
(CHECK_BOX_SINGLE_VALUE | CHECK_BOX_BOTH_VALUES) (tabIndex="TAB_INDEX")?
xd:xctname="CheckBox" xd:CtrlId="CONTROL_ID" (xd:postbackModel="POSTBACKMODEL")?
(style="CHECK_BOX_STYLE")?>
    CHECK_FOR_GETDOM_BEGIN1
    <xsl:attribute name="xd:value">
        <xsl:value-of select="LEAF_XPATH1" />
    </xsl:attribute>
    <xsl:if test="BOOLEAN_XPATH_EXPRESSION">
        <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
    </xsl:if>
    CHECK_FOR_GETDOM_END1
</input>
(ANY_STRING2)?
```

CHECK_BOX_WITH_CONDITIONAL_FORMATTING:

```
<input class="xdBehavior_Boolean" title="ANY_STRING1" type="checkbox"
(accessKey="SINGLE_CHARACTER")? xd:binding="LEAF_XPATH1" xd:boundProp="xd:value"
(CHECK_BOX_SINGLE_VALUE | CHECK_BOX_BOTH_VALUES) (tabIndex="TAB_INDEX")?
xd:xctname="CheckBox" xd:CtrlId="CONTROL_ID" (xd:postbackModel="POSTBACKMODEL")?
(style="CHECK_BOX_STYLE")?>
    CHECK_FOR_GETDOM_BEGIN1
    <xsl:attribute name="style">
        <xsl:choose>
            (<xsl:when test="BOOLEAN_XPATH_EXPRESSIONY">STYLE_CAPTION</xsl:when>)+
```

```
        </xsl:choose>
    </xsl:attribute>
    (<xsl:choose>
        (<xsl:when test="BOOLEAN_XPATH_EXPRESSIONY">
            <xsl:attribute name="disabled">true</xsl:attribute>
        </xsl:when>)+
    </xsl:choose>)?
    <xsl:attribute name="xd:value">
        <xsl:value-of select="LEAF_XPATH1" />
    </xsl:attribute>
    <xsl:if test="BOOLEAN_XPATH_EXPRESSION">
        <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
    </xsl:if>
    CHECK_FOR_GETDOM_END1
</input>
(ANY_STRING2)?
```

CHECK_BOX_ONVALUE:

```
xd:onValue="(ISO_646_DIGIT+)|(&quot;ANY_STRING&quot;)"
```

CHECK_BOX_OFFVALUE:

```
xd:offValue="(ISO_646_DIGIT+)|(&quot;ANY_STRING&quot;)"
```

CHECK_BOX_SINGLE_VALUE: MUST be "**CHECK_BOX_OFFVALUE**" or "**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**?).

The following table lists control-specific attributes used by the check box control.

Attribute	Section
xd:binding	2.4.2.6
xd:boundProp	2.4.2.9
xd:CtrlId	2.4.2.10
xd:offValue	2.4.2.27
xd:onValue	2.4.2.28
xd:postbackModel	2.4.2.29
xd:value	2.4.2.34

Attribute	Section
xd:xctname	2.4.2.35

2.4.1.7 Contact Selector Control

The contact selector control provides the ability to select one or more entities from a user information list (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_XPATH1" xd:server="ANY_STRING"
(title="ANY_STRING")? (accessKey="SINGLE_CHARACTER")?
(xd:AllowMultiple="CONTACT_SELECTOR_ALLOW_MULTIPLE")?
(xd:SearchPeopleOnly="CONTACT_SELECTOR_SEARCH_PEOPLE_ONLY")?
(xd:SharePointGroup="CONTACT_SELECTOR_SHAREPOINT_GROUP")?>
    <xsl:if test="function-available('xdImage:getImageUrl')">
        <xsl:attribute name="src">
            <xsl:value-of select="xdImage:getImageUrl(GROUP_XPATH1)"/>
        </xsl:attribute>
    </xsl:if>
    (<xsl:attribute name="style">CONTACT_SELECTOR_STYLE?<xsl:choose>
            (<xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_DISPLAY_NONE;
STYLE_CAPTION</xsl:when>|
            <xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_CAPTION</xsl:when>)+
        </xsl:choose>
    </xsl:attribute>)?
    (<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_XPATH1"/>
    <param NAME="ObjectIdXPath" VALUE="CONTACT_SELECTOR_ACCOUNT_ID_XPATH1"/>
    <param NAME="ObjectTypeXPath" VALUE="CONTACT_SELECTOR_ACCOUNT_TYPE_XPATH1"/>
    <param NAME="SiteUrlXPath" VALUE="/Context/@siteUrl"/>
    <param NAME="SiteUrlDataSource" VALUE="Context"/>
    <param NAME="NewNodeTemplate" VALUE="CONTACT_SELECTOR_NEW_NODE_TEMPLATE"/>
    <param NAME="BackgroundColor" VALUE="CONTACT_SELECTOR_BACKGROUND_COLOR"/>
    <param NAME="MaxLines" VALUE="CONTACT_SELECTOR_MAX_LINES"/>
    <param NAME="Direction" VALUE="CONTACT_SELECTOR_DIRECTION"/>
</object>
```

The following table lists 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.

Parameter	Specification
ButtonText	This parameter specifies the text that is displayed on the button that opens the address book.
DisplayNameXPath	This parameter specifies the **LEAF_XPATH** containing the display names.
ObjectIdXPath	This parameter specifies the **LEAF_XPATH** containing the objectIDs.
ObjectTypeXPath	This parameter specifies the **LEAF_XPATH** containing the object types.
SiteUrlXPath	This parameter specifies the **LEAF_XPATH** in the secondary data source (2) containing the server URL, whose user information list (1) this control is querying. This parameter MUST 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 MUST be ignored by the form server.
NewNodeTemplate	This parameter specifies the XML template that is inserted in the form (1) when a new contact is selected.
BackgroundColor	This parameter specifies the background color of the contact selector input box.
MaxLines	This parameter specifies the maximum number of lines used by the contact selector input box to render display names.
Direction	This parameter specifies whether this control is displaying left-to-right or **right-to-left**.

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**?, (**BACKGROUND-COLOR**: transparent; **STYLE_XD_BACKGROUND_COLOR**)?, **STYLE_BORDER**?, **STYLE_FONT**?, **STYLE_COLOR**?, **STYLE_VERTICAL_ALIGN**?, **STYLE_DIRECTION**?)

CONTACT_SELECTOR_BACKGROUND_COLOR: "2147483653" or MUST be an integer value that represents a **red-green-blue (RGB)** color. The value MUST be calculated using three variables (blue part, red part, green part), each of which MUST be an **integer** between zero and 255, in the following formula:

blue part * 65536 + green part * 256 + red part

CONTACT_SELECTOR_MAX_LINES: MUST be an integer between zero and 999, inclusive.

CONTACT_SELECTOR_DIRECTION: Specifies if the control is rendered left-to-right or right-to-left. The following table lists the possible values and explanations.

Value	Description
"0"	Use the form's orientation.
"1"	Left-to-right.
"2"	Right-to-left.

CONTACT_SELECTOR_PERSON_XPATH: RELATIVE_REPREATING_GROUP_XPATH.

CONTACT_SELECTOR_DISPLAY_NAME_XPATH: **RELATIVE_LEAF_XPATH.**

CONTACT_SELECTOR_ACCOUNT_ID_XPATH: RELATIVE_LEAF_XPATH.

CONTACT_SELECTOR_ACCOUNT_TYPE_XPATH: **RELATIVE_LEAF_XPATH.**

CONTACT_SELECTOR_NEW_NODE_TEMPLATE:

```
&lt;CONTACT_SELECTOR_PERSON_XPATH1&gt;&#xA;

&lt;CONTACT_SELECTOR_DISPLAY_NAME_XPATH1&gt;&lt;/CONTACT_SELECTOR_DISPLAY_NAME_XPATH1&gt;&#xA
;

&lt;CONTACT_SELECTOR_ACCOUNT_ID_XPATH1&gt;&lt;/CONTACT_SELECTOR_ACCOUNT_ID_XPATH1&gt;&#xA;

&lt;CONTACT_SELECTOR_ACCOUNT_TYPE_XPATH1&gt;&lt;/CONTACT_SELECTOR_ACCOUNT_TYPE_XPATH1&gt;&#xA
;
&lt;/CONTACT_SELECTOR_PERSON_XPATH1&gt;
```

GROUP_XPATH: MUST point to an XML node in the main data source.

CONTACT_SELECTOR_ALLOW_MULTIPLE: "true" or "false".

CONTACT_SELECTOR_SEARCH_PEOPLE_ONLY: "true" or "false".

CONTACT_SELECTOR_SHAREPOINT_GROUP: Values MUST be those for **UserSelectionScope,** as specified in [MS-WSSFO2] section 2.2.8.3.3.1.

The following table lists control-specific attributes used by the contact selector control.

Attribute	Section
xd:AllowMultiple	2.4.2.37.1
xd:binding	2.4.2.6
xd:bindingProperty	2.4.2.7
xd:bindingType	2.4.2.8
xd:boundProp	2.4.2.9
xd:CtrlId	2.4.2.10
xd:disableEditing	2.4.2.12
xd:SearchPeopleOnly	2.4.2.37.7
xd:server	2.4.2.37.8
xd:SharePointGroup	2.4.2.37.9
xd:xctname	2.4.2.35

The **xdImage:getImageUrl** XSL function extension, as specified in section 2.4.3.5, is used by the contact selector control.

2.4.1.8 Date Picker Control

A date picker control is used to select and display a date. The following table describes the symbols for a date picker control.

Symbol	Description
SIMPLE_DATE_PICKER	A date picker is a control with the capabilities of displaying, as well as selecting, a date. This is usually accomplished by having a button that displays a view of a calendar. Clicking on the calendar allows the user to select a specific date. The date can also be manually entered in the date picker's display text box.
DATE_PICKER_WITH_CONDITIONAL_FORMATTING	Similar to **SIMPLE_DATE_PICKER,** except that it allows for conditional formatting. Conditional text formatting attributes such as bold, italics, and color can be applied to the displayed date. Conditional disabling disables both the text box and the date picker's calendar button. Conditionally hiding the control hides both the display box and the calendar button.
DATE_PICKER_WITH_DATA_FORMATTING	Similar to **SIMPLE_DATE_PICKER,** except that the data is formatted from its natural XML value.
DATE_PICKER_WITH_DATA_FORMATTING_AND_CONDITIONAL_FORMATTING	Similar to **DATE_PICKER_WITH_CONDITIONAL_FORMATTING,** except that the data is formatted from its natural XML value.
DATE_PICKER_WITH_PLACEHOLDER_TEXT	Similar to **SIMPLE_DATE_PICKER**, with the addition of text that is displayed in the control until actual data is entered. This displayed value is not persisted in the field (3) as a value.
DATE_PICKER_WITH_CONDITIONAL_FORMATTING_AND_PLACEHOLDER_TEXT	Similar to **DATE_PICKER_WITH_PLACEHOLDER_TEXT** and **DATE_PICKER_WITH_CONDITIONAL_FORMATTING**.
DATE_PICKER_WITH_DATA_FORMATTING_AND_PLACEHOLDER_TEXT	Similar to **DATE_PICKER_WITH_PLACEHOLDER_TEXT** and **DATE_PICKER_WITH_DATA_FORMATTING**.
DATE_PICKER_WITH_DATA_FORMATTING_AND_CONDITIONAL_FORMATTING_AND_PLACEHOLDER_TEXT	Similar to **DATE_PICKER_WITH_PLACEHOLDER_TEXT** and **DATE_PICKER_WITH_DATA_FORMATTING_AND_CONDITIONAL_FORMATTING**.

SIMPLE_DATE_PICKER:

```
<div class="xdDTPicker" title="ANY_STRING1" style="DATE_PICKER_STYLE" noWrap="1"
xd:CtrlId="CONTROL_ID" xd:xctname="DTPicker" (xd:postbackModel="POSTBACKMODEL")?>
    <span class="DATE_PICKER_TEXT_BOX_CLASS_NAME xdBehavior_NoBUI" hideFocus="1"
(title="ANY_STRING1")? (accessKey="SINGLE_CHARACTER")? xd:xctname="DTPicker_DTText"
xd:binding="LEAF_XPATH1" (tabIndex="TAB_INDEX1")? xd:innerCtrl="_DTText" (INPUT_SCOPE)?>
    CHECK_FOR_GETDOM_BEGIN1
        <xsl:value-of select="LEAF_XPATH1" />
    CHECK_FOR_GETDOM_END1
    </span>
    <button class="DATE_PICKER_BUTTON_CLASS_NAME" (title="ANY_STRING1")?
xd:xctname="DTPicker_DTButton" xd:innerCtrl="_DTButton" (tabIndex="TAB_INDEX1")?>
        <img (title="ANY_STRING1")? src="res://infopath.exe/calendar.gif" (Linked="true")?/>
    </button>
</div>
```

DATE_PICKER_WITH_CONDITIONAL_FORMATTING:

```
<div class="xdDTPicker" title="ANY_STRING1" (style="DATE_PICKER_STYLE")? noWrap="1"
xd:CtrlId="CONTROL_ID" xd:xctname="DTPicker" (xd:postbackModel="POSTBACKMODEL")?>
    <span class="DATE_PICKER_TEXT_BOX_CLASS_NAME xdBehavior_NoBUI" hideFocus="1"
(title="ANY_STRING1")? (accessKey="SINGLE_CHARACTER")? xd:xctname="DTPicker_DTText"
xd:binding="LEAF_XPATH1" (tabIndex="TAB_INDEX1")? xd:innerCtrl="_DTText" (INPUT_SCOPE)?>
    CHECK_FOR_GETDOM_BEGIN1
        (<xsl:attribute name="style">
            <xsl:choose>
                (<xsl:when
test="BOOLEAN_XPATH_EXPRESSIONx">(DATE_PICKER_STYLE_CONDITIONAL_FORMATTING)?</xsl:when>)*
            </xsl:choose>
        </xsl:attribute>)?
        (<xsl:choose>
            (<xsl:when test="BOOLEAN_XPATH_EXPRESSIONy">
                (<xsl:attribute name="contentEditable">false</xsl:attribute>)?
            </xsl:when>)*
        </xsl:choose>)?
        <xsl: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="TAB_INDEX1")?>
        <img (title="ANY_STRING1")? src="res://infopath.exe/calendar.gif" (Linked="true")?/>
    </button>
</div>
```

DATE_PICKER_WITH_DATA_FORMATTING:

```
<div class="xdDTPicker" title="ANY_STRING1" style="DATE_PICKER_STYLE" noWrap="1"
xd:CtrlId="CONTROL_ID" xd:xctname="DTPicker" (xd:postbackModel="POSTBACKMODEL")?>
    <span class="DATE_PICKER_TEXT_BOX_CLASS_NAME xdBehavior_FormattingNoBUI" hideFocus="1"
contentEditable="true" xd:xctname="DTPicker_DTText" xd:datafmt="DATA_FMT_CTRL_DATE_PICKER1"
DATA_FMT2_ATTRIBUTE_DATE_PICKER1 xd:boundProp="xd:num" xd:binding="LEAF_XPATH1"
(accessKey="SINGLE_CHARACTER")? (title="ANY_STRING1")? (tabIndex="TAB_INDEX1")?
xd:innerCtrl="_DTText" (INPUT_SCOPE)?>
    CHECK_FOR_GETDOM_BEGIN1
        (<xsl:attribute name="xd:num">
            <xsl:value-of select="LEAF_XPATH1" />
        </xsl:attribute>)?
```

```
        <xsl:choose>
            DATA_FMT2_FUNCTION_DATE_PICKER1
            <xsl:when test="function-available('xdFormatting:formatString')">
                <xsl:value-of select="xdFormatting:formatString(LEAF_XPATH1,
DATA_FMT_CTRL_DATE_PICKER1)" />
            </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="TAB_INDEX1")?>
        <img (title="ANY_STRING1")? src="res://infopath.exe/calendar.gif" (Linked="true")?/>
    </button>
</div>
```

DATE_PICKER_WITH_DATA_FORMATTING_AND_CONDITIONAL_FORMATTING:

```
<div class="xdDTPicker" title="ANY_STRING1" (style="DATE_PICKER_STYLE")? noWrap="1"
xd:CtrlId="CONTROL_ID" xd:xctname="DTPicker" (xd:postbackModel="POSTBACKMODEL")?>
    <span class="DATE_PICKER_TEXT_BOX_CLASS_NAME xdBehavior_FormattingNoBUI" hideFocus="1"
contentEditable="true" xd:xctname="DTPicker_DTText" xd:datafmt="DATA_FMT_CTRL_DATE_PICKER1"
DATA_FMT2_ATTRIBUTE_DATE_PICKER1 xd:boundProp="xd:num" xd:binding="LEAF_XPATH1"
(accessKey="SINGLE_CHARACTER")? (title="ANY_STRING1")? (tabIndex="TAB_INDEX1")?
xd:innerCtrl="_DTText" (INPUT_SCOPE)?>
    CHECK_FOR_GETDOM_BEGIN1
        (<xsl:attribute name="style">
            <xsl:choose>
                (<xsl:when
test="BOOLEAN_XPATH_EXPRESSIONx">(DATE_PICKER_STYLE_CONDITIONAL_FORMATTING)?</xsl:when>)*
            </xsl:choose>
        </xsl:attribute>)?
        (<xsl:choose>
            (<xsl:when test="BOOLEAN_XPATH_EXPRESSIONy">
                (<xsl:attribute name="contentEditable">false</xsl:attribute>)?
            </xsl:when>)*
        </xsl:choose>)?
        (<xsl:attribute name="xd:num">
            <xsl:value-of select="LEAF_XPATH1" />
        </xsl:attribute>)?
        <xsl:choose>
            DATA_FMT2_FUNCTION_DATE_PICKER1
            <xsl:when test="function-available('xdFormatting:formatString')">
                <xsl:value-of select="xdFormatting:formatString(LEAF_XPATH1,
DATA_FMT_CTRL_DATE_PICKER1)" />
            </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="TAB_INDEX1")?>
```

```
        <img (title="ANY_STRING1")? src="res://infopath.exe/calendar.gif" (Linked="true")?/>
    </button>
</div>
```

DATE_PICKER_WITH_PLACEHOLDER_TEXT:

```
<div class="xdDTPicker" title="ANY_STRING1" style="DATE_PICKER_STYLE" noWrap="1"
xd:CtrlId="CONTROL_ID" xd:xctname="DTPicker" (xd:postbackModel="POSTBACKMODEL")?>
    <span class="DATE_PICKER_TEXT_BOX_CLASS_NAME xdBehavior_GhostedTextNoBUI" hideFocus="1"
contentEditable="true" (title="ANY_STRING1")? (accessKey="SINGLE_CHARACTER")?
xd:xctname="DTPicker_DTText" xd:binding="LEAF_XPATH1" (tabIndex="TAB_INDEX1")?
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="TAB_INDEX1")?>
        <img (title="ANY_STRING1")? src="res://infopath.exe/calendar.gif" (Linked="true")?/>
    </button>
</div>
```

DATE_PICKER_WITH_CONDITIONAL_FORMATTING_AND_PLACEHOLDER_TEXT:

```
<div class="xdDTPicker" title="ANY_STRING1" (style="DATE_PICKER_STYLE")? noWrap="1"
xd:CtrlId="CONTROL_ID" xd:xctname="DTPicker" (xd:postbackModel="POSTBACKMODEL")?>
    <span class="DATE_PICKER_TEXT_BOX_CLASS_NAME xdBehavior_GhostedTextNoBUI" hideFocus="1"
contentEditable="true" (title="ANY_STRING1")? (accessKey="SINGLE_CHARACTER")?
xd:xctname="DTPicker_DTText" xd:binding="LEAF_XPATH1" (tabIndex="TAB_INDEX1")?
xd:innerCtrl="_DTText" (INPUT_SCOPE)?>
    CHECK_FOR_GETDOM_BEGIN1
        (<xsl:attribute name="style">
            <xsl:choose>
                (<xsl:when
test="BOOLEAN_XPATH_EXPRESSIONx">(DATE_PICKER_STYLE_CONDITIONAL_FORMATTING)?</xsl:when>)*
            </xsl:choose>
        </xsl:attribute>)?
        (<xsl:choose>
            (<xsl:when test="BOOLEAN_XPATH_EXPRESSIONy">
                (<xsl:attribute name="contentEditable">false</xsl:attribute>)?
            </xsl:when>)*
        </xsl:choose>)?
        (<xsl:choose>
            <xsl:when test="not(string(LEAF_XPATH1))">
                <xsl:attribute name="xd:ghosted">true</xsl:attribute>
                    ANY_STRING
```

```
                </xsl:when>
                <xsl:otherwise>
                    <xsl:value-of select="LEAF_XPATH1" />
                </xsl:otherwise>
            </xsl:choose>) | (<xsl:value-of select="LEAF_XPATH1" />)
        CHECK_FOR_GETDOM_END1
        </span>
        <button class="DATE_PICKER_BUTTON_CLASS_NAME" (title="ANY_STRING1")?
    xd:xctname="DTPicker_DTButton" xd:innerCtrl="_DTButton" (tabIndex="TAB_INDEX1")?>
            <img (title="ANY_STRING1")? src="res://infopath.exe/calendar.gif" (Linked="true")?/>
        </button>
    </div>
```

DATE_PICKER_WITH_DATA_FORMATTING_AND_PLACEHOLDER_TEXT:

```
    <div class="xdDTPicker" title="ANY_STRING1" style="DATE_PICKER_STYLE" noWrap="1"
    xd:CtrlId="CONTROL_ID" xd:xctname="DTPicker" (xd:postbackModel="POSTBACKMODEL")?>
        <span class="DATE_PICKER_TEXT_BOX_CLASS_NAME xdBehavior_GTFormattingNoBUI" hideFocus="1"
    contentEditable="true" contentEditable="true" xd:xctname="DTPicker_DTText"
    xd:datafmt="DATA_FMT_CTRL_DATE_PICKER1" DATA_FMT2_ATTRIBUTE_DATE_PICKER1
    xd:boundProp="xd:num" xd:binding="LEAF_XPATH1" (accessKey="SINGLE_CHARACTER")?
    (title="ANY_STRING1")? (tabIndex="TAB_INDEX1")? xd:innerCtrl="_DTText" (INPUT_SCOPE)?>
        CHECK_FOR_GETDOM_BEGIN1
            (<xsl:attribute name="xd:num">
                <xsl:value-of select="LEAF_XPATH1" />
            </xsl:attribute>)?
            <xsl:choose>
                (<xsl:when test="not(string(LEAF_XPATH1))">
                    <xsl:attribute name="xd:ghosted">true</xsl:attribute>
                    ANY_STRING
                </xsl:when>)?
                DATA_FMT2_FUNCTION_DATE_PICKER1
                <xsl:when test="function-available('xdFormatting:formatString')">
                    <xsl:value-of select="xdFormatting:formatString(LEAF_XPATH1,
    DATA_FMT_CTRL_DATE_PICKER1)" />
                </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="TAB_INDEX1")?>
            <img (title="ANY_STRING1")? src="res://infopath.exe/calendar.gif" (Linked="true")?/>
        </button>
    </div>
```

DATE_PICKER_WITH_DATA_FORMATTING_AND_CONDITIONAL_FORMATTING_AND_PLACEHOLDER_TEXT:

```
    <div class="xdDTPicker" title="ANY_STRING1" (style="DATE_PICKER_STYLE")? noWrap="1"
    xd:CtrlId="CONTROL_ID" xd:xctname="DTPicker" (xd:postbackModel="POSTBACKMODEL")?>
```

```
    <span class="DATE_PICKER_TEXT_BOX_CLASS_NAME xdBehavior_GTFormattingNoBUI" hideFocus="1"
contentEditable="true" contentEditable="true" xd:xctname="DTPicker_DTText"
xd:datafmt="DATA_FMT_CTRL_DATE_PICKER1" DATA_FMT2_ATTRIBUTE_DATE_PICKER1
xd:boundProp="xd:num" xd:binding="LEAF_XPATH1" (accessKey="SINGLE_CHARACTER")?
(title="ANY_STRING1")? (tabIndex="TAB_INDEX1")? xd:innerCtrl="_DTText" (INPUT_SCOPE)?>
    CHECK_FOR_GETDOM_BEGIN1
        (<xsl:attribute name="style">
            <xsl:choose>
                (<xsl:when
test="BOOLEAN_XPATH_EXPRESSIONx">(DATE_PICKER_STYLE_CONDITIONAL_FORMATTING)?</xsl:when>)*
            </xsl:choose>
        </xsl:attribute>)?
        <xsl:choose>
            (<xsl:when test="BOOLEAN_XPATH_EXPRESSIONy">
                (<xsl:attribute name="contentEditable">false</xsl:attribute>)?
            </xsl:when>)*
        </xsl:choose>
        (<xsl:attribute name="xd:num">
            <xsl:value-of select="LEAF_XPATH1" />
        </xsl:attribute>)?
        <xsl:choose>
            (<xsl:when test="not(string(LEAF_XPATH1))">
                <xsl:attribute name="xd:ghosted">true</xsl:attribute>
                ANY_STRING
            </xsl:when>)?
            DATA_FMT2_FUNCTION_DATE_PICKER1
            <xsl:when test="function-available('xdFormatting:formatString')">
                <xsl:value-of select="xdFormatting:formatString(LEAF_XPATH1,
DATA_FMT_CTRL_DATE_PICKER1)" />
            </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="TAB_INDEX1")?>
        <img (title="ANY_STRING1")? src="res://infopath.exe/calendar.gif" (Linked="true")?/>
    </button>
</div>
```

DATE_PICKER_TEXT_BOX_CLASS_NAME: xdDTText or **xdDTTextRTL**.

DATE_PICKER_BUTTON_CLASS_NAME: **xdDTButton** or **xdDTButtonRTL**.

DATE_PICKER_STYLE: Semicolon-delimited list of (**STYLE_WIDTH**?, **STYLE_PADDING**?, **STYLE_FONT**?, **STYLE_HEIGHT**?, **STYLE_TEXT_ALIGN**?, **STYLE_MARGIN**?, **WHITE-SPACE**: nowrap?, **STYLE_TEXT_DECORATION**?, **STYLE_BORDER**?, **STYLE_VERTICAL_ALIGN**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_COLOR**?, **STYLE_DIRECTION**?)

DATE_PICKER_STYLE_CONDITIONAL_FORMATTING: Semicolon-delimited list of (**STYLE_FONT_WEIGHT**?, **STYLE_COLOR**?, **STYLE_TEXT_DECORATION**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_FONT_STYLE**?, **STYLE_CAPTION**)

DATA_FMT2_ATTRIBUTE_DATE_PICKER:

```
(xd:datafmt2="DATA_FMT21")?
```

DATA_FMT2_FUNCTION_DATE_PICKER:

```
(<xsl:when test="function-available('xdFormatting:formatString2')">
    <xsl:value-of select="xdFormatting:formatString2(LEAF_XPATH1, DATA_FMT_CTRL_DATE_PICKER1,
'DATA_FMT21')" />
</xsl:when>)?
```

DATA_FMT2_ATTRIBUTE_DATE_PICKER and **DATA_FMT2_FUNCTION_DATE_PICKER** symbols always appear in pairs in the EBNF rules. Subscripts are used to mark the pairs.

If the yield of **DATA_FMT2_ATTRIBUTE_DATE_PICKER** in one production is empty, the yield of the pairing **DATA_FMT2_FUNCTION_DATE_PICKER** MUST be empty.

If the yield of **DATA_FMT2_FUNCTION_DATE_PICKER** in one production is empty, the yield of the pairing **DATA_FMT2_ATTRIBUTE_DATE_PICKER** MUST be empty.

The following table list control-specific attributes used by the date picker control.

Attribute	Section
xd:allowNonMatching	2.4.2.2
xd:binding	2.4.2.6
xd:boundProp	2.4.2.9
xd:CtrlId	2.4.2.10
xd:datafmt	2.4.2.11
xd:datafmt2	2.4.2.37.4
xd:innerCtrl	2.4.2.19
xd:inputScope	2.4.2.20
xd:num	2.4.2.26
xd:postbackModel	2.4.2.29
xd:xctname	2.4.2.35

The **xdFormatting:formatString** XSL function extension, as specified in section 2.4.3.4, is used by the contact selector control.

2.4.1.9 Drop-Down List Control

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

Symbol	Description
SIMPLE_DROPDOWN_LIST_BOX	A drop down list control 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**, except that it allows conditional formatting (text formatting and disabling).
DROPDOWN_LIST_BOX_WTIH_VALUES_FROM_DATA_SOURCE	Similar to **SIMPLE_DROPDOWN_LIST_BOX**, with the exception that the values for the collection are drawn from another location within the form's data source (2).
DROPDOWN_LIST_BOX_WITH_VALUES_FROM_DATA_SOURCE_AND_CONDITIONAL_FORMATTING	Similar to **DROPDOWN_LIST_BOX_WITH_VALUES_FROM_EXTERNAL_DATA_SOURCE** and **DROPDOWN_LIST_BOX_WTIH_VALUES_FROM_DATA_SOURCE_AND_CONDITIONAL_FORMATTING**.
DROPDOWN_LIST_BOX_WITH_VALUES_FROM_DATA_SOURCE_AND_UNIQUE_DISPLAY_NAMES	Similar to **DROPDOWN_LIST_BOX_WITH_VALUES_FROM_DATA_SOURCE**, with the exception that each value from the collection of values is unique.
DROPDOWN_LIST_BOX_WITH_VALUES_FROM_DATA_SOURCE_AND_UNIQUE_DISPLAY_NAMES_AND_CONDITIONAL_FORMATTING	Similar to **DROPDOWN_LIST_BOX_WITH_VALUES_FROM_DATA_SOURCE_AND_UNIQUE_DISPLAY_NAMES** and **DROPDOWN_LIST_BOX_WITH_CONDITIONAL_FORMATTING**.

SIMPLE_DROPDOWN_LIST_BOX:

```
<select class="xdComboBox xdBehavior_Select" title="ANY_STRING1"
(accessKey="SINGLE_CHARACTER")? size="FONT_SIZE" xd:binding="LEAF_XPATH1"
xd:boundProp="value" xd:xctname="dropdown" (xd:postbackModel="POSTBACKMODEL")?
(tabIndex="TAB_INDEX")? xd:CtrlId="CONTROL_ID" (style="DROPDOWN_LIST_BOX_STYLES")?>
    CHECK_FOR_GETDOM_BEGIN1
    <xsl:attribute name="value">
        <xsl:value-of select="LEAF_XPATH1" />
    </xsl:attribute>
    ((<option (value="")?>
        <xsl:if test="LEAF_XPATH1=&amp;quot;&amp;quot;">
            <xsl:attribute name="selected">selected</xsl:attribute>
        </xsl:if>
        ANY_STRING2
    </option>) |
    (<option value="LEAF_VALUE1">
        <xsl:if test="LEAF_XPATH1=LEAF_VALUE1">
            <xsl:attribute name="selected">selected</xsl:attribute>
        </xsl:if>
```

```
        ANY_STRINGX
    </option>))+
    CHECK_FOR_GETDOM_END1
</select>
```

DROPDOWN_LIST_BOX_WITH_CONDITIONAL_FORMATTING:

```
<select class="xdComboBox xdBehavior_Select" title="ANY_STRING1"
(accessKey="SINGLE_CHARACTER")? (style="DROPDOWN_LIST_BOX_STYLES")? size="FONT_SIZE"
xd:binding="LEAF_XPATH1" xd:boundProp="value" xd:xctname="dropdown"
(xd:postbackModel="POSTBACKMODEL")? (tabIndex="TAB_INDEX")? xd:CtrlId="CONTROL_ID">
    CHECK_FOR_GETDOM_BEGIN1
    (<xsl:attribute name="style">
        DROPDOWN_LIST_BOX_STYLES
        <xsl:choose>
            (DROPDOWN_LIST_BOX_CONDITIONAL_FORMATTING)*
        </xsl:choose>
    </xsl:attribute>)?
    (<xsl:choose>
        (DROPDOWN_LIST_BOX_CONDITIONAL_FORMATTING | DROPDOWN_LIST_BOX_CONDITIONAL_DISABLING)+
    </xsl:choose>)?
    <xsl:attribute name="value">
        <xsl:value-of select="LEAF_XPATH1" />
    </xsl:attribute>
    ((<option (value="")?>
        <xsl:if test="LEAF_XPATH1=&amp;quot;&amp;quot;">
            <xsl:attribute name="selected">selected</xsl:attribute>
        </xsl:if>
        ANY_STRING2
    </option>) |
    (<option value="LEAF_VALUE1">
        <xsl:if test="LEAF_XPATH1=LEAF_VALUE1">
            <xsl:attribute name="selected">selected</xsl:attribute>
        </xsl:if>
        ANY_STRINGX
    </option>))+
    CHECK_FOR_GETDOM_END1
</select>
```

DROPDOWN_LIST_BOX_WITH_VALUES_FROM_DATA_SOURCE:

```
<select class="xdComboBox xdBehavior_Select" title="ANY_STRING1"
(accessKey="SINGLE_CHARACTER")? size="FONT_SIZE" xd:binding="LEAF_XPATH1"
xd:boundProp="value" value="ANY_STRING2" xd:xctname="dropdown"
(xd:postbackModel="POSTBACKMODEL")? (tabIndex="TAB_INDEX")? xd:CtrlId="CONTROL_ID"
(style="DROPDOWN_LIST_BOX_STYLES")?>
    CHECK_FOR_GETDOM_BEGIN1
    <xsl:attribute name="value">
        <xsl:value-of select="LEAF_XPATH1" />
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="function-available('xdXDocument:GetDOM')">
            (<option />)?
            <xsl:variable name="val" select="LEAF_XPATH1" />
```

```
            <xsl:if test="not(REPEATING_LEAF_XPATH1([PREDICATE_XPATH1])?[LEAF_XPATH=$val] or
$val='')">
                <option selected="selected">
                    <xsl:attribute name="value">
                        <xsl:value-of select="$val" />
                    </xsl:attribute>
                    <xsl:value-of select="$val" />
                </option>
            </xsl:if>
            <xsl:for-each select="REPEATING_LEAF_XPATH1([PREDICATE_XPATH1])?">
                <option>
                    <xsl:attribute name="value">
                        <xsl:value-of select="RELATIVE_LEAF_XPATH1" />
                    </xsl:attribute>
                    <xsl:if test="$val=RELATIVE_LEAF_XPATH1">
                        <xsl:attribute name="selected">selected</xsl:attribute>
                    </xsl:if>
                    <xsl:value-of select="RELATIVE_LEAF_XPATH2"/>
                </option>
            </xsl:for-each>
        </xsl:when>
        <xsl:otherwise>
            <option>
                <xsl:value-of select="LEAF_XPATH1" />
            </option>
        </xsl:otherwise>
    </xsl:choose>
    CHECK_FOR_GETDOM_END1
</select>
```

DROPDOWN_LIST_BOX_WITH_VALUES_FROM_DATA_SOURCE_AND_CONDITIONAL_FORMATTING:

```
<select class="xdComboBox xdBehavior_Select" title="ANY_STRING1"
(accessKey="SINGLE_CHARACTER")? (style="DROPDOWN_LIST_BOX_STYLES")? size="FONT_SIZE"
xd:binding="LEAF_XPATH1" xd:boundProp="value" value="ANY_STRING2" xd:xctname="dropdown"
(xd:postbackModel="POSTBACKMODEL")? (tabIndex="TAB_INDEX")? xd:CtrlId="CONTROL_ID">
    CHECK_FOR_GETDOM_BEGIN1
    (<xsl:attribute name="style">
        DROPDOWN_LIST_BOX_STYLES
        <xsl:choose>
            (DROPDOWN_LIST_BOX_CONDITIONAL_FORMATTING)*
        </xsl:choose>
    </xsl:attribute>)?
    (<xsl:choose>
                (DROPDOWN_LIST_BOX_CONDITIONAL_FORMATTING |
DROPDOWN_LIST_BOX_CONDITIONAL_DISABLING)+
    </xsl:choose>)?
    <xsl:attribute name="value">
        <xsl:value-of select="LEAF_XPATH1" />
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="function-available('xdXDocument:GetDOM')">
            (<option />)?
            <xsl:variable name="val" select="LEAF_XPATH1" />
```

```
            <xsl:if test="not(REPEATING_LEAF_XPATH1([PREDICATE_XPATH1])?[LEAF_XPATH=$val] or
$val='')">
                <option selected="selected">
                    <xsl:attribute name="value">
                        <xsl:value-of select="$val" />
                    </xsl:attribute>
                    <xsl:value-of select="$val" />
                </option>
            </xsl:if>
            <xsl:for-each select="REPEATING_LEAF_XPATH1([PREDICATE_XPATH1])?">
                <option>
                    <xsl:attribute name="value">
                        <xsl:value-of select="RELATIVE_LEAF_XPATH1" />
                    </xsl:attribute>
                    <xsl:if test="$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([PREDICATE_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([PREDICATE_XPATH1])?" />
            </xsl:variable>
            <xsl:variable name="uniqueItems" select="msxsl:node-set($items)/*[not(LEAF_XPATH
= preceding::LEAF_XPATH2)]" />
            <xsl:for-each select="$uniqueItems">
                <option>
                    <xsl:attribute name="value">
                        <xsl:value-of select="RELATIVE_LEAF_XPATH1" />
                    </xsl:attribute>
                    <xsl:if test="$val=RELATIVE_LEAF_XPATH1">
                        <xsl:attribute name="selected">selected</xsl:attribute>
                    </xsl:if>
                    <xsl:value-of select="RELATIVE_LEAF_XPATH2"/>
                </option>
            </xsl:for-each>
        </xsl:when>
        <xsl:otherwise>
            <option>
                <xsl:value-of select="LEAF_XPATH1" />
            </option>
        </xsl:otherwise>
    </xsl:choose>
    CHECK_FOR_GETDOM_END1
</select>
```

DROPDOWN_LIST_BOX_WITH_VALUES_FROM_DATA_SOURCE_AND_UNIQUE_DISPLAY_NAMES_AND_CONDITIONAL_FORMATTING:

```
<select class="xdComboBox xdBehavior_Select" title="ANY_STRING1"
(accessKey="SINGLE_CHARACTER")? (style="DROPDOWN_LIST_BOX_STYLES")? size="FONT_SIZE"
xd:binding="LEAF_XPATH1" xd:boundProp="value" value="ANY_STRING2" xd:xctname="dropdown"
(xd:postbackModel="POSTBACKMODEL ")? (tabIndex="TAB_INDEX")? xd:CtrlId="CONTROL_ID">
    CHECK_FOR_GETDOM_BEGIN1
    (<xsl:attribute name="style">
        DROPDOWN_LIST_BOX_STYLES
        <xsl:choose>
            (DROPDOWN_LIST_BOX_CONDITIONAL_FORMATTING)*
        </xsl:choose>
    </xsl:attribute>)?
    (<xsl:choose>
        (DROPDOWN_LIST_BOX_CONDITIONAL_FORMATTING | DROPDOWN_LIST_BOX_CONDITIONAL_DISABLING)+
    </xsl:choose>)?
    <xsl:attribute name="value">
        <xsl:value-of select="LEAF_XPATH1" />
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="function-available('xdXDocument:GetDOM')">
            (<option />)?
            <xsl:variable name="val" select="LEAF_XPATH1" />
            <xsl:if test="not(REPEATING_LEAF_XPATH1([PREDICATE_XPATH1])?[LEAF_XPATH=$val] or
$val='')">
                <option selected="selected">
                    <xsl:attribute name="value">
                        <xsl:value-of select="$val" />
                    </xsl:attribute>
```

```
                    <xsl:value-of select="$val" />
                </option>
            </xsl:if>
            <xsl:variable name="items">
                <xsl:copy-of select="REPEATING_LEAF_XPATH1([PREDICATE_XPATH1])?" />
            </xsl:variable>
            <xsl:variable name="uniqueItems" select="msxsl:node-set($items)/*[not(LEAF_XPATH
= preceding::LEAF_XPATH2)]" />
            <xsl:for-each select="$uniqueItems">
                <option>
                    <xsl:attribute name="value">
                        <xsl:value-of select="RELATIVE_LEAF_XPATH1" />
                    </xsl:attribute>
                    <xsl:if test="$val=RELATIVE_LEAF_XPATH1">
                        <xsl:attribute name="selected">selected</xsl:attribute>
                    </xsl:if>
                    <xsl:value-of select="RELATIVE_LEAF_XPATH2"/>
                </option>
            </xsl:for-each>
        </xsl:when>
        <xsl:otherwise>
            <option>
                <xsl:value-of select="LEAF_XPATH1" />
            </option>
        </xsl:otherwise>
    </xsl:choose>
    CHECK_FOR_GETDOM_END1
</select>
```

DROPDOWN_LIST_BOX_CONDITIONAL_DISABLING:

```
<xsl:when test="BOOLEAN_XPATH_EXPRESSIONY">
    <xsl:attribute name="disabled">true</xsl:attribute>
</xsl:when>
```

DROPDOWN_LIST_BOX_CONDITIONAL_FORMATTING:

```
<xsl:when
test="BOOLEAN_XPATH_EXPRESSIONX">(LEAF_CONTROL_CONDITIONAL_FORMATTING_CAPTION)?</xsl:when>
```

DROPDOWN_LIST_BOX_STYLES: Semicolon-delimited list of (**STYLE_WIDTH**?, **STYLE_FONT**?, **STYLE_MARGIN**?, **STYLE_VERTICAL_ALIGN**?, **STYLE_TEXT_DECORATION**?,**STYLE_COLOR**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_DIRECTION**?)

The following table lists control-specific attributes used by the drop down list control.

Attribute	Section
xd:binding	2.4.2.6
xd:boundProp	2.4.2.9
xd:CtrlId	2.4.2.10

Attribute	Section
xd:postbackModel	2.4.2.29
xd:xctname	2.4.2.35

The **xdXDocument:GetDOM** XSL function extension, as specified in section 2.4.3.9.2, is used by the drop down list control.

2.4.1.10 Expression Box Control

The expression box control is a read-only control that displays the result of an XPath evaluation. The following table describes the symbols for an expression box control.

Symbol	Description
SIMPLE_EXPRESSION_BOX	An expression box is a control that displays the value of an XPath expression. It is constantly disabled.
EXPRESSIONBOX_WITH_CONDITIONAL_FORMATTING	Similar to **SIMPLE_EXPRESSION_BOX,** with text formatting and conditional formatting.
EXPRESSIONBOX_WITH_DATA_FORMATTING	Similar to **SIMPLE_EXPRESSION_BOX,** with the result formatted as a type of data.
EXPRESSIONBOX_WITH_DATA_FORMATTING_AND_CONDITIONAL_FORMATTING	Similar to **EXPRESSIONBOX_WITH_CONDITIONAL_FORMATTING** and **EXPRESSIONBOX_WITH_DATA_FORMATTING.**

SIMPLE_EXPRESSION_BOX:

```
<span class="xdExpressionBox xdDataBindingUI (xdBehavior_Formatting)?" title="ANY_STRING"
(tabIndex="-1")? xd:xctname="ExpressionBox" xd:CtrlId="CONTROL_ID" xd:disableEditing="yes"
(xd:binding="EXPRESSION_BOX_XPATH1")? style="EXPRESSION_BOX_STYLE">
    CHECK_FOR_GETDOM_BEGIN1
    <xsl:value-of select="EXPRESSION_BOX_XPATH1" />
    CHECK_FOR_GETDOM_END1
</span>
```

EXPRESSIONBOX_WITH_CONDITIONAL_FORMATTING:

```
<span class="xdExpressionBox xdDataBindingUI" title="ANY_STRING" (tabIndex="-1")?
xd:xctname="ExpressionBox" xd:CtrlId="CONTROL_ID" xd:disableEditing="yes"
(xd:binding="EXPRESSION_BOX_XPATH1")?>
    CHECK_FOR_GETDOM_BEGIN1
    <xsl:attribute name="style">EXPRESSION_BOX_STYLE
       <xsl:choose>
             (<xsl:when
test="BOOLEAN_XPATH_EXPRESSIONX">LEAF_CONTROL_CONDITIONAL_FORMATTING_CAPTION</xsl:when>)+
        </xsl:choose>
    </xsl:attribute>
```

```
    <xsl:value-of select="EXPRESSION_BOX_XPATH1" />
    CHECK_FOR_GETDOM_END1
</span>
```

EXPRESSION_BOX_WITH_DATA_FORMATTING:

```
<span class="xdExpressionBox xdDataBindingUI( xdBehavior_Formatting)?" title="ANY_STRING"
(tabIndex="-1")? xd:xctname="ExpressionBox" xd:CtrlId="CONTROL_ID" xd:disableEditing="yes"
(xd:binding="EXPRESSION_BOX_XPATH1")? xd:datafmt="DATA_FMT_CTRL_EXPBOX1"
DATA_FMT2_ATTRIBUTE_EXPRESSION_BOX1 (xd:num="")? style="EXPRESSION_BOX_STYLE">
    CHECK_FOR_GETDOM_BEGIN1
    (<xsl:attribute name="xd:num">
        <xsl:value-of select="EXPRESSION_BOX_XPATH1" />
    </xsl:attribute>)?
    <xsl:choose>
        DATA_FMT2_FUNCTION_EXPRESSION_BOX1
        <xsl:when test="function-available('xdFormatting:formatString')">
            <xsl:value-of select="xdFormatting:formatString(EXPRESSION_BOX_XPATH1,
DATA_FMT_CTRL_EXPBOX1)" />
        </xsl:when>
        <xsl:otherwise>
            <xsl:value-of select="EXPRESSION_BOX_XPATH1" />
        </xsl:otherwise>
    </xsl:choose>
    CHECK_FOR_GETDOM_END1
</span>
```

EXPRESSION_BOX_WITH_DATA_FORMATTING_AND_CONDITIONAL_FORMATTING:

```
<span class="xdExpressionBox xdDataBindingUI( xdBehavior_Formatting)?" title="ANY_STRING"
(tabIndex="-1")? xd:xctname="ExpressionBox" xd:CtrlId="CONTROL_ID" xd:disableEditing="yes"
(xd:binding="EXPRESSION_BOX_XPATH1")? xd:datafmt="DATA_FMT_CTRL_EXPBOX1"
DATA_FMT2_ATTRIBUTE_EXPRESSION_BOX1 (xd:num="")?>
    CHECK_FOR_GETDOM_BEGIN1
    <xsl:attribute name="style">EXPRESSION_BOX_STYLE
       <xsl:choose>
            (<xsl:when
test="BOOLEAN_XPATH_EXPRESSIONx">LEAF_CONTROL_CONDITIONAL_FORMATTING_CAPTION</xsl:when>)+
        </xsl:choose>
    </xsl:attribute>
    (<xsl:attribute name="xd:num">
        <xsl:value-of select="EXPRESSION_BOX_XPATH1" />
    </xsl:attribute>)?
    <xsl:choose>
        DATA_FMT2_FUNCTION_EXPRESSION_BOX1
        <xsl:when test="function-available('xdFormatting:formatString')">
            <xsl:value-of select="xdFormatting:formatString(EXPRESSION_BOX_XPATH1,
DATA_FMT_CTRL_EXPBOX1)" />
        </xsl:when>
        <xsl:otherwise>
            <xsl:value-of select="EXPRESSION_BOX_XPATH1" />
        </xsl:otherwise>
    </xsl:choose>
    CHECK_FOR_GETDOM_END1
</span>
```

EXPRESSION_BOX_XPATH: LEAF_XPATH or **STRING_XPATH_EXPRESSION**.

EXPRESSION_BOX_OVERFLOW_Y: OVERFLOW-Y: auto.

EXPRESSION_BOX_OVERFLOW_X: OVERFLOW-X: auto or **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**?, **STYLE_TEXT_ALIGN**?)

DATA_FMT2_ATTRIBUTE_EXPRESSION_BOX:

```
(xd:datafmt2="DATA_FMT21")?
```

DATA_FMT2_FUNCTION_EXPRESSION_BOX:

```
(<xsl:when test="function-available('xdFormatting:formatString2')">
    <xsl:value-of select="xdFormatting:formatString2(LEAF_XPATH1, DATA_FMT_CTRL_EXPBOX1,
'DATA_FMT21')" />
</xsl:when>)?
```

DATA_FMT2_ATTRIBUTE_EXPRESSION_BOX and **DATA_FMT2_FUNCTION_EXPRESSION_BOX** symbols always appear in pairs in the EBNF rules. Subscripts are used to mark the pairs.

If the yield of **DATA_FMT2_ATTRIBUTE_EXPRESSION_BOX** in one production is empty, the yield of the pairing **DATA_FMT2_FUNCTION_EXPRESSION_BOX** MUST be empty.

If the yield of **DATA_FMT2_FUNCTION_EXPRESSION_BOX** in one production is empty, the yield of the pairing **DATA_FMT2_ATTRIBUTE_EXPRESSION_BOX** MUST be empty.

The following table lists control-specific attributes used by the expression box control.

Attribute	Section
xd:binding	2.4.2.6
xd:CtrlId	2.4.2.10
xd:datafmt	2.4.2.11
xd:datafmt2	2.4.2.37.4
xd:disableEditing	2.4.2.12
xd:num	2.4.2.26
xd:xctname	2.4.2.35

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.

The following table lists control-specific attributes used by the file attachment control.

Attribute	Section
xd:binding	2.4.2.6
xd:boundProp	2.4.2.9
xd:CtrlId	2.4.2.10
xd:disableEditing	2.4.2.12
xd:xctname	2.4.2.35

The **xdImage:getImageUrl** XSL function extension, as specified in section 2.4.3.5, is used by the file attachment control.

2.4.1.12 Hyperlink Control

The hyperlink control allows the user to create a hyperlink that navigates the default browser to a specified URL. The following table describes the symbols for a hyperlink control.

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

Symbol	Description
	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**.

SIMPLE_HYPERLINK:

```
<a href="ANY_STRING" (title="ANY_STRING")? (accessKey="SINGLE_CHARACTER")?
(tabIndex="TAB_INDEX")? xd:disableEditing="yes">ANCHOR_TEXT</a>
```

HYPERLINK_WITH_DYNAMIC_LINK:

```
<a class="xdDataBindingUI" (title="ANY_STRING")? (accessKey="SINGLE_CHARACTER")?
(tabIndex="TAB_INDEX")? xd:disableEditing="yes">
    CHECK_FOR_GETDOM_BEGIN1
    <xsl:attribute name="href">
        <xsl:value-of select="LEAF_XPATH"/>
    </xsl:attribute>
ANCHOR_TEXT
    CHECK_FOR_GETDOM_END1
</a>
```

HYPERLINK_WITH_DYNAMIC_TEXT:

```
<span class="xdHyperlink" hideFocus="1" (title="ANY_STRING")?
style="DYNAMIC_HYPERLINK_TEXT_STYLE" xd:xctname="hyperlink">
    <a class="xdDataBindingUI" (title="ANY_STRING")? (accessKey="SINGLE_CHARACTER")?
(tabIndex="TAB_INDEX")? href="ANY_STRING" xd:disableEditing="yes" xd:CtrlId="CONTROL_ID">
    CHECK_FOR_GETDOM_BEGIN1
        <xsl:value-of select="LEAF_XPATH"/>
    CHECK_FOR_GETDOM_END1
    </a>
</span>
```

HYPERLINK_WITH_DYNAMIC_LINK_AND_DYNAMIC_TEXT:

```
<span class="xdHyperlink" hideFocus="1" (title="ANY_STRING")?
style="DYNAMIC_HYPERLINK_TEXT_STYLE" xd:xctname="hyperlink">
    <a class="xdDataBindingUI" (title="ANY_STRING")? (accessKey="SINGLE_CHARACTER")?
(tabIndex="TAB_INDEX")? xd:disableEditing="yes" xd:CtrlId="CONTROL_ID">
    CHECK_FOR_GETDOM_BEGIN1
        <xsl:attribute name="href">
            <xsl:value-of select="LEAF_XPATH1"/>
        </xsl:attribute>
        <xsl:value-of select="LEAF_XPATH2"/>
    CHECK_FOR_GETDOM_END1
```

```
    </a>
</span>
```

DYNAMIC_HYPERLINK_TEXT_STYLE: Semicolon-delimited list of (**OVERFLOW**: visible, **STYLE_WIDTH**?, **STYLE_TEXT_ALIGN**?, **STYLE_BORDER**?, **STYLE_FONT**?, **STYLE_VERTICAL_ALIGN**?, **STYLE_TEXT_DECORATION**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_COLOR**?)

The following table lists control-specific attributes used by the hyperlink control.

Attribute	Section
xd:CtrlId	2.4.2.10
xd:disableEditing	2.4.2.12
xd:xctname	2.4.2.35

2.4.1.13 List Box Control

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

Symbol	Description
LIST_BOX_WITH_MANUAL_ENTRIES	The list box displays selection options that have been manually specified by the form template designer.
LIST_BOX_WITH_LOOKUP_ENTRIES	The list box displays selection options that are populated from a data source (2).
LIST_BOX_WITH_UNIQUE_LOOKUP_ENTRIES	The list box displays only unique selection options that are populated from a data source (2).

LIST_BOX_CONDITIONAL_FORMATTING:

```
(<xsl:attribute name="style">LIST_BOX_STYLE<xsl:choose>
        (<xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_DISPLAY_NONE;
STYLE_CAPTION</xsl:when>|
        <xsl:when test="BOOLEAN_XPATH_EXPRESSION"/>|
        <xsl:when
test="BOOLEAN_XPATH_EXPRESSION">LIST_BOX_CONDITIONAL_FORMATTING_STYLE</xsl:when>)+
    </xsl:choose>
</xsl:attribute>)?
(<xsl:choose>
    (<xsl:when test="BOOLEAN_XPATH_EXPRESSION"/>|
    <xsl:when test="BOOLEAN_XPATH_EXPRESSION">
        <xsl:attribute name="disabled">true</xsl:attribute>
    </xsl:when>)+
</xsl:choose>)?
```

LIST_BOX_WITH_MANUAL_ENTRIES:

```
<select class="xdListBox xdBehavior_Select" title="ANY_STRING" size="3"
(tabindex="TAB_INDEX")? xd:CtrlId="CONTROL_ID" xd:xctname="ListBox" xd:binding="LEAF_XPATH1"
xd:boundProp="value" (style="LIST_BOX_STYLE")? (xd:postbackModel="POSTBACKMODEL")?
(accessKey="SINGLE_CHARACTER")?>
    CHECK_FOR_GETDOM_BEGIN1
    LIST_BOX_CONDITIONAL_FORMATTING?
    <xsl:attribute name="value">
        <xsl:value-of select="LEAF_XPATH1"/>
    </xsl:attribute>
    ((<option>
        <xsl:if test="LEAF_XPATH1=&amp;quot;&amp;quot;">
            <xsl:attribute name="selected">selected</xsl:attribute>
        </xsl:if>OPTION_DISPLAY_VALUE?</option>)|
    (<option value="OPTION_VALUE1">
        <xsl:if test="LEAF_XPATH1=&amp;quot;OPTION_VALUE1&amp;quot;">
            <xsl:attribute name="selected">selected</xsl:attribute>
        </xsl:if>OPTION_DISPLAY_VALUE?</option>))+
    CHECK_FOR_GETDOM_END1
</select>
```

LIST_BOX_WITH_LOOKUP_ENTRIES:

```
<select class="xdListBox xdBehavior_Select" title="ANY_STRING" size="3"
(tabindex="TAB_INDEX")? xd:CtrlId="CONTROL_ID" xd:xctname="ListBox" xd:binding="LEAF_XPATH1"
xd:boundProp="value" (value="ANY_STRING")? (style="LIST_BOX_STYLE")?
(xd:postbackModel="POSTBACKMODEL")? (accessKey="SINGLE_CHARACTER")?>
    LIST_BOX_CONDITIONAL_FORMATTING?
    <xsl:attribute name="value">
        <xsl:value-of select="LEAF_XPATH1"/>
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="function-available('xdXDocument:GetDOM')">
            <option/>
            <xsl:variable name="val" select="LEAF_XPATH1"/>
            <xsl:if test="not(GROUP_XPATH1/RELATIVE_REPEATING_GROUP_XPATH1
([PREDICATE_XPATH1])?|( [RELATIVE_LEAF_XPATH1 =$val] or $val='')">
                <option selected="selected">
                    <xsl:attribute name="value">
                        <xsl:value-of select="$val"/>
                    </xsl:attribute>
                    <xsl:value-of select="$val"/>
                </option>
            </xsl:if>
            <xsl:for-each select="GROUP_XPATH1/RELATIVE_REPEATING_GROUP_XPATH1(
[PREDICATE_XPATH1])?">
                <option>
                    <xsl:attribute name="value">
                        <xsl:value-of select="RELATIVE_LEAF_XPATH1"/>
                    </xsl:attribute>
                    <xsl:if test="$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([PREDICATE_XPATH1])?
[RELATIVE_LEAF_XPATH1=$val] or $val='')">
                <option selected="selected">
                    <xsl:attribute name="value">
                        <xsl:value-of select="$val"/>
                    </xsl:attribute>
                    <xsl:value-of select="$val"/>
                </option>
            </xsl:if>
            <xsl:variable name="items">
                <xsl:copy-of select="GROUP_XPATH1/RELATIVE_REPEATING_GROUP_XPATH1(
[PREDICATE_XPATH1])?"/>
            </xsl:variable>
            <xsl:variable name="uniqueItems" select="msxsl:node-
set($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**?, **STYLE_CAPTION**).

The following table lists control-specific attributes used by the list box control.

Attribute	Section
xd:binding	2.4.2.6
xd:boundProp	2.4.2.9
xd:CtrlId	2.4.2.10
xd:postbackModel	2.4.2.29
xd:xctname	2.4.2.35

The **xdXDocument:GetDOM** XSL function extension, as specified in section 2.4.3.9.2, is used by the list box control.

2.4.1.14 Option Button Control

An option button is a bi-state control that has a value when selected and no value when not selected. Option button controls are meant to be used in groups (1), with selection among the option buttons in the group (1) being mutually exclusive. The following table describes the symbols for an option button control.

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	Similar to **SIMPLE_OPTION_BUTTON**, but allows the button to be conditionally disabled.

SIMPLE_OPTION_BUTTON:

```
<input class="xdBehavior_Boolean" title="ANY_STRING1" type="radio" name="{generate-
id(LEAF_XPATH1)}" (accessKey="SINGLE_CHARACTER")? xd:binding="LEAF_XPATH1"
xd:boundProp="xd:value" (xd:onValue="(ISO_646_DIGIT+)|(&quot;ANY_STRING2&quot;)")?
(tabIndex="LEAF_CONTROL_TAB_INDEX")? xd:xctname="OptionButton" xd:CtrlId="CONTROL_ID"
(xd:postbackModel="POSTBACKMODEL")? (style="OPTION_BUTTON_STYLE")?>
    CHECK_FOR_GETDOM_BEGIN1
    <xsl:attribute name="xd:value">
        <xsl:value-of select="LEAF_XPATH1" />
```

```
    </xsl:attribute>
    <xsl:if test="BOOLEAN_XPATH_EXPRESSION">
        <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
    </xsl:if>
    CHECK_FOR_GETDOM_END1
</input>
ANY_STRING3
```

OPTION_BUTTON_WITH_CONDITIONAL_FORMATTING:

```
<input class="xdBehavior_Boolean" title="ANY_STRING1" type="radio" name="{generate-
id(LEAF_XPATH1)}" (accessKey="SINGLE_CHARACTER")? xd:binding="LEAF_XPATH1"
xd:boundProp="xd:value" (xd:onValue="(ISO_646_DIGIT+)|(&quot;ANY_STRING2&quot;)")?
(tabIndex="LEAF_CONTROL_TAB_INDEX")? xd:xctname="OptionButton" xd:CtrlId="CONTROL_ID"
(xd:postbackModel="POSTBACKMODEL")? (style="OPTION_BUTTON_STYLE")?>
    CHECK_FOR_GETDOM_BEGIN1
    <xsl:attribute name="style">
        <xsl:choose>
            (<xsl:when test="BOOLEAN_XPATH_EXPRESSIONY">STYLE_CAPTION</xsl:when>)+
        </xsl:choose>
    </xsl:attribute>
    (<xsl:choose>
        (<xsl:when test="BOOLEAN_XPATH_EXPRESSIONY">
            <xsl:attribute name="disabled">true</xsl:attribute>
        </xsl:when>)+
    </xsl:choose>)?
    <xsl:attribute name="xd:value">
        <xsl:value-of select="LEAF_XPATH1" />
    </xsl:attribute>
    <xsl:if test="BOOLEAN_XPATH_EXPRESSION">
        <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
    </xsl:if>
    CHECK_FOR_GETDOM_END1
</input>
ANY_STRING3
```

OPTION_BUTTON_STYLE: Semicolon-delimited list of (**STYLE_MARGIN**?, **STYLE_FONT**?, **STYLE_VERTICAL_ALIGN**?, **STYLE_BORDER**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_COLOR**?, **STYLE_TEXT_DECORATION**?, **STYLE_WIDTH**?, **STYLE_HEIGHT**?).

The following table lists control-specific attributes used by the option button control.

Attribute	Section
xd:binding	2.4.2.6
xd:boundProp	2.4.2.9
xd:CtrlId	2.4.2.10
xd:onValue	2.4.2.28
xd:postbackModel	2.4.2.29
xd:value	2.4.2.34

Attribute	Section
xd:xctname	2.4.2.35

2.4.1.15 Repeating Section Control

A repeating section control acts as a container for other controls that can appear multiple times in the same form (1). The following table describes the symbols for a repeating section control.

Symbol	Description
REPEATING_SECTION_CALL	The first part of the repeating section control that indicates where the repeating sections is rendered.
REPEATING_SECTION_BODY_ WITHOUT_CONDITIONAL_HIDING	The second part of the repeating section that defines the properties of the repeating section and its content.
REPEATING_SECTION_BODY_ WITHOUT_CONDITIONAL_HIDING	The second part of the repeating section that defines the properties of the repeating section and its content. This part is used when the section is conditionally hidden.

REPEATING_SECTION MUST consist of a **REPEATING_SECTION_CALL** at the point in the XSL where the control appears, which is either the body of the main template or another XSL template, and a **REPEATING_SECTION_BODY** that defines the section and appears as a separate XSL template. **TEMPLATE_MODE_ID** values in **REPEATING_SECTION_CALL** and **REPEATING_SECTION_BODY** MUST match.

In any production defined in this section, any use of a **TAB_INDEX** with the same subscript MUST have an identical yield.

In any production defined in this section, any use of a **CONTROL_ID** with the same subscript MUST have an identical yield.

REPEATING_SECTION_CALL:

```
SIMPLE_SECTION_CALL |
(CHECK_FOR_GETDOM_BEGIN1
<xsl:apply-templates select="(GROUP_XPATH/)?RELATIVE_REPEATING_GROUP_XPATH"
mode="TEMPLATE_MODE_ID1"/>
    (<div class="optionalPlaceholder" xd:xmlToEdit="XML_TO_EDIT_NAME" tabIndex="TABINDEX1"
xd:action="xCollection::insert" align="ALIGN" style="STYLE_WIDTH">ANY_STRING</div>)?
CHECK_FOR_GETDOM_END1) |
(CHECK_FOR_GETDOM_BEGIN1
(<xsl:variable name=(&quot;filterParentHasNewRows(CONTROL_ID1)&quot;)
select=&quot;xdXDocument:GetNamedNodeProperty(RELATIVE_GROUP_XPATH1,&quot;filterHasNewRows&qu
ot;, &quot;false&quot;)&quot;/>
<xsl:variable name=(&quot;filterParentVersion(CONTROL_ID1)&quot;)
select=&quot;xdXDocument:GetNamedNodeProperty(RELATIVE_GROUP_XPATH1,&quot;parentFilterVersion
&quot;,&quot;0&quot;)&quot;/>)
<xsl:apply-templates select="(GROUP_XPATH/)?RELATIVE_REPEATING_GROUP_XPATH([PREDICATE_XPATH
or ($filterParentHasNewRows(CONTROL_ID1)= "true" and xdXDocument:GetNamedNodeProperty(.,
"filterVersion", "0") > $filterParentVersion(CONTROL_ID1))]) " mode="TEMPLATE_MODE_ID1"/>
    (<div class="optionalPlaceholder" xd:xmlToEdit="XML_TO_EDIT_NAME" tabIndex="TABINDEX1"
xd:action="xCollection::insert" align="ALIGN" style="STYLE_WIDTH">ANY_STRING</div>)?
CHECK_FOR_GETDOM_END1)
```

REPEATING_SECTION_BODY:
REPEATING_SECTION_BODY_WITHOUT_CONDITIONAL_HIDING or
REPEATING_SECTION_BODY_WITH_CONDITIONAL_HIDING.

REPEATING_SECTION_BODY_WITHOUT_CONDITIONAL_HIDING:

```
<xsl:template match="RELATIVE_REPEATING_GROUP_XPATH" mode="TEMPLATE_MODE_ID1">
    <div class="xdRepeatingSection xdRepeating" title="ANY_STRING" (style="SECTION_STYLE")?
align="ALIGN" xd:xctname="RepeatingSection" xd:CtrlId="CONTROL_ID1" (tabIndex="-1")?
xd:widgetIndex="TABINDEX1" (xd:postbackModel="POSTBACKMODEL")?>
        XML_HTML_4_1_WITH_CONTROLS
        (<xsl:attribute name="style">SECTION_STYLE(<xsl:choose>
                (<xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_CAPTION</xsl:when>|
                <xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_BACKGROUND_COLOR;
STYLE_CAPTION</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_ID1"
(tabIndex="-1")? xd:widgetIndex="TABINDEX1" (xd:postbackModel="POSTBACKMODEL")?
HIDDEN_FORMATTING_CAPTION>
            XML_HTML_4_1_WITH_CONTROLS
            (<xsl:attribute name="style">SECTION_STYLE(<xsl:choose>
                    (<xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_CAPTION</xsl:when>|
                    <xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_BACKGROUND_COLOR;
STYLE_CAPTION</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>
```

The following table lists control-specific attributes used by the repeating section control.

Attribute	Section
xd:action	2.4.2.1
xd:CtrlId	2.4.2.10
xd:postbackModel	2.4.2.29

Attribute	Section
xd:widgetIndex	2.4.2.37.10
xd:xctname	2.4.2.35
xd:xmlToEdit	2.4.2.36

2.4.1.16 Repeating Table Control

A repeating table control acts as a container for other controls, and can appear multiple times in an instance of a form (1). It has a tabular format. The following table describes the symbols for a repeating table control.

Symbol	Description
SIMPLE_REPEATING_TABLE	A repeating table is a structural control that overloads the HTML **table** element. It is possible to dynamically add and remove rows.
REPEATING_TABLE_WITH_CONDITIONAL_FORMATTING	Similar to **SIMPLE_REPEATING_TABLE**, with the ability to conditionally change **background color** and disable adding or removing rows.
REPEATING_TABLE_WITH_PREDICATE_XPATH	Similar to **SIMPLE_REPEATING_TABLE**, with the ability to filter table rows.
REPEATING_TABLE_WITH_PREDICATE_XPATH_AND_CONDITIONAL_FORMATTING	Similar to **REPEATING_TABLE_WITH_CONDITIONAL_FORMATTING**, with the ability to filter table rows.

In any production defined in this section, any use of a **TAB_INDEX** with the same subscript MUST have an identical yield.

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")? xd:widgetIndex="TABINDEX1"
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_INDEX1"
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")?
xd:widgetIndex="TABINDEX1">
    <colgroup>
        TABLE_COLUMN+
    </colgroup>
    (<tbody (valign="VALIGN")? (style="STYLE_DIRECTION")? class="xdTableHeader">
        TABLE_ROW*
    </tbody>)?
    <tbody (style="STYLE_DIRECTION")? xd:xctname="RepeatingTable">
    CHECK_FOR_GETDOM_BEGIN1
        <xsl:for-each select="GROUP_XPATH">
            REPEATING_TABLE_ROWS_WITH_CONDITIONALVISIBILITY |
TABLE_ROW_WITH_CONDITIONAL_FORMATTING*
        </xsl:for-each>
    CHECK_FOR_GETDOM_END1
    </tbody>
    (<tbody (valign="VALIGN")? (style="STYLE_DIRECTION")? class="xdTableFooter">
        TABLE_ROW*
    </tbody>)?
</table>
(<div class="optionalPlaceholder" xd:xmlToEdit="XML_TO_EDIT_NAME" tabIndex="TAB_INDEX1"
xd:action="xCollection::insert" style="STYLE_WIDTH">ANY_STRING</div>)?
```

REPEATING_TABLE_WITH_PREDICATE_XPATH:

```
<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_ID1" (xd:postBackModel="POST_BACK_MODEL_VALUE")?
xd:widgetIndex="TABINDEX1">
    <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:variable name=(&quot;filterParentHasNewRows(CONTROL_ID1)&quot;)
select=&quot;xdXDocument:GetNamedNodeProperty(RELATIVE_GROUP_XPATH1,&quot;filterHasNewRows&qu
ot;, &quot;false&quot;)&quot;/>
    <xsl:variable name=(&quot;filterParentVersion(CONTROL_ID1)&quot;)
select=&quot;xdXDocument:GetNamedNodeProperty(RELATIVE_GROUP_XPATH1,&quot;parentFilterVersion
&quot;,&quot;0&quot;)&quot;/>)
```

```
<xsl:for-each select="GROUP_XPATH([PREDICATE_XPATH or ($filterParentHasNewRows(CONTROL_ID1)=
"true" and xdXDocument:GetNamedNodeProperty(., "filterVersion", "0") >
$filterParentVersion(CONTROL_ID1))])">
        </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_INDEX1"
xd:action="xCollection::insert" style="STYLE_WIDTH">ANY_STRING</div>)?
```

REPEATING_TABLE_WITH_PREDICATE_XPATH_AND_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_ID1" (xd:postBackModel="POST_BACK_MODEL_VALUE")?
xd:widgetIndex="TABINDEX1">
    <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:variable name=(&quot;filterParentHasNewRows(CONTROL_ID1)&quot;)
select=&quot;xdXDocument:GetNamedNodeProperty(RELATIVE_GROUP_XPATH1,&quot;filterHasNewRows&qu
ot;, &quot;false&quot;)&quot;/>
    <xsl:variable name=(&quot;filterParentVersion(CONTROL_ID1)&quot;)
select=&quot;xdXDocument:GetNamedNodeProperty(RELATIVE_GROUP_XPATH1,&quot;parentFilterVersion
&quot;,&quot;0&quot;)&quot;/>)
<xsl:for-each select="GROUP_XPATH([PREDICATE_XPATH or ($filterParentHasNewRows(CONTROL_ID1)=
"true" and xdXDocument:GetNamedNodeProperty(., "filterVersion", "0") >
$filterParentVersion(CONTROL_ID1))]) ">
            REPEATING_TABLE_ROWS_WITH_CONDITIONALVISIBILITY |
TABLE_ROW_WITH_CONDITIONAL_FORMATTING*
</xsl:for-each>
    CHECK_FOR_GETDOM_END1
    </tbody>
    (<tbody (valign="VALIGN")? (style="STYLE_DIRECTION")? class="xdTableFooter">
        TABLE_ROW*
    </tbody>)?
</table>
(<div class="optionalPlaceholder" xd:xmlToEdit="XML_TO_EDIT_NAME" tabIndex="TAB_INDEX1"
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_CONDITIONALVISIBILITY*
</xsl:if>
```

TABLE_ROW_WITH_CONDITIONAL_FORMATTING:

```
<tr>
    (TABLE_ROW_CONDITIONAL_FORMATTING_EXPR)?
    (TABLE_CELL)+
</tr>
```

TABLE_ROW_WITH_CONDITIONALVISIBILITY:

```
<tr HIDDEN_FORMATTING_CAPTION>
    (TABLE_ROW_CONDITIONAL_FORMATTING_EXPR)?
    (TABLE_CELL)+
</tr>
```

TABLE_ROW_CONDITIONAL_FORMATTING_EXPR:

```
<xsl:attribute name="style">
    MIN_HEIGHT?
    <xsl:choose>
       (<xsl:when
test="BOOLEAN_XPATH_EXPRESSIONx">CONTAINER_CONDITIONAL_FORMATTING</xsl:when>)+
    </xsl:choose>
    (<xsl:if test="BOOLEAN_XPATH_EXPRESSION">STYLE_DISABLE_CHILD_XML_TO_EDIT</xsl:if>)*
</xsl:attribute>
```

CONTAINER_CONDITIONAL_FORMATTING: Semicolon-delimited list of (**STYLE_BACKGROUND_COLOR**?, **STYLE_CAPTION**).

The following table lists control-specific attributes used by the repeating table control.

Attribute	Section
Xd:action	
xd:CtrlId	
xd:postBackModel	
xd:widgetIndex	
xd:xctname	
xd:xmlToEdit	

2.4.1.17 Rich Text Box Control

The rich text box control allows the user to enter rich text, sch as formatted text, tables, hyperlinks, and images, in the form (1).

Symbol	Description
RICH_TEXT_BOX_PLAIN	The rich text box.

Symbol	Description
RICH_TEXT_BOX_WITH_PLACEHOLDER_TEXT	The rich text box shows place holder text as long as the field (3) it is bound to contains no data. Important: Place holder text is not supported on the form server.

RICH_TEXT_BOX_PLAIN:

```
<span class="xdRichTextBox(RTL)?" hideFocus="1" title="ANY_STRING" xd:binding="LEAF_XPATH"
(tabIndex="TAB_INDEX")? xd:xctname="RichText" xd:CtrlId="CONTROL_ID"
(style="RICH_TEXT_BOX_STYLE")? (accessKey="SINGLE_CHARACTER")?
(xd:postbackModel="POSTBACKMODEL")? (INPUT_SCOPE)? (contentEditable="true" |
xd:disableEditing="(yes|no)")>
    CHECK_FOR_GETDOM_BEGIN1
    RICH_TEXT_BOX_CONDITIONAL_FORMATTING?
    <xsl:copy-of select="LEAF_XPATH/node()"/>
    CHECK_FOR_GETDOM_END1
</span>
```

RICH_TEXT_BOX_WITH_PLACEHOLDER_TEXT:

```
<span class="xdRichTextBox(RTL)? xdBehavior_GhostedText" hideFocus="1" title="ANY_STRING"
xd:binding="LEAF_XPATH" (tabIndex="TAB_INDEX")? xd:xctname="RichText" xd:CtrlId="CONTROL_ID"
(style="RICH_TEXT_BOX_STYLE")? (accessKey="SINGLE_CHARACTER")?
(xd:postbackModel="POSTBACKMODEL")? (INPUT_SCOPE)? (contentEditable="true" |
xd:disableEditing=("yes"|"no"))>
    CHECK_FOR_GETDOM_BEGIN1
    RICH_TEXT_BOX_CONDITIONAL_FORMATTING?
    (<xsl:choose>
        <xsl:when test="not(string(LEAF_XPATH) or LEAF_XPATH/node())">
            <xsl:attribute name="xd:ghosted">true</xsl:attribute>ANY_STRING</xsl:when>
        <xsl:otherwise>
            <xsl:copy-of select="LEAF_XPATH/node()"/>
        </xsl:otherwise>
    </xsl:choose>) | (<xsl:copy-of select="LEAF_XPATH/node()"/>)
    CHECK_FOR_GETDOM_END1
</span>
```

RICH_TEXT_BOX_SCROLLING_STYLE: OVERFLOW-X: visible or OVERFLOW-Y: scroll; OVERFLOW-X: scroll or OVERFLOW-Y: auto; OVERFLOW-X: auto or OVERFLOW-Y: auto or 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**?, **STYLE_CAPTION**).

RICH_TEXT_BOX_CONDITIONAL_FORMATTING:

```
(<xsl:attribute name="style">RICH_TEXT_BOX_STYLE<xsl:choose>
```

```
        (<xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_DISPLAY_NONE;
STYLE_CAPTION</xsl:when>|
        <xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_CAPTION</xsl:when>|
        <xsl:when test="BOOLEAN_XPATH_EXPRESSION">RICH_TEXT_BOX_CONDITIONAL_FORMATTING_STYLE
        </xsl:when>)+
    </xsl:choose>
</xsl:attribute>)?
(<xsl:choose>
    (<xsl:when test="BOOLEAN_XPATH_EXPRESSION"/>|
    <xsl:when test="BOOLEAN_XPATH_EXPRESSION">
        <xsl:attribute name="contentEditable">false</xsl:attribute>
    </xsl:when>)+
</xsl:choose>)?
```

The following table lists control-specific attributes used by the rich text box control.

Attribute	Section
xd:allowNonMatching	2.4.2.2
xd:binding	2.4.2.6
xd:CtrlId	2.4.2.10
xd:disableEditing	2.4.2.12
xd:ghosted	2.4.2.15
xd:inputScopeId	2.4.2.21
xd:postbackModel	2.4.2.29
xd:xctname	2.4.2.35

2.4.1.18 Section Control and Optional Section Control

A section control acts as a container for other controls. An optional section control has the same functionality as a regular section, but it can also be deleted or inserted by the user. The following table describes the symbols for a section control and optional section control.

Symbol	Description
SIMPLE_SECTION_CALL	The first part of the section that indicates where the section control is rendered.
OPTIONAL_SECTION_CALL	The first part of the optional section that indicates where the optional section control is rendered. This part can be configured to not allow the

Symbol	Description
	user to delete or insert the optional section.
SIMPLE_SECTION_BODY	The second part of the section or optional section that defines the properties of the section and its content.
SIMPLE_SECTION_BODY_WITH_DIGITAL_SIGNATURE	The second part of the section that defines the properties of the section and its content. It also contains a digital signature (1) component that enables the user to sign it.
SIMPLE_SECTION_BODY_WITH_CONDITIONAL_HIDING	The second part of the section or optional section that defines the properties of the section and its content. This part is used when the section is conditionally hidden.
SIMPLE_SECTION_BODY_WITH_CONDITIONAL_HIDING_AND_DIGITAL_SIGNATURE	The second part of the section or optional section that defines the properties of the section and its content. This part is used when the section is conditionally hidden. It also contains a digital signature (1) component that enables the user to sign it.

SECTION MUST consist of a **SIMPLE_SECTION_CALL** at the point in the XSL where the control appears, which is either the body of the main template or another XSL template, and a **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, which is either the body of the main template or another XSL template, and a **SECTION_BODY** that defines the section and appears as a separate XSL template. **TEMPLATE_MODE_ID** values in **OPTIONAL_SECTION_CALL** and **SECTION_BODY** MUST match.

In any production defined in this section, any use of a **TAB_INDEX** with the same subscript MUST have an identical yield.

SIMPLE_SECTION_CALL:

```
CHECK_FOR_GETDOM_BEGIN1
<xsl:apply-templates select="(GROUP_XPATH/)?RELATIVE_GROUP_XPATH" mode="TEMPLATE_MODE_ID1"/>
CHECK_FOR_GETDOM_END1
```

OPTIONAL_SECTION_CALL: SIMPLE_SECTION_CALL or

```
(CHECK_FOR_GETDOM_BEGIN1
<xsl:choose>
    <xsl:when test="OPTIONAL_SECTION_XPATH1">
        <xsl:apply-templates select="OPTIONAL_SECTION_XPATH1" mode="TEMPLATE_MODE_ID1"/>
    </xsl:when>
    <xsl:otherwise>
        <div class="optionalPlaceholder" xd:xmlToEdit="XML_TO_EDIT_NAME" tabIndex="TABINDEX1"
align="ALIGN" style="STYLE_WIDTH">ANY_STRING</div>
    </xsl:otherwise>
</xsl:choose>
CHECK_FOR_GETDOM_END1)
```

OPTIONAL_SECTION_XPATH: (**GROUP_XPATH**/)?**RELATIVE_GROUP_XPATH.**

SECTION_STYLE: Semicolon-delimited list of (**STYLE_SIZE**?, **STYLE_DIRECTION**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_BORDER**?, **STYLE_MARGIN**?, **STYLE_PADDING**?).

DIGITAL_SIGNATURE_STYLE: Semicolon-delimited list of (**STYLE_MARGIN**?, **BEHAVIOR**: url (#default#SignaturesInDocUI), **STYLE_WIDTH**?).

SECTION_BODY: SIMPLE_SECTION_BODY or
SIMPLE_SECTION_BODY_WITH_DIGITAL_SIGNATURE or
SIMPLE_SECTION_BODY_WITH_CONDITIONAL_HIDING or
SIMPLE_SECTION_BODY_WITH_CONDITIONAL_HIDING_AND_DIGITAL_SIGNATURE.

SIMPLE_SECTION_BODY:

```
<xsl:template match="RELATIVE_GROUP_XPATH" mode="TEMPLATE_MODE_ID1">
    <div class="xdSection xdRepeating" title="ANY_STRING" (style="SECTION_STYLE")?
align="ALIGN" xd:xctname="Section" xd:CtrlId="CONTROL_ID" (tabIndex="-1")?
xd:widgetIndex="TABINDEX1" (xd:postbackModel="POSTBACKMODEL")?>
        XML_HTML_4_1_WITH_CONTROLS
        (<xsl:attribute name="style">SECTION_STYLE(<xsl:choose>
              (<xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_CAPTION</xsl:when>|

<xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_BACKGROUND_COLOR; STYLE_CAPTION</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:widgetIndex="TABINDEX1" (xd:postbackModel="POSTBACKMODEL")?>
        XML_HTML_4_1_WITH_CONTROLS
        (<xsl:attribute name="style">SECTION_STYLE(<xsl:choose>
                (<xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_CAPTION</xsl:when>|
                <xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_BACKGROUND_COLOR;
STYLE_CAPTION</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:widgetIndex="TABINDEX1" (xd:postbackModel="POSTBACKMODEL")? HIDDEN_FORMATTING_CAPTION>
            XML_HTML_4_1_WITH_CONTROLS
            (<xsl:attribute name="style">SECTION_STYLE(<xsl:choose>
                    (<xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_CAPTION</xsl:when>|
                    <xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_BACKGROUND_COLOR;
STYLE_CAPTION</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:widgetIndex="TABINDEX1" (xd:postbackModel="POSTBACKMODEL")?
HIDDEN_FORMATTING_CAPTION>
```

```
            XML_HTML_4_1_WITH_CONTROLS
            (<xsl:attribute name="style">SECTION_STYLE(<xsl:choose>
                    (<xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_CAPTION</xsl:when>|
                    <xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_BACKGROUND_COLOR;
STYLE_CAPTION</xsl:when>)+
                </xsl:choose>)?
                (<xsl:if
test="BOOLEAN_XPATH_EXPRESSION">STYLE_DISABLE_CHILD_XML_TO_EDIT</xsl:if>)*
            </xsl:attribute>)?
        </div>
        (SECTION_DIGITAL_SIGNATURE_BLOCK |
SECTION_DIGITAL_SIGNATURE_BLOCK_NOT_SHOWING_SIGNATURES)
    </xsl:if>
</xsl:template>
```

SECTION_DIGITAL_SIGNATURE_BLOCK_VALID_SIGNATURE_BUTTON:

```
<button title="" style="width: 100%; height: 100%; text-align: left; border: 0px solid;
padding: 2px; background-color: window; cursor: hand;">
    <table style="color: windowtext;" class="defaultInDocUI">
        <tbody>
            <tr>
                <xsl:choose>
                    <xsl:when test="function-available('xdImage:getImageUrl') and
sig:Object/sig:SignatureProperties/sig:SignatureProperty/xdSignatureProperties:NonRepudiation
/xdSignatureProperties:ValidSignedImage">
                        <td style="display: none;"><img
src="res://infopath.exe/signaturevalid.gif"/></td>
                        <td><img
src="{xdImage:getImageUrl(sig:Object/sig:SignatureProperties/sig:SignatureProperty/xdSignatur
eProperties:NonRepudiation/xdSignatureProperties:ValidSignedImage)}"/> </td>
                    </xsl:when>
                    <xsl:otherwise>
                        <td><img src="res://infopath.exe/signaturevalid.gif" width="12"
height="12"/></td>
                        <td style="color: gray;">
                            <div><b><xsl:value-of select="xdXDocument:GetNamedNodeProperty(.,
'SignedBy', '???')"/></b><span style="margin: 0pt 20pt">ANY_STRING</span></div>
                            <div><xsl:value-of select="xdXDocument:GetNamedNodeProperty(.,
'SignedOn', '???')"/></div>
                        </td>
                    </xsl:otherwise>
                </xsl:choose>
            </tr>
        </tbody>
    </table>
</button>
```

SECTION_DIGITAL_SIGNATURE_BLOCK_INVALID_SIGNATURE_BUTTON:

```
<button title="" style="width: 100%; height: 100%; text-align: left; border: 0px solid;
padding: 2px; background-color: window; cursor: hand;">
    <table style="font: message-box; color: windowtext;">
        <tbody>
            <tr>
                <xsl:choose>
```

```
                    <xsl:when test="function-available('xdImage:getImageUrl') and
sig:Object/sig:SignatureProperties/sig:SignatureProperty/xdSignatureProperties:NonRepudiation
/xdSignatureProperties:InvalidSignedImage">
                        <td style="display: none;"><img
src="res://infopath.exe/signaturevalid.gif"/></td>
                        <td><img
src="{xdImage:getImageUrl(sig:Object/sig:SignatureProperties/sig:SignatureProperty/xdSignatur
eProperties:NonRepudiation/xdSignatureProperties:InvalidSignedImage)}"/> </td>
                    </xsl:when>
                    <xsl:otherwise>
                        <td><img src="res://infopath.exe/signatureerror.gif" width="12"
height="12"/></td>
                        <td style="color: red;"><b>ANY_STRING</b><span style="margin: 0pt
20pt">ANY_STRING</span></td>
                    </xsl:otherwise>
                </xsl:choose>
            </tr>
        </tbody>
    </table>
</button>
```

SECTION_DIGITAL_SIGNATURE_BLOCK:

```
<div xd:disableEditing="yes" xd:SignatureBlock="ANY_STRING1"
xd:SignedSectionDisplaySignatures="true" style="DIGITAL_SIGNATURE_STYLE">
    <xsl:if test="function-available('xdXDocument:GetNamedNodeProperty')">
        <xsl:if test="xdXDocument:GetNamedNodeProperty(DIGITAL_SIGNATURE_XPATH,
'CanAddSignature', 'false') = 'true'">
            <button title="" style="width: 100%; height: 100%; text-align: left; border: 0px
solid; padding: 2px; background-color: window; cursor: hand;">
                <table style="color: windowtext;" class="defaultInDocUI">
                    <tbody>
                        <tr>
                            <td><img src="res://infopath.exe/AddSignatureButton.gif"
width="12" height="12"/></td>
                            <td>ANY_STRING</td>
                        </tr>
                    </tbody>
                </table>
            </button>
        </xsl:if>
        <xsl:for-each select="DIGITAL_SIGNATURE_XPATH">
            <xsl:for-each select="sig:Signature">
                <xsl:choose>
                    <xsl:when test="xdXDocument:GetNamedNodeProperty(., 'IsValidSignature',
'false') = 'true'">
                        SECTION_DIGITAL_SIGNATURE_BLOCK_VALID_SIGNATURE_BUTTON
                    </xsl:when>
                    <xsl:otherwise>
                        SECTION_DIGITAL_SIGNATURE_BLOCK_INVALID_SIGNATURE_BUTTON
                    </xsl:otherwise>
                </xsl:choose>
            </xsl:for-each>
        </xsl:for-each>
    </xsl:if>
</div>
```

SECTION_DIGITAL_SIGNATURE_BLOCK_NOT_SHOWING_SIGNATURES:

```
<div xd:disableEditing="yes" xd:SignatureBlock="ANY_STRING1" style="DIGITAL_SIGNATURE_STYLE">
    <xsl:if test="function-available('xdXDocument:GetNamedNodeProperty')">
        <xsl:if test="xdXDocument:GetNamedNodeProperty(DIGITAL_SIGNATURE_XPATH,
'CanAddSignature', 'false') = 'true'">
            <button title="" style="width: 100%; height: 100%; text-align: left; border: 0px
solid; padding: 2px; background-color: window; cursor: hand;">
                <table style="color: windowtext;" class="defaultInDocUI">
                    <tbody>
                        <tr>
                            <td><img src="res://infopath.exe/AddSignatureButton.gif"
width="12" height="12"/></td>
                            <td>ANY_STRING</td>
                        </tr>
                    </tbody>
                </table>
            </button>
        </xsl:if>
    </xsl:if>
</div>
```

The following table lists control-specific attributes used by the section control.

Attribute	Section
xd:CtrlId	2.4.2.10
xd:disableEditing	2.4.2.12
xd:postbackMode	2.4.2.29
xd:SignatureBlock	2.4.2.31
xd:SignedSectionDisplaySignatures	2.4.2.32
xd:SignedSectionName	2.4.2.33
xd:widgetIndex	2.4.2.37.10
xd:xctname	2.4.2.35
xd:xmlToEdit	2.4.2.36

The following table lists XSL function extensions used by the section control.

Function	Section
xdImage:getImageUr	2.4.3.5
xdXDocument:GetNamedNodeProperty	2.4.3.9.3

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. The following table describes the symbols for a table control.

Symbol	Description
TABLE	Table maps to a standard html layout table, as specified in [HTML] section 11.2.1.
TABLE_COLUMN	Maps to the HTML **col** element, as specified in [HTML] section 11.2.4.
TABLE_ROW	Maps to the HTML **tr** element, as specified in [HTML] section 11.2.5.
TABLE_CELL	Maps to the HTML **td** element, as specified in [HTML] section 11.2.6.

TABLE:

```
<table class="TABLE_STYLE_CLASS" 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="TABLE_ROW_STYLE_CLASS")? (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 (class="TABLE_CELL_STYLE_CLASS")? (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**?).

TABLE_STYLE_CLASS: xdFormLayout (**xdLayout** or **xdTableStyleOneCol** or **xdTableStyleTwoCol** or **xdTableStyleFourCol**).

TABLE_ROW_STYLE_CLASS: xdHeadingRow or **xdSubHeadingRow** or **xdTableLabelControlStackedRow** or **xdTableStyleTitleRow** or **xdTitleRow.**

TABLE_CELL_STYLE_CLASS: xdTitleCell or **xdTitleCellWithHeading** or **xdHeadingCell** or **xdSubheadingCell** or **xdBodyTextCell** or **xdTableStyleTitleCell** or **xdTableLabel** or **xdTableComponent** or **xdVerticalFill** or **xdTableStyleTitleCellVerticalOffset.**

The xd:**layoutText** attribute, as specified in section 2.4.2.22, is the only control-specific attribute used by the table control.

2.4.1.20 Text Box Control

The text box control allows the user to enter simple text in the form (1). The following table describes the symbols for the text box control.

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 of input.
SIMPLE_TEXT_BOX_WITH_DATA_FORMATTING	The text box with data formatting.

Symbol	Description
TEXT_BOX_WITH_CONDITIONAL_FORMATTING_AND_DATA_FORMATTING	The text box with data formatting and conditional formatting.
SIMPLE_TEXT_BOX_WITH_PLACEHOLDER_TEXT	The text box with placeholder text. The placeholder text is ignored by the form server.
SIMPLE_TEXT_BOX_WITH_PLACEHOLER_TEXT_AND_DATA_FORMATTING	The text box with placeholder text and data formatting. The placeholder text is ignored by the form server.
TEXT_BOX_MULTI_LINE_PLACEHOLDER_TEXT	The text box that allows multiple lines input and has placeholder text. The placeholder text is ignored by the form server.
TEXT_BOX_MULTI_LINE_CONDITIONAL_FORMATTING_AND_PLACEHOLDER_TEXT	The text box that allows multiple lines input and has placeholder text with conditional formatting. The

Symbol	Description
	placeholder text is ignored by the form server.
TEXT_BOX_WITH_CONDITIONAL_FORMATTING_AND_PLACEHOLDER_TEXT	The text box that has placeholder text with conditional formatting. The placeholder text is ignored by the form server.
TEXT_BOX_WITH_CONDITIONAL_FORMATTING_AND_PLACEHOLDER_TEXT_AND_DATA_FORMATTING	The text box that has placeholder text with conditional formatting and data formatting. The placeholder text is ignored by the form server.

TEXT_BOX_EDITING: contentEditable="true" or **xd:disableEditing**="(yes|no)" or (**contentEditable**="true" **xd:disableEditing**="yes".

TEXT_BOX_AUTOADVANCE: xd:**autoAdvance**="yes".

WHITESPACE_NO_WRAP: WHITE-SPACE: nowrap.

TEXT_BOX_OUTPUT_ESC: **disable-output-escaping**="yes".

TEXT_BOX_STYLE: Semicolon-delimited list of (**STYLE_SIZE**?, **STYLE_MARGIN**?, **STYLE_PADDING**?, **STYLE_TEXT_DECORATION**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_BORDER**?, **STYLE_FONT**?, **STYLE_COLOR**?, **WHITESPACE_NO_WRAP**?, **STYLE_WIDTH**?, **STYLE_WRAP**?, **STYLE_TEXT_ALIGN**?, (**OVERFLOW-Y**: auto; **OVERFLOW-X**: auto;)?, **STYLE_VERTICAL_ALIGN**?, **STYLE_DIRECTION**?).

TEXT_BOX_STYLE_CONDITIONAL_FORMATTING: Semicolon-delimited list of (**STYLE_TEXT_DECORATION**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_FONT**?, **STYLE_COLOR**?, **STYLE_TEXT_ALIGN**?, **STYLE_CAPTION**).

TEXT_BOX_BASE_CLASS_NAME: xdTextBox or **xdTextBoxRTL.**

SIMPLE_TEXT_BOX:

```
<span class="TEXT_BOX_BASE_CLASS_NAME" hideFocus="1" title="ANY_STRING"
xd:binding="LEAF_XPATH1"
    tabIndex="TAB_INDEX"  xd:xctname="PlainText" xd:CtrlId="CONTROL_ID"
    (TEXT_BOX_EDITING)? style="TEXT_BOX_STYLE" (TEXT_BOX_AUTOADVANCE)?
    (xd:postbackModel="POSTBACKMODEL")? (accessKey="SINGLE_CHARACTER")? (INPUT_SCOPE)?>
    CHECK_FOR_GETDOM_BEGIN1
    <xsl:value-of select="LEAF_XPATH1" TEXT_BOX_OUTPUT_ESC? />
    CHECK_FOR_GETDOM_END1
</span>
```

TEXT_BOX_WITH_CONDITIONAL_FORMATTING:

```
<span class="TEXT_BOX_BASE_CLASS_NAME" hideFocus="1" title="ANY_STRING"
xd:binding="LEAF_XPATH1" (style="TEXT_BOX_STYLE")?
    tabIndex="TAB_INDEX" xd:xctname="PlainText" xd:CtrlId="CONTROL_ID"
    (TEXT_BOX_EDITING)? (TEXT_BOX_AUTOADVANCE)?
    (xd:postbackModel="POSTBACKMODEL")? (accessKey="SINGLE_CHARACTER")? (INPUT_SCOPE)?>
        CHECK_FOR_GETDOM_BEGIN1
        TEXT_BOX_CONDITIONAL_FORMATTING
        <xsl:value-of select="LEAF_XPATH1" TEXT_BOX_OUTPUT_ESC? />
        CHECK_FOR_GETDOM_END1
</span>
```

TEXT_BOX_CONDITIONAL_FORMATTING: TEXT_BOX_CONDITIONAL_FORMATTING_ATT (TEXT_BOX_CONDITIONAL_FORMATTING_CHOOSE)? or **TEXT_BOX_CONDITIONAL_FORMATTING_CHOOSE.**

TEXT_BOX_CONDITIONAL_FORMATTING_ATT:

```
<xsl:attribute name="style">TEXT_BOX_STYLE<xsl:choose>
        (<xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_DISPLAY_NONE;
STYLE_CAPTION</xsl:when>|
        <xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_CAPTION</xsl:when>|
        <xsl:when test="BOOLEAN_XPATH_EXPRESSION">TEXT_BOX_STYLE_CONDITIONAL_FORMATTING
        </xsl:when>)+
    </xsl:choose>
</xsl:attribute>
```

TEXT_BOX_CONDITIONAL_FORMATTING_CHOOSE:

```
<xsl:choose>
    (<xsl:when test="BOOLEAN_XPATH_EXPRESSION"/>|
    <xsl:when test="BOOLEAN_XPATH_EXPRESSION">
        <xsl:attribute name="contentEditable">false</xsl:attribute>
    </xsl:when>)+
</xsl:choose>
```

SIMPLE_TEXT_BOX_MULTI_LINE:

```
<span class="TEXT_BOX_BASE_CLASS_NAME" hideFocus="1" title="ANY_STRING"
    xd:binding="LEAF_XPATH1" tabIndex="TAB_INDEX" (TEXT_BOX_AUTOADVANCE)?
    xd:datafmt="DATA_FMT_CAT_STRING" xd:xctname="PlainText" xd:CtrlId="CONTROL_ID"
    (TEXT_BOX_EDITING)? TEXT_BOX_STYLE
    (xd:postbackModel="POSTBACKMODEL")? (accessKey="SINGLE_CHARACTER")? (INPUT_SCOPE)?>
    CHECK_FOR_GETDOM_BEGIN1
    <xsl:choose>
        <xsl:when test="function-available('xdFormatting:formatString')">
            <xsl:value-of select="xdFormatting:formatString(LEAF_XPATH1,
DATA_FMT_CAT_STRING)" TEXT_BOX_OUTPUT_ESC />
        </xsl:when>
        <xsl:otherwise>
            <xsl:value-of select="LEAF_XPATH1" TEXT_BOX_OUTPUT_ESC />
        </xsl:otherwise>
    </xsl:choose>
    CHECK_FOR_GETDOM_END1
</span>
```

TEXT_BOX_MULTI_LINE_WITH_CONDITIONAL_FORMATTING:

```
<span class="TEXT_BOX_BASE_CLASS_NAME" hideFocus="1" title="ANY_STRING"
    xd:binding="LEAF_XPATH1"  tabIndex="TAB_INDEX" (style="TEXT_BOX_STYLE")?
    xd:datafmt="DATA_FMT_CAT_STRING" xd:xctname="PlainText"
    xd:CtrlId="CONTROL_ID" (TEXT_BOX_EDITING)? (TEXT_BOX_AUTOADVANCE)?
    (xd:postbackModel="POSTBACKMODEL")? (accessKey="SINGLE_CHARACTER")? (INPUT_SCOPE)?>
    CHECK_FOR_GETDOM_BEGIN1
    TEXT_BOX_CONDITIONAL_FORMATTING
    <xsl:choose>
        <xsl:when test="function-available('xdFormatting:formatString')">
            <xsl:value-of select="xdFormatting:formatString(LEAF_XPATH1,
DATA_FMT_CAT_STRING)" TEXT_BOX_OUTPUT_ESC />
        </xsl:when>
        <xsl:otherwise>
            <xsl:value-of select="LEAF_XPATH1" TEXT_BOX_OUTPUT_ESC />
        </xsl:otherwise>
    </xsl:choose>
    CHECK_FOR_GETDOM_END1
</span>
```

DATA_FMT_TEXT_BOX_VAL: xd:datafmt="DATA_FMT_CTRL_TEXTBOX$_1$" DATA_FMT2_ATTRIBUTE_TEXT_BOX$_1$.

DATA_FMT_XSL_BASE_TEXTBOX:

```
DATA_FMT2_FUNCTION_TEXT_BOX1
<xsl:when test="function-available('xdFormatting:formatString')">
   <xsl:value-of select="xdFormatting:formatString(LEAF_XPATH1, DATA_FMT_CTRL_TEXTBOX1)"/>
</xsl:when>
```

DATA_FMT_XSL_NUM:

```
<xsl:attribute name="xd:num">
    <xsl:value-of select="LEAF_XPATH1"/>
```

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

DATA_FMT_XSL_TEXTBOX:

```
DATA_FMT_XSL_NUM
<xsl:choose>
    DATA_FMT_XSL_BASE_TEXTBOX
    <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_TEXTBOX
    CHECK_FOR_GETDOM_END1
</span>
```

TEXT_BOX_WITH_CONDITIONAL_FORMATTING_AND_DATA_FORMATTING:

```
    <span class="TEXT_BOX_BASE_CLASS_NAME xdBehavior_Formatting" hideFocus="1"
    title="ANY_STRING" xd:binding="LEAF_XPATH1" xd:boundProp="xd:num"
    tabIndex="TAB_INDEX" DATA_FMT_TEXT_BOX_VAL xd:xctname="PlainText" xd:CtrlId="CONTROL_ID"
    (TEXT_BOX_EDITING)? (TEXT_BOX_AUTOADVANCE)? (style="TEXT_BOX_STYLE")?
    (xd:postbackModel="POSTBACKMODEL")? (accessKey="SINGLE_CHARACTER")? (INPUT_SCOPE)?>
    CHECK_FOR_GETDOM_BEGIN1
    TEXT_BOX_CONDITIONAL_FORMATTING
    DATA_FMT_XSL_TEXTBOX
    CHECK_FOR_GETDOM_END1
</span>
```

PLACEHOLDER_TEXT_XSL_BASE:

```
<xsl:when test="not(string(LEAF_XPATH1))">
    <xsl:attribute name="xd:ghosted">true</xsl:attribute>ANY_STRING
</xsl:when>
```

PLACEHOLDER_TEXT_XSL:

```
<xsl:choose>
    PLACEHOLDER_TEXT_XSL_BASE
    <xsl:otherwise>
        <xsl:value-of select="LEAF_XPATH1" TEXT_BOX_OUTPUT_ESC? />
    </xsl:otherwise>
```

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

SIMPLE_TEXT_BOX_WITH_PLACEHOLDER_TEXT:

```
<span class="TEXT_BOX_BASE_CLASS_NAME xdBehavior_GhostedText" hideFocus="1"
title="ANY_STRING" xd:binding="LEAF_XPATH1" (TEXT_BOX_AUTOADVANCE)?
    tabIndex="TAB_INDEX"  xd:xctname="PlainText" xd:CtrlId="CONTROL_ID"
    (TEXT_BOX_EDITING)? style="TEXT_BOX_STYLE"
    (xd:postbackModel="POSTBACKMODEL")? (accessKey="SINGLE_CHARACTER")? (INPUT_SCOPE)?>
    CHECK_FOR_GETDOM_BEGIN1
    PLACEHOLDER_TEXT_XSL
    CHECK_FOR_GETDOM_END1
</span>
```

SIMPLE_TEXT_BOX_WITH_PLACEHOLER_TEXT_AND_DATA_FORMATTING:

```
<span class="TEXT_BOX_BASE_CLASS_NAME xdBehavior_GTFormatting" hideFocus="1"
    title="ANY_STRING" xd:binding="LEAF_XPATH1" xd:boundProp="xd:num"
    tabIndex="TAB_INDEX" DATA_FMT_TEXT_BOX_VAL xd:xctname="PlainText" xd:CtrlId="CONTROL_ID"
    (TEXT_BOX_EDITING)? style="TEXT_BOX_STYLE" (TEXT_BOX_AUTOADVANCE)?
    (xd:postbackModel="POSTBACKMODEL")? (accessKey="SINGLE_CHARACTER")? (INPUT_SCOPE)?>
    CHECK_FOR_GETDOM_BEGIN1
    DATA_FMT_XSL_NUM
    <xsl:choose>
        PLACEHOLDER_TEXT_XSL_BASE
        DATA_FMT_XSL_BASE_TEXTBOX
        <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_TEXTBOX
        <xsl:otherwise>
            <xsl:value-of select="LEAF_XPATH1"/>
        </xsl:otherwise>
    </xsl:choose>
    CHECK_FOR_GETDOM_END1
</span>
```

DATA_FMT2_ATTRIBUTE_TEXT_BOX: (**xd:datafmt2**="DATA_FMT2$_1$")?

DATA_FMT2_FUNCTION_TEXT_BOX:

```
(<xsl:when test="function-available('xdFormatting:formatString2')">
    <xsl:value-of select="xdFormatting:formatString2(LEAF_XPATH1, DATA_FMT_CTRL_TEXTBOX1,
'DATA_FMT21')" />
</xsl:when>)?
```

DATA_FMT2_ATTRIBUTE_TEXT_BOX and **DATA_FMT2_FUNCTION_TEXT_BOX** symbols always appear in pairs in the EBNF rules. Subscripts are used to mark the pairs.

If the yield of **DATA_FMT2_ATTRIBUTE_TEXT_BOX** in **DATA_FMT_TEXT_BOX_VAL** is empty, the yield of the pairing **DATA_FMT2_FUNCTION_TEXT_BOX** in **DATA_FMT_XSL_BASE_TEXTBOX** MUST be empty.

If the yield of **DATA_FMT2_FUNCTION_TEXT_BOX** in **DATA_FMT_XSL_BASE_TEXTBOX** is empty, the yield of the pairing **DATA_FMT2_ATTRIBUTE_TEXT_BOX** in **DATA_FMT_TEXT_BOX_VAL** MUST be empty.

The following table lists control-specific attributes used by the textbox control.

Attribute	Section
xd:allownonmatching	2.4.2.2
xd:autoAdvance	2.4.2.3
xd:binding	2.4.2.6
xd:boundProp	2.4.2.9
xd:CtrlId	2.4.2.10
xd:datafmt	2.4.2.11
xd:datafmt2	2.4.2.37.4
xd:disableEditing	2.4.2.12
xd:ghosted	2.4.2.15
xd:inputScopeId	2.4.2.21
xd:num	2.4.2.26
xd:postbackModel	2.4.2.29

Attribute	Section
xd:xctname	2.4.2.35

2.4.1.21 View Representation for Controls Introduced in Version 2 of the Structure Specification

2.4.1.21.1 Choice Group Control and Choice Section Control

A choice control allows the user to select from multiple choice sections. Each acts as a container for other controls. The following table describes the symbols for a choice group control or choice section control.

Symbol	Description
CHOICE_GROUP_CALL	The first part of the choice control that indicates where the choice control is rendered.
SIMPLE_CHOICE_SECTION_BODY	One possible representation for the second part of the choice control. Defines the properties of the section and its content.
CHOICE_SECTION_BODY_WITH_CONDITIONAL_FORMATTING	One possible representation for the second part of the choice control. Defines the properties of the section and its content. This part is used when the section is allowed to be conditionally hidden or have a background color applied.

CHOICE_GROUP MUST consist of a **CHOICE_GROUP_CALL** at the point in the XSL where the control appears, which is either the body of the main template or another XSL template, and a **CHOICE_SECTION_BODY** that defines the section and appears as a separate XSL template. **TEMPLATE_MODE_ID** values in **CHOICE_GROUP_CALL** and **CHOICE_SECTION_BODY** MUST match.

CHOICE_GROUP_CALL:

```
<div style="CHOICE_GROUP_STYLE" class="xdSection xdRepeating" xd:xctname="choicegroup"
xd:ref="GROUP_XPATH1">
    CHOICE_GROUP_XHTML
</div>
```

CHOICE_GROUP_STYLE: Semicolon-delimited list of (**STYLE_WIDTH**?, **STYLE_HEIGHT**?, **STYLE_BORDER**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_FONT_WEIGHT**?, **MARGIN-BOTTOM**: 6px)

CHOICE_GROUP_XHTML: 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]. The element inside the fragment MUST NOT contain the **xd:xctname** attribute, and MUST NOT conform to one of the control productions specified for controls in section 2.4.1.5 to section 2.4.1.21. If the fragment

contains an XSL element with the syntax of **CHOICE_SECTION_CALL**, it MUST be located only in the locations where an HTML <**DIV**/> element could also be placed.

CHOICE_SECTION_CALL:

```
CHECK_FOR_GETDOM_BEGIN1
    <xsl:apply-templates select="(GROUP_XPATH1/)?RELATIVE_GROUP_XPATH"
mode="TEMPLATE_MODE_ID1"/>
CHECK_FOR_GETDOM_END1
```

CHOICE_SECTION_BODY: SIMPLE_CHOICE_SECTION_BODY or **CHOICE_SECTION_BODY_WITH_CONDITIONAL_FORMATTING** or **CHOICE_SECTION_BODY_WITH_CONDITIONAL_HIDING.**

SIMPLE_CHOICE_SECTION_BODY:

```
<xsl:template match="RELATIVE_GROUP_XPATH" mode="TEMPLATE_MODE_ID1">
    <div style="CHOICE_SECTION_STYLE" class="xdSection xdRepeating" title="ANY_STRING"
align="ALIGN" xd:xctname="choiceterm" xd:CtrlId="CONTROL_ID" tabIndex="-1"
xd:widgetIndex="TABINDEX" (xd:postbackModel="POSTBACKMODEL)?>
        XML_HTML_4_1_WITH_CONTROLS
</xsl:template>
```

CHOICE_SECTION_BODY_WITH_CONDITIONAL_FORMATTING:

```
<xsl:template match="RELATIVE_GROUP_XPATH" mode="TEMPLATE_MODE_ID1">
    <div style="CHOICE_SECTION_STYLE1" class="xdSection xdRepeating" title="ANY_STRING"
align="ALIGN" xd:xctname="choiceterm" xd:CtrlId="CONTROL_ID" tabIndex="-1"
xd:widgetIndex="TABINDEX" (xd:postbackModel="POSTBACKMODEL)?>
    <xsl:attribute name="style">CHOICE_SECTION_STYLE1
        <xsl:choose>
            (<xsl:when
test="BOOLEAN_XPATH_EXPRESSIONx">CHOICE_SECTION_CONDITIONAL_FORMATTING</xsl:when>)*
        </xsl:choose>
    </xsl:attribute>
        XML_HTML_4_1_WITH_CONTROLS
</xsl:template>
```

CHOICE_SECTION_BODY_WITH_CONDITIONAL_HIDING:

```
<xsl:template match="RELATIVE_GROUP_XPATH" mode="TEMPLATE_MODE_ID1">
    <xsl:if test="BOOLEAN_XPATH_EXPRESSION">
        <div style="CHOICE_SECTION_STYLE1" class="xdSection xdRepeating" title="ANY_STRING"
align="ALIGN" xd:xctname="choiceterm" xd:CtrlId="CONTROL_ID" tabIndex="-1"
xd:widgetIndex="TABINDEX" (xd:postbackModel="POSTBACKMODEL)? HIDDEN_FORMATTING_CAPTION>
        (<xsl:attribute name="style">CHOICE_SECTION_STYLE1
            <xsl:choose>
                (<xsl:when
test="BOOLEAN_XPATH_EXPRESSIONx">CHOICE_SECTION_CONDITIONAL_FORMATTING</xsl:when>)*
            </xsl:choose>
        </xsl:attribute>)?
            XML_HTML_4_1_WITH_CONTROLS
    </xsl:if>
```

```
</xsl:template>
```

CHOICE_SECTION_CONDITIONAL_FORMATTING: Semicolon-delimited list of (**STYLE_BACKGROUND_COLOR**, **STYLE_CAPTION**).

CHOICE_SECTION_STYLE: Semicolon-delimited list of (**STYLE_WIDTH**?, **STYLE_HEIGHT**?, **STYLE_FONT_WEIGHT**?, **STYLE_BORDER**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_MARGIN**?, **STYLE_PADDING**?)

The following table lists control-specific attributes used by the section control.

Attribute	Section
xd:CtrlId	2.4.2.10
xd:postbackModel	2.4.2.29
xd:ref	2.4.2.30
xd:widgetIndex	2.4.2.37.10
xd:xctname	2.4.2.35

2.4.1.21.2 Combo Box Control

The combo box control enables the user to select or specify a single value from a list of options that can be specified manually by the form template designer or be populated from a data source (2). The following table describes the symbols for a combo box control.

Symbol	Description
SIMPLE_COMBO_BOX	The combo box control enables the user to select or specify a single value from a list of options that can be specified manually by the form template designer or be populated from a data source (2).
COMBO_BOX_WITH_CONDITIONAL_FORMATTING	**SIMPLE_COMBO_BOX**, but allows conditional formatting, such as text formatting and disabling.
COMBO_BOX_WITH_VALUES_FROM_DATA_SOURCE	**SIMPLE_COMBO_BOX**, where the values for the collection are drawn from another location within the form's data source (2).
COMBO_BOX_WITH_VALUES_FROM_DATA_SOURCE_AND_CONDITIONAL_FORMATTING	Combination of **COMBO_BOX_WITH_VALUES_FROM_DATA_SOURCE** and **COMBO_BOX_WITH_CONDITIONAL_FORMATTING.**
COMBO_BOX_WITH_UNIQUE_VALUES_FROM_DATA_SOURCE	**DROPDOWN_LIST_BOX_WITH_VALUES_FROM_DATA_SOURCE**, where each value from the collection of values is unique.
COMBO_BOX	Combination of

Symbol	Description
_WITH_UNIQUE_VALUES_FROM_DATA_SOURCE_AND_CONDITIONAL_FORMATTING	**COMBO_BOX_WITH_UNIQUE_VALUES_FROM_DATA_SOURCE** and **COMBO_BOX_WITH_CONDITIONAL_FORMATTING.**

SIMPLE_COMBO_BOX:

```
<span class="xdComboBox xdBehavior_ComboBox" xd:xctname="combobox" style="COMBO_BOX_STYLE1">
    <xsl:if test="function-available('ipApp:GetMajorVersion') and ipApp:GetMajorVersion()
&gt;= 12">
        <select tabIndex="-1" disabled="true"
style="STYLE_WIDTH1;VISIBILITY:hidden;WIDTH:100%;"/>
        <span xd:xctname="PlainText" hideFocus="1" class="TEXT_BOX_BASE_CLASS_NAME
xdBehavior_ComboBoxTextField" (accessKey="SINGLE_CHARACTER1")? title="ANY_STRING1"
xd:binding="LEAF_XPATH1" (tabIndex="TAB_INDEX1")? (value="ANY_STRING")?
xd:CtrlId="CONTROL_ID1" (xd:postbackModel="POSTBACKMODEL)? (INPUT_SCOPE)?
(xd:datafmt="DATA_FMT_CTRL_COMBOBOX")?>
            <xsl:attribute name="style">STYLE_WIDTH1;POSITION:absolute;WIDTH:0px;WORD-
WRAP:normal</xsl:attribute>
            <xsl:choose>
                (<xsl:when test="BOOLEAN_XPATH_EXPRESSIONx">ANY_STRINGy</xsl:when>)*
                <xsl:otherwise>
                    <xsl:value-of select="LEAF_XPATH1"/>
                </xsl:otherwise>
            </xsl:choose>
        </span>
    </xsl:if>
    <select (accessKey="SINGLE_CHARACTER1")? class="xdComboBox xdBehavior_Select"
title="ANY_STRING1" size="1" xd:binding="LEAF_XPATH1" xd:xctname="dropdown"
(value="ANY_STRING")? xd:CtrlId="CONTROL_ID1" (xd:postbackModel="POSTBACKMODEL)?
(INPUT_SCOPE)? xd:boundProp="value">
        <xsl:choose>
            <xsl:when test="function-available('ipApp:GetMajorVersion') and
ipApp:GetMajorVersion() &gt;= 12">
                <xsl:attribute name="tabIndex">-1</xsl:attribute>
            </xsl:when>
            <xsl:otherwise>
                <xsl:attribute name="tabIndex">TAB_INDEX1</xsl:attribute>
            </xsl:otherwise>
        </xsl:choose>
        <xsl:attribute name="style">STYLE_WIDTH1
            <xsl:choose>
                <xsl:when test="function-available('ipApp:GetMajorVersion') and
ipApp:GetMajorVersion() &gt;= 12">;POSITION:absolute;WIDTH:0px;</xsl:when>
                <xsl:otherwise>COMBO_BOX_STYLE1</xsl:otherwise>
            </xsl:choose>
        </xsl:attribute>
        <xsl:attribute name="value">
            <xsl:value-of select="LEAF_XPATH1"/>
        </xsl:attribute>
        (<option (value="ANY_STRING")?>
            <xsl:if test="BOOLEAN_XPATH_EXPRESSIONx">
                <xsl:attribute name="selected">selected</xsl:attribute>
            </xsl:if>ANY_STRINGy
        </option>)*
    </select>
```

```
</span>
```

COMBO_BOX_WITH_CONDITIONAL_FORMATTING:

```
<span class="xdComboBox xdBehavior_ComboBox" xd:xctname="combobox" style="COMBO_BOX_STYLE1">
    <xsl:if test="function-available('ipApp:GetMajorVersion') and ipApp:GetMajorVersion()
&gt;= 12">
        <select tabIndex="-1" disabled="true"
style="STYLE_WIDTH1;VISIBILITY:hidden;WIDTH:100%;"/>
        <span xd:xctname="PlainText" hideFocus="1" class="xdTextBox
xdBehavior_ComboBoxTextField" (accessKey="SINGLE_CHARACTER1")? title="ANY_STRING1"
xd:binding="LEAF_XPATH1" (tabIndex="TAB_INDEX1")? (value="ANY_STRING")?
xd:CtrlId="CONTROL_ID1" (xd:postbackModel="POSTBACKMODEL")? (INPUT_SCOPE)?
(xd:datafmt="DATA_FMT_CTRL_COMBOBOX")?>
            <xsl:attribute name="style">STYLE_WIDTH1
                <xsl:choose>
                    (<xsl:when
test="BOOLEAN_XPATH_EXPRESSIONx">COMBO_BOX_CONDITIONAL_FORMATTING </xsl:when>)+
                </xsl:choose>
            ;POSITION:absolute;WIDTH:0px;WORD-WRAP:normal</xsl:attribute>
            (<xsl:choose>
                    (<xsl:when test="BOOLEAN_XPATH_EXPRESSIONx">
                        (<xsl:attribute name="contentEditable">false</xsl:attribute>)?
                    </xsl:when>)+
            </xsl:choose>)?
            <xsl:choose>
                (<xsl:when test="BOOLEAN_XPATH_EXPRESSIONx">ANY_STRINGy</xsl:when>)*
                <xsl:otherwise>
                    <xsl:value-of select="LEAF_XPATH1"/>
                </xsl:otherwise>
            </xsl:choose>
        </span>
    </xsl:if>
    <select (accessKey="SINGLE_CHARACTER1")? class="xdComboBox xdBehavior_Select"
title="ANY_STRING1" size="1" xd:binding="LEAF_XPATH1" xd:xctname="dropdown"
(value="ANY_STRING")? xd:CtrlId="CONTROL_ID1" (xd:postbackModel="POSTBACKMODEL")?
(INPUT_SCOPE)? xd:boundProp="value">
        <xsl:choose>
            <xsl:when test="function-available('ipApp:GetMajorVersion') and
ipApp:GetMajorVersion() &gt;= 12">
                <xsl:attribute name="tabIndex">-1</xsl:attribute>
            </xsl:when>
            <xsl:otherwise>
                <xsl:attribute name="tabIndex">TAB_INDEX1</xsl:attribute>
            </xsl:otherwise>
        </xsl:choose>
        <xsl:attribute name="style">STYLE_WIDTH1
            <xsl:choose>
                (<xsl:when test="BOOLEAN_XPATH_EXPRESSIONx">COMBO_BOX_CONDITIONAL_FORMATTING
</xsl:when>)+
            </xsl:choose>
            <xsl:choose>
                <xsl:when test="function-available('ipApp:GetMajorVersion') and
ipApp:GetMajorVersion() &gt;= 12">;POSITION:absolute;WIDTH:0px;</xsl:when>
                <xsl:otherwise>COMBO_BOX_STYLE1</xsl:otherwise>
            </xsl:choose>
        </xsl:attribute>
```

```
        <xsl:choose>
            (<xsl:when test="BOOLEAN_XPATH_EXPRESSIONx">
                (<xsl:attribute name="disabled">true</xsl:attribute>)?
            </xsl:when>)+
        </xsl:choose>
        <xsl:attribute name="value">
            <xsl:value-of select="LEAF_XPATH1"/>
        </xsl:attribute>
        (<option (value="ANY_STRING")?>
            <xsl:if test="BOOLEAN_XPATH_EXPRESSIONx
                <xsl:attribute name="selected">selected</xsl:attribute>
            </xsl:if>ANY_STRINGy
        </option>)*
    </select>
</span>
```

COMBO_BOX_WITH_VALUES_FROM_DATA_SOURCE:

```
<span class="xdComboBox xdBehavior_ComboBox" xd:xctname="combobox" style="COMBO_BOX_STYLE1">
    <xsl:if test="function-available('ipApp:GetMajorVersion') and ipApp:GetMajorVersion()
&gt;= 12">
        <select tabIndex="-1" disabled="true"
style="STYLE_WIDTH1;VISIBILITY:hidden;WIDTH:100%;"/>
        <span xd:xctname="PlainText" hideFocus="1" class="xdTextBox
xdBehavior_ComboBoxTextField" (accessKey="SINGLE_CHARACTER1")? title="ANY_STRING1"
xd:binding="LEAF_XPATH1" (tabIndex="TAB_INDEX1")? (value="ANY_STRING")?
xd:CtrlId="CONTROL_ID1" (xd:postbackModel="POSTBACKMODEL")? (INPUT_SCOPE)?
(xd:datafmt="DATA_FMT_CTRL_COMBOBOX")?>
            <xsl:attribute name="style">STYLE_WIDTH1;POSITION:absolute;WIDTH:0px;WORD-
WRAP:normal</xsl:attribute>
                (<xsl:variable name="val" select="LEAF_XPATH1"/>)?
                <xsl:choose>
                    <xsl:when test="REPEATING_LEAF_XPATH1[LEAF_XPATH2 =$val]/.">
                        <xsl:value-of select=" REPEATING_LEAF_XPATH1 [LEAF_XPATH2 =$val]/."/>
                    </xsl:when>
                    <xsl:otherwise>
                        (<xsl:choose>
                            <xsl:when test="function-available('xdFormatting:formatString')">
                                <xsl:value-of
select="xdFormatting:formatString(LEAF_XPATH1,"string","plainMultiline")" />
                            </xsl:when>
                            <xsl:otherwise>
                                <xsl:value-of select="LEAF_XPATH1" />
                            </xsl:otherwise>
                        </xsl:choose>)|
                        (<xsl:value-of select="LEAF_XPATH1"/>)
                    </xsl:otherwise>
                </xsl:choose>
        </span>
    </xsl:if>
    <select (accessKey="SINGLE_CHARACTER1")? class="xdComboBox xdBehavior_Select"
title="ANY_STRING1" size="1" xd:binding="LEAF_XPATH1" xd:xctname="dropdown"
(value="ANY_STRING")? xd:CtrlId="CONTROL_ID1" (xd:postbackModel="POSTBACKMODEL)?
(INPUT_SCOPE)? xd:boundProp="value">
        <xsl:choose>
            <xsl:when test="function-available('ipApp:GetMajorVersion') and
ipApp:GetMajorVersion() &gt;= 12">
```

```
                <xsl:attribute name="tabIndex">-1</xsl:attribute>
            </xsl:when>
            <xsl:otherwise>
                <xsl:attribute name="tabIndex">TAB_INDEX1</xsl:attribute>
            </xsl:otherwise>
        </xsl:choose>
        <xsl:attribute name="style">STYLE_WIDTH1
            <xsl:choose>
                <xsl:when test="function-available('ipApp:GetMajorVersion') and
ipApp:GetMajorVersion() &gt;= 12">;POSITION:absolute;WIDTH:0px;</xsl:when>
                <xsl:otherwise>COMBO_BOX_STYLE1</xsl:otherwise>
            </xsl:choose>
        </xsl:attribute>
        <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_XPATH2=$val] or $val='')">
                    <option selected="selected">
                        <xsl:attribute name="value">
                            <xsl:value-of select="$val"/>
                        </xsl:attribute>
                        <xsl:value-of select="$val"/>
                    </option>
                </xsl:if>
                <xsl:for-each select=" REPEATING_LEAF_XPATH1">
                    <option>
                        <xsl:attribute name="value">
                             <xsl:value-of select="."/>
                         </xsl:attribute>
                        <xsl:if test="$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="LEAF_XPATH1"/>
                </option>
            </xsl:otherwise>
        </xsl:choose>
    </select>
</span>
```

COMBO_BOX_WITH_VALUES_FROM_DATA_SOURCE_AND_CONDITIONAL_FORMATTING

```
<span class="xdComboBox xdBehavior_ComboBox" xd:xctname="combobox" style="COMBO_BOX_STYLE1">
    <xsl:if test="function-available('ipApp:GetMajorVersion') and ipApp:GetMajorVersion()
&gt;= 12">
```

```
        <select tabIndex="-1" disabled="true"
style="STYLE_WIDTH1;VISIBILITY:hidden;WIDTH:100%;"/>
        <span xd:xctname="PlainText" hideFocus="1" class="xdTextBox
xdBehavior_ComboBoxTextField" (accessKey="SINGLE_CHARACTER1")? title="ANY_STRING1"
xd:binding="LEAF_XPATH1" (tabIndex="TAB_INDEX1")? (value="ANY_STRING")?
xd:CtrlId="CONTROL_ID1" (xd:postbackModel="POSTBACKMODEL")? (INPUT_SCOPE)?
(xd:datafmt="DATA_FMT_CTRL_COMBOBOX")?>
            <xsl:attribute name="style">STYLE_WIDTH1
                <xsl:choose>
                    (<xsl:when
test="BOOLEAN_XPATH_EXPRESSIONx">COMBO_BOX_CONDITIONAL_FORMATTING
                    </xsl:when>)+
                </xsl:choose>
            ;POSITION:absolute;WIDTH:0px;WORD-WRAP:normal</xsl:attribute>
            (<xsl:choose>
                    (<xsl:when test="BOOLEAN_XPATH_EXPRESSIONx">
                        (<xsl:attribute name="contentEditable">false</xsl:attribute>)?
                    </xsl:when>)+
            </xsl:choose>)?
                (<xsl:variable name="val" select="LEAF_XPATH1"/>)?
                <xsl:choose>
                    <xsl:when test="REPEATING_LEAF_XPATH1[LEAF_XPATH2 =$val]/.">
                        <xsl:value-of select="REPEATING_LEAF_XPATH1 [LEAF_XPATH2 =$val]/."/>
                    </xsl:when>
                    <xsl:otherwise>
                        (<xsl:choose>
                            <xsl:when test="function-available('xdFormatting:formatString')">
                                <xsl:value-of
select="xdFormatting:formatString(LEAF_XPATH1,"string","plainMultiline")" />
                            </xsl:when>
                            <xsl:otherwise>
                                <xsl:value-of select="LEAF_XPATH1" />
                            </xsl:otherwise>
                        </xsl:choose>)|
                        (<xsl:value-of select="LEAF_XPATH1"/>)
                    </xsl:otherwise>
                </xsl:choose>
        </span>
    </xsl:if>
    <select (accessKey="SINGLE_CHARACTER1")? class="xdComboBox xdBehavior_Select"
title="ANY_STRING1" size="1" xd:binding="LEAF_XPATH1" xd:xctname="dropdown"
(value="ANY_STRING")? xd:CtrlId="CONTROL_ID1" (xd:postbackModel="POSTBACKMODEL)?
xd:boundProp="value" (INPUT_SCOPE)?>
        <xsl:choose>
            <xsl:when test="function-available('ipApp:GetMajorVersion') and
ipApp:GetMajorVersion() &gt;= 12">
                <xsl:attribute name="tabIndex">-1</xsl:attribute>
            </xsl:when>
            <xsl:otherwise>
                <xsl:attribute name="tabIndex">TAB_INDEX1</xsl:attribute>
            </xsl:otherwise>
        </xsl:choose>
        <xsl:attribute name="style">STYLE_WIDTH1
            <xsl:choose>
                (<xsl:when
test="BOOLEAN_XPATH_EXPRESSIONx">COMBO_BOX_CONDITIONAL_FORMATTING</xsl:when>)+
            </xsl:choose>
            <xsl:choose>
```

```
                <xsl:when test="function-available('ipApp:GetMajorVersion') and
ipApp:GetMajorVersion() &gt;= 12">;POSITION:absolute;WIDTH:0px;</xsl:when>
                <xsl:otherwise>COMBO_BOX_STYLE1</xsl:otherwise>
            </xsl:choose>
        </xsl:attribute>
        <xsl:choose>
            (<xsl:when test="BOOLEAN_XPATH_EXPRESSIONx">
                (<xsl:attribute name="disabled">true</xsl:attribute>)?
            </xsl:when>)+
        </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_XPATH2=$val] or $val='')">
                    <option selected="selected">
                        <xsl:attribute name="value">
                            <xsl:value-of select="$val"/>
                        </xsl:attribute>
                        <xsl:value-of select="$val"/>
                    </option>
                </xsl:if>
                <xsl:for-each select=" REPEATING_LEAF_XPATH1">
                    <option>
                        <xsl:attribute name="value">
                             <xsl:value-of select="LEAF_XPATH2"/>
                         </xsl:attribute>
                        <xsl:if test="$val=LEAF_XPATH2">
                            <xsl:attribute name="selected">selected
                            </xsl:attribute>
                        </xsl:if>
                        <xsl:value-of select="LEAF_XPATH2"/>
                    </option>
                </xsl:for-each>
            </xsl:when>
            <xsl:otherwise>
                <option>
                    <xsl:value-of select="LEAF_XPATH1"/>
                </option>
            </xsl:otherwise>
        </xsl:choose>
    </select>
</span>
```

COMBO_BOX_WITH_UNIQUE_VALUES_FROM_DATA_SOURCE:

```
<span class="xdComboBox xdBehavior_ComboBox" xd:xctname="combobox" style="COMBO_BOX_STYLE1">
    <xsl:if test="function-available('ipApp:GetMajorVersion') and ipApp:GetMajorVersion()
&gt;= 12">
        <select tabIndex="-1" disabled="true"
style="STYLE_WIDTH1;VISIBILITY:hidden;WIDTH:100%;"/>
        <span xd:xctname="PlainText" hideFocus="1" class="xdTextBox
xdBehavior_ComboBoxTextField" (accessKey="SINGLE_CHARACTER1")? title="ANY_STRING1"
```

```
xd:binding="LEAF_XPATH1" (tabIndex="TAB_INDEX1")? (value="ANY_STRING")?
xd:CtrlId="CONTROL_ID1" (xd:postbackModel="POSTBACKMODEL)? (INPUT_SCOPE)?
(xd:datafmt="DATA_FMT_CTRL_COMBOBOX")?>
            <xsl:attribute name="style">STYLE_WIDTH1;POSITION:absolute;WIDTH:0px;WORD-
WRAP:normal</xsl:attribute>
                (<xsl:variable name="val" select="LEAF_XPATH1"/>)?
                <xsl:choose>
                    <xsl:when test="REPEATING_LEAF_XPATH1[LEAF_XPATH2 =$val]/.">
                        <xsl:value-of select=" REPEATING_LEAF_XPATH1 [LEAF_XPATH2 =$val]/."/>
                    </xsl:when>
                    <xsl:otherwise>
                        <xsl:value-of select="LEAF_XPATH1"/>
                    </xsl:otherwise>
                </xsl:choose>
        </span>
    </xsl:if>
    <select (accessKey="SINGLE_CHARACTER1")? class="xdComboBox xdBehavior_Select"
title="ANY_STRING1" size="1" xd:binding="LEAF_XPATH1" xd:xctname="dropdown"
(value="ANY_STRING")? xd:CtrlId="CONTROL_ID1" (xd:postbackModel="POSTBACKMODEL)?
(INPUT_SCOPE)? xd:boundProp="value">
        <xsl:choose>
            <xsl:when test="function-available('ipApp:GetMajorVersion') and
ipApp:GetMajorVersion() &gt;= 12">
                <xsl:attribute name="tabIndex">-1</xsl:attribute>
            </xsl:when>
            <xsl:otherwise>
                <xsl:attribute name="tabIndex">TAB_INDEX1</xsl:attribute>
            </xsl:otherwise>
        </xsl:choose>
        <xsl:attribute name="style">STYLE_WIDTH1
            <xsl:choose>
                <xsl:when test="function-available('ipApp:GetMajorVersion') and
ipApp:GetMajorVersion() &gt;= 12">;POSITION:absolute;WIDTH:0px;</xsl:when>
                <xsl:otherwise>COMBO_BOX_STYLE1</xsl:otherwise>
            </xsl:choose>
        </xsl:attribute>
        <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_XPATH2=$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_XPATH1= preceding::REPEATING_LEAF_XPATH1)]"/>
                <xsl:for-each select="$uniqueItems">
                    <option>
                        <xsl:attribute name="value">
```

```
                        <xsl:value-of select="LEAF_XPATH2"/>
                    </xsl:attribute>
                    <xsl:if test="$val=LEAF_XPATH2">
                       <xsl:attribute name="selected">selected
                       </xsl:attribute>
                    </xsl:if>
                    <xsl:value-of select="LEAF_XPATH2"/>
                </option>
            </xsl:for-each>
        </xsl:when>
        <xsl:otherwise>
            <option>
                <xsl:value-of select="LEAF_XPATH1"/>
            </option>
        </xsl:otherwise>
    </xsl:choose>
</select>
</span>
```

COMBO_BOX_WITH_UNIQUE_VALUES_FROM_DATA_SOURCE_AND_CONDITIONAL_FORMATTING:

```
<span class="xdComboBox xdBehavior_ComboBox" xd:xctname="combobox" style="COMBO_BOX_STYLE1">
    <xsl:if test="function-available('ipApp:GetMajorVersion') and ipApp:GetMajorVersion()
&gt;= 12">
        <select tabIndex="-1" disabled="true"
style="STYLE_WIDTH1;VISIBILITY:hidden;WIDTH:100%;"/>
        <span xd:xctname="PlainText" hideFocus="1" class="xdTextBox
xdBehavior_ComboBoxTextField" (accessKey="SINGLE_CHARACTER1")? title="ANY_STRING1"
xd:binding="LEAF_XPATH1" (tabIndex="TAB_INDEX1")? (value="ANY_STRING")?
xd:CtrlId="CONTROL_ID1" (xd:postbackModel="POSTBACKMODEL)? (INPUT_SCOPE)?
(xd:datafmt="DATA_FMT_CTRL_COMBOBOX")?>
                        <xsl:attribute name="style">STYLE_WIDTH1
                <xsl:choose>
                    (<xsl:when
test="BOOLEAN_XPATH_EXPRESSIONx">COMBO_BOX_CONDITIONAL_FORMATTING
                    </xsl:when>)+
                </xsl:choose>
                <xsl:choose>
                    (<xsl:when test="BOOLEAN_XPATH_EXPRESSIONx">
                        (<xsl:attribute name="contentEditable">false</xsl:attribute>)?
                    </xsl:when>)+
                </xsl:choose>
            ;POSITION:absolute;WIDTH:0px;WORD-WRAP:normal</xsl:attribute>
                (<xsl:variable name="val" select="LEAF_XPATH1"/>)?
                <xsl:choose>
                <xsl:when test="REPEATING_LEAF_XPATH1[LEAF_XPATH2 =$val]/.">
                    <xsl:value-of select=" REPEATING_LEAF_XPATH1 [LEAF_XPATH2 =$val]/."/>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:value-of select="LEAF_XPATH1"/>
                </xsl:otherwise>
                </xsl:choose>
        </span>
    </xsl:if>
```

```
    <select (accessKey="SINGLE_CHARACTER1")? class="xdComboBox xdBehavior_Select"
title="ANY_STRING1" size="1" xd:binding="LEAF_XPATH1" xd:xctname="dropdown"
(value="ANY_STRING")? xd:CtrlId="CONTROL_ID1" (xd:postbackModel="POSTBACKMODEL)?
(INPUT_SCOPE)? xd:boundProp="value">
        <xsl:choose>
            <xsl:when test="function-available('ipApp:GetMajorVersion') and
ipApp:GetMajorVersion() &gt;= 12">
                <xsl:attribute name="tabIndex">-1</xsl:attribute>
            </xsl:when>
            <xsl:otherwise>
                <xsl:attribute name="tabIndex">TAB_INDEX1</xsl:attribute>
            </xsl:otherwise>
        </xsl:choose>
        <xsl:attribute name="style">STYLE_WIDTH1
            <xsl:choose>
                (<xsl:when
test="BOOLEAN_XPATH_EXPRESSIONx">COMBO_BOX_CONDITIONAL_FORMATTING</xsl:when>)+
            </xsl:choose>
            <xsl:choose>
                <xsl:when test="function-available('ipApp:GetMajorVersion') and
ipApp:GetMajorVersion() &gt;= 12">;POSITION:absolute;WIDTH:0px;</xsl:when>
                <xsl:otherwise>COMBO_BOX_STYLE1</xsl:otherwise>
            </xsl:choose>
        </xsl:attribute>
        (<xsl:choose>
                (<xsl:when test="BOOLEAN_XPATH_EXPRESSIONx">
                    (<xsl:attribute name="contentEditable">false</xsl:attribute>)?
                </xsl:when>)+
        </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_XPATH2=$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_XPATH1= preceding::REPEATING_LEAF_XPATH1)]"/>
                <xsl:for-each select="$uniqueItems">
                    <option>
                        <xsl:attribute name="value">
                             <xsl:value-of select="LEAF_XPATH2"/>
                         </xsl:attribute>
                        <xsl:if test="$val=LEAF_XPATH2">
                            <xsl:attribute name="selected">selected
                            </xsl:attribute>
                        </xsl:if>
```

```
                        <xsl:value-of select="LEAF_XPATH2"/>
                    </option>
                </xsl:for-each>
            </xsl:when>
            <xsl:otherwise>
                <option>
                    <xsl:value-of select="LEAF_XPATH1"/>
                </option>
            </xsl:otherwise>
        </xsl:choose>
    </select>
</span>
```

DATA_FMT_CTRL_COMBOBOX: DATA_FMT_CAT_TIME or **DATA_FMT_CAT_DATE** or **DATA_FMT_CAT_DATETIME** or **DATA_FMT_CAT_NUMBER** or **DATA_FMT_CAT_PERCENTAGE**.

COMBO_BOX_CONDITIONAL_FORMATTING: LEAF_CONTROL_CONDITIONAL_FORMATTING_CAPTION COMBO_BOX_CONDITIONAL_FORMATTING.

COMBO_BOX_STYLE: Semicolon-delimited list of (**STYLE_WIDTH**?, **STYLE_FONT**?, **STYLE_MARGIN**?, **STYLE_VERTICAL_ALIGN**?, **STYLE_TEXT_DECORATION**?,**STYLE_COLOR**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_DIRECTION**?, layout-grid: none?).

The following table lists control-specific attributes used by the combo box control.

Attribute	Section
xd:allowNonMatching	2.4.2.2
xd:binding	2.4.2.6
xd:boundProp	2.4.2.9
xd:CtrlId	2.4.2.10
xd:datafmt	2.4.2.11
xd:inputscope	2.4.2.20
xd:num	2.4.2.26
xd:postbackModel	2.4.2.29
xd:xctname	2.4.2.35

2.4.1.21.3 Date and Time Picker Control

The date and time picker control enables the user to enter a date and time.

A date and time picker control consists of a date picker control, as specified in section 2.4.1.8, and a text box control, as specified in section 2.4.1.20, bound to the same XML element

2.4.1.21.4 External Item Picker Control

The external item picker control enables the user to select one or more instances of an external content type.

ENTITY_PICKER:

```
<object class="xdActiveX" hideFocus="1" style="ENTITY_PICKER_STYLE" classid="clsid:ad74fc20-
e09f-4e47-8a87-1da49930867a" tabIndex="TAB_INDEX" (height="HEIGHT" width="WIDTH")?
tabStop="true" xd:xctname="entitypicker" xd:CtrlId="CONTROL_ID" xd:server="ANY_STRING"
xd:bindingType="xmlNode" xd:bindingProperty="InfoPathValue" xd:boundProp="xd:inline"
contentEditable="false" xd:binding="GROUP_XPATH1" (title="ANY_STRING")?
(accessKey="SINGLE_CHARACTER")?>
    <xsl:if test="function-available('xdImage:getImageUrl')">
        <xsl:attribute name="src">
            <xsl:value-of select="xdImage:getImageUrl(GROUP_XPATH1)"/>
        </xsl:attribute>
    </xsl:if>
    (<xsl:attribute name="style">ENTITY_PICKER_STYLE?<xsl:choose>
            (<xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_DISPLAY_NONE;
STYLE_CAPTION</xsl:when>|
            <xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_CAPTION</xsl:when>)+
        </xsl:choose>
    </xsl:attribute>)?
    (<xsl:choose>
        (<xsl:when test="BOOLEAN_XPATH_EXPRESSION">
            <xsl:attribute name="xd:disableEditing">yes</xsl:attribute>
        </xsl:when>)+
    </xsl:choose>)?
    <param NAME="ButtonFont" VALUE="ENTITY_PICKER_BUTTON_FONT"/>
    <param NAME="Caption" VALUE="ANY_STRING"/>
    <param NAME="EntityNamespace" VALUE="(ENTITY_PICKER_ENTITY_NAMESPACE)?"/>
    <param NAME="EntityName" VALUE="(ENTITY_PICKER_ENTITY_NAME)?"/>
    <param NAME="EntityFinderName" VALUE="ANY_STRING"/>
    <param NAME="BDCInstanceName" VALUE="ANY_STRING"/>
    <param NAME="DisplayFieldName" VALUE="ENTITY_PICKER_DISPLAY_FIELD"/>
    <param NAME="SystemInstanceName" VALUE="ENTITY_PICKER_SYSTEM_INSTANCE"/>
    <param NAME="EntityBDCServerURL" VALUE="ANY_STRING"/>
    <param NAME="DefaultQuery" VALUE="ANY_STRING"/>
    <param NAME="AssociationName" VALUE="ANY_STRING"/>
    <param NAME="EIRLocation" VALUE="(LEAF_XPATH1)?"/>
    <param NAME="DisplayNameLocation" VALUE="(LEAF_XPATH2)?"/>
    <param NAME="Id1Location" VALUE="(LEAF_XPATH3)?"/>
    <param NAME="Id2Location" VALUE="(LEAF_XPATH4)?"/>
    <param NAME="Id3Location" VALUE="(LEAF_XPATH5)?"/>
    <param NAME="Id4Location" VALUE="(LEAF_XPATH6)?"/>
    <param NAME="Id5Location" VALUE="(LEAF_XPATH7)?"/>
    <param NAME="PickerDialogTitle" VALUE="ANY_STRING"/>
    <param NAME="BackgroundColor" VALUE="ENTITY_PICKER_BACKGROUND_COLOR"/>
    <param NAME="MaxLines" VALUE="ENTITY_PICKER_SELECTOR_MAX_LINES"/>
    <param NAME="Direction" VALUE="ENTITY_PICKER_DIRECTION"/>
    <param NAME="MaxResults" VALUE="ANY_STRING"/>
    <param NAME="EntityButtonWidth" VALUE="ANY_STRING"/>
    <param NAME="QueryRequired" VALUE="ANY_STRING"/>
    <param NAME="RefreshOnOpen" VALUE="ANY_STRING"/>
    <param NAME="PickerTargetMode" VALUE="ANY_STRING"/>
    <param NAME="MultiItem" VALUE="ANY_STRING"/>
```

```
</object>
```

The following table lists parameters used by the external item picker control.

Parameter	Specification
ButtonFont	This parameter specifies the font that is used to render the button text and display names.
Caption	This parameter specifies the text that is displayed on the button that opens the external item picker dialog.
EntityNamespace	This parameter specifies the entity namespace of an external content type.
EntityName	This parameter specifies the name of an external content type.
EntityFinderName	This parameter MUST be ignored.
BDCInstanceName	This parameter MUST be ignored.
DisplayFieldName	This parameter specifies the location, within the instance of an external content type, from where to obtain the suggested value to be used when displaying the instance of an external content type in the external item picker control.
SystemInstanceName	This parameter specifies the name of a LobSystemInstance.
EntityBDCServerURL	This parameter MUST be ignored.
DefaultQuery	This parameter MUST be ignored.
AssociationName	This parameter MUST be ignored.
EIRLocation	This parameter specifies the **LEAF_XPATH** containing the reference to the first instance of the external content type selected in the external item picker control.
DisplayNameLocation	This parameter specifies the **LEAF_XPATH** containing the suggested value to be used when displaying the instance of an external content type in the external item picker control.
Id1Location	This parameter specifies the **LEAF_XPATH** containing the first **Identifier** value for the first instance of the external content type selected in the external item picker control.
Id2Location	This parameter specifies the **LEAF_XPATH** containing the second Identifier value for the first instance of the external content type selected in the external item picker control.
Id3Location	This parameter specifies the **LEAF_XPATH** containing the third Identifier value for the first instance of the external content type selected in the external item picker control.
Id4Location	This parameter specifies the **LEAF_XPATH** containing the fourth Identifier value for the first instance of the external content type selected in the external item picker control.
Id5Location	This parameter specifies the **LEAF_XPATH** containing the fifth Identifier value for the first instance of the external content type selected in the external item picker control.

Parameter	Specification
PickerDialogTitle	This parameter specifies the text that is displayed as the title of the external item picker dialog.
BackgroundColor	This parameter specifies the background color of the external item picker input box.
MaxLines	This parameter specifies the maximum number of lines used by the external item picker input box to render instances of an external content type.
Direction	This parameter specifies whether the external item picker control is displaying left-to-right or right-to-left.
MaxResults	This parameter MUST be ignored.
EntityButtonWidth	This parameter MUST be ignored.
QueryRequired	This parameter MUST be ignored.
RefreshOnOpen	This parameter MUST be ignored.
PickerTargetMode	This parameter MUST be ignored.
MultiItem	This parameter MUST be ignored.

ENTITY_PICKER_BUTTON_FONT: FONT, **FONT_SIZE**, **CHARACTER_SET**, **FONT_WEIGHT**, **FONT_ITALIC**, **FONT_UNDERLINE**, **FONT_STRIKETHROUGH.**

ENTITY_PICKER_STYLE: Semicolon-delimited list of (**STYLE_SIZE**?, **STYLE_MARGIN**?, **STYLE_TEXT_DECORATION**?, (**BACKGROUND-COLOR**: transparent; **STYLE_XD_BACKGROUND_COLOR**)?, **STYLE_BORDER**?, **STYLE_FONT**?, **STYLE_COLOR**?, **STYLE_VERTICAL_ALIGN**?, **STYLE_DIRECTION**?).

ENTITY_PICKER_BACKGROUND_COLOR: MUST be "2147483653" or 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 zero and 255, in the following formula:

- blue part * 65536 + green part * 256 + red part

ENTITY_PICKER_MAX_LINES: MUST be an **integer** between zero and 999, inclusive.

The **ENTITY_PICKER_DIRECTION** symbol specifies if the control is rendered left-to-right or right-to-left. The following table describes the possible values.

Value	Description
"0"	Use the form's orientation.
"1"	Left-to-right.
"2"	Right-to-left.

ENTITY_PICKER_ENTITY_NAMESPACE: The value MUST be an **entityNamespace**, as specified in section 2.2.1.2.129.

ENTITY_PICKER_ENTITY_NAME: The value MUST be an **entityName**, as specified in section 2.2.1.2.129.

ENTITY_PICKER_SYSTEM_INSTANCE: The value MUST be an **LobSystemInstance**, as specified in section 2.2.1.2.129.

ENTITY_PICKER_DISPLAY_FIELD: MUST be an encoded **string**, as specified in [XML10], and MUST be specified in a hierarchical fashion. The first token in the location MUST be the name of one of the fields (3) of the external content type. Any subsequent token MUST be the name of a field (3) contained in the field (3) identified by the previous token. Tokens MUST be separated by a single period ("."). If appearing within the name of a token, the following characters MUST be replaced by their encoded string counterparts:

Character	Encoded string counterpart
Left bracket ("[")	Backslash, left bracket ("\[")
Period (".")	Backslash, period ("\.")
Backslash ("\")	Backslash, backslash ("\\")

GROUP_XPATH MUST point to an XML node in the main data source.

The following table lists control-specific attributes used by the external item picker control.

Attribute	Section
xd:binding	2.4.2.6
xd:bindingProperty	2.4.2.7
xd:bindingType	2.4.2.8
xd:boundProp	2.4.2.9
xd:CtrlId	2.4.2.10
xd:disableEditing	2.4.2.12
xd:server	2.4.2.37.8
xd:xctname	2.4.2.35

The **xdImage:getImageUrl** XSL function extension, as specified in section 2.4.3.5, is used by the external item picker control.

2.4.1.21.5 Embedded Picture Control

The embedded picture control enables users to include a picture in a form (1). The following table describes the symbols for an embedded picture control.

Symbol	Description
SIMPLE_EMBEDDED_PICTURE	An embedded picture is a control that displays a picture attached by a user filling

Symbol	Description
	out the form (1).
EMBEDDED_PICTURE_WITH_CONDITIONAL_FORMATTING	Similar to **SIMPLE_EMBEDDED_PICTURE**, but allows conditional formatting and disabling.

SIMPLE_EMBEDDED_PICTURE:

```
<xsl:if test="function-available('xdImage:getImageUrl')">
    <img class="xdInlinePicture" hideFocus="1" (title="ANY_STRING")?
(accessKey="SINGLE_CHARACTER")? (style="EMBEDDED_PICTURE_STYLE")? (alt="ANY_STRING")?
(align="ALIGN")? tabStop="true" xd:boundProp="" xd:binding="LEAF_XPATH1" tabIndex="TAB_INDEX"
xd:xctname="InlineImage" xd:CtrlId="CONTROL_ID" (Linked="true")? (xd:disableEditing="yes")?
xd:inline="LEAF_XPATH1" src="{xdImage:getImageUrl(LEAF_XPATH1)}" />
</xsl:if>
```

EMBEDDED_PICTURE_WITH_CONDITIONAL_FORMATTING:

```
<xsl:if test="function-available('xdImage:getImageUrl')">
    <img class="xdInlinePicture" hideFocus="1" (title="ANY_STRING")?
(accessKey="SINGLE_CHARACTER")? (style="EMBEDDED_PICTURE_STYLE")? (alt="ANY_STRING")?
(align="ALIGN")? tabStop="true" xd:boundProp="" xd:binding="LEAF_XPATH1" tabIndex="TAB_INDEX"
xd:xctname="InlineImage" xd:CtrlId="CONTROL_ID" (Linked="true")? (xd:disableEditing="yes")?
xd:inline="LEAF_XPATH1" src="{xdImage:getImageUrl(LEAF_XPATH1)}">
        EMBEDDED_PICTURE_CONDITIONAL_FORMATTING
    </img>
</xsl:if>
```

EMBEDDED_PICTURE_CONDITIONAL_FORMATTING:

```
(<xsl:attribute name="style">EMBEDDED_PICTURE_STYLE?
    <xsl:choose>
        (<xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_DISPLAY_NONE;
STYLE_CAPTION</xsl:when>|
        <xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_CAPTION</xsl:when>|
        <xsl:when
test="BOOLEAN_XPATH_EXPRESSION">EMBEDDED_PICTURE_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>)?
```

EMBEDDED_PICTURE_STYLE: Semicolon-delimited list of (**STYLE_WIDTH**?, **STYLE_HEIGHT**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_BORDER**?, **STYLE_COLOR**?, **STYLE_DIRECTION**?)

EMBEDDED_PICTURE_CONDITIONAL_FORMATTING_STYLE: Semicolon-delimited list of (**STYLE_BACKGROUND_COLOR**?, **STYLE_COLOR**?, **STYLE_CAPTION**)

LEAF_XPATH$_1$ MUST point to an XML node in the main data source.

The following table lists control-specific attributes used by the embedded picture control.

Attribute	Section
xd:binding	2.4.2.6
xd:boundProp	2.4.2.9
xd:CtrlId	2.4.2.10
xd:disableEditing	2.4.2.12
xd:xctname	2.4.2.35
xd:inline	2.4.2.18

The **xdImage:getImageUrl** XSL function extension, as specified in section 2.4.3.5, is used by the embedded picture control.

2.4.1.21.6 Hyperlink Input Control

The hyperlink input control allows the user to create an editable hyperlink that navigates the default browser to a specified URL. The following table describes the symbols for a hyperlink input control.

Symbol	Description
HYPERLINK_BOX_NO_BINDING	A static hyperlink input control where the user cannot specify a web location.
HYPERLINK_BOX_SINGLE_BOUND	Hyperlink inputs are editable controls where the user can specify a web location and open a new browser window to that location.
HYPERLINK_BOX_SINGLE_BOUND_WITH_CONDITIONAL_FORMATTING	Similar to **HYPERLINK_BOX_SINGLE_BOUND**, except that the hyperlink input control supports conditional formatting.
HYPERLINK_BOX_DUAL_BOUND	Similar to **HYPERLINK_BOX_SINGLE_BOUND**, except that the hyperlink input control is bound to a second node that provides the text for the hyperlink.
HYPERLINK_BOX_DUAL_BOUND_WITH_CONDITIONAL_FORMATTING	Similar to **HYPERLINK_BOX_DUAL_BOUND**, except that the hyperlink input control supports conditional formatting.
HYPERLINK_BOX_SINGLE_BOUND_WITH_COMPLEX_XPATHS	Similar to **HYPERLINK_BOX_SINGLE_BOUND**, except that the hyperlink input control is bound to a complex XPath expression.
HYPERLINK_BOX_SINGLE_BOUND_WITH_CONDITIONAL_FOR	Similar to

Symbol	Description
MATTING_AND_COMPLEX_XPATHS	**HYPERLINK_BOX_SINGLE_BOUND_WITH_COMPLEX_XPATHS**, except that the hyperlink input control supports conditional formatting.
HYPERLINK_BOX_DUAL_BOUND_WITH_COMPLEX_XPATHS	Similar to **HYPERLINK_BOX_SINGLE_BOUND_WITH_COMPLEX_XPATHS**, except that the hyperlink input control is bound to a node in the form (1) that provides the text for the hyperlink.
HYPERLINK_BOX_DUAL_BOUND_WITH_CONDITIONAL_FORMATTING_AND_COMPLEX_XPATHS	Similar to **HYPERLINK_BOX_DUAL_BOUND_WITH_COMPLEX_XPATHS**, except that the hyperlink input control supports conditional formatting.

HYPERLINK_BOX_NO_BINDING:

```
<span HYPERLINK_BOX_COMMON_ATTRIBUTES (style="HYPERLINK_BOX_STYLE")?
(accessKey="SINGLE_CHARACTER")? (tabIndex="TAB_INDEX">)?
    <button class="xdHyperlinkBoxButtonClickable">
        <img style="VERTICAL-ALIGN: bottom" src="res://infopath.exe/hyperlinkbox.png" />
    </button>
    <span style="WIDTH: 5px"></span>ANY_STRING</span>
```

HYPERLINK_BOX_SINGLE_BOUND:

```
<span HYPERLINK_BOX_COMMON_ATTRIBUTES (xd:disableEditing="yes")? tabIndex="-1"
xd:boundProp="href" xd:binding="LEAF_XPATH1" (style="HYPERLINK_BOX_STYLE")?>
    <button class="xdHyperlinkBoxButtonClickable" tabIndex="TAB_INDEX1">
        (<xsl:if test="string-length(LEAF_XPATH1)=0">
            <xsl:attribute name="accessKey">SINGLE_CHARACTER1</xsl:attribute>
        </xsl:if>)?
        <img style="vertical-align:bottom" src="res://infopath.exe/hyperlinkbox.png" />
    </button>
    <span style="width:5px;" />
    <xsl:choose>
        <xsl:when test="string-length(LEAF_XPATH1)!=0">
            <A class="hyperlinkAnchor" tabIndex="TAB_INDEX1" (accesskey="SINGLE_CHARACTER1")?
(style="(STYLE_COLOR;)?")?>
                <xsl:attribute name="title">
                    <xsl:value-of select="LEAF_XPATH1" />
                </xsl:attribute>
                <xsl:attribute name="href">
                    <xsl:value-of select="LEAF_XPATH1" />
                </xsl:attribute>
                <xsl:value-of select="substring(normalize-space(LEAF_XPATH1), 0, 256)" />
            </A>
        </xsl:when>
        <xsl:otherwise>ANY_STRING</xsl:otherwise>
    </xsl:choose>
```

```
</span>
```

HYPERLINK_BOX_SINGLE_BOUND_WITH_CONDITIONAL_FORMATTING:

```
<span HYPERLINK_BOX_COMMON_ATTRIBUTES (xd:disableEditing="yes")? tabIndex="-1"
xd:boundProp="href" xd:binding="LEAF_XPATH1">
    CHECK_FOR_GETDOM_BEGIN1
    HYPERLINK_BOX_CONDITIONAL_FORMATTING_WITH_STYLE
    CHECK_FOR_GETDOM_END1
    <button class="xdHyperlinkBoxButtonClickable" tabIndex="TAB_INDEX1">
        (<xsl:if test="string-length(LEAF_XPATH1)=0">
            <xsl:attribute name="accessKey">SINGLE_CHARACTER1</xsl:attribute>
        </xsl:if>)?
        <img style="vertical-align:bottom" src="res://infopath.exe/hyperlinkbox.png" />
    </button>
    <span style="width:5px;" />
    <xsl:choose>
        <xsl:when test="string-length(LEAF_XPATH1)!=0">
            <A class="hyperlinkAnchor" tabIndex="TAB_INDEX1" (accessKey="SINGLE_CHARACTER1")?
(style="(STYLE_COLOR;)?")?>
                CHECK_FOR_GETDOM_BEGIN1
                HYPERLINK_BOX_ANCHOR_CONDITIONAL_FORMATTING_WITH_STYLE
                CHECK_FOR_GETDOM_END1
                <xsl:attribute name="title">
                    <xsl:value-of select="LEAF_XPATH1" />
                </xsl:attribute>
                <xsl:attribute name="href">
                    <xsl:value-of select="LEAF_XPATH1" />
                </xsl:attribute>
                <xsl:value-of select="substring(normalize-space(LEAF_XPATH1), 0, 256)" />
            </A>
        </xsl:when>
        <xsl:otherwise>ANY_STRING</xsl:otherwise>
    </xsl:choose>
</span>
```

HYPERLINK_BOX_DUAL_BOUND:

```
<span HYPERLINK_BOX_COMMON_ATTRIBUTES (xd:disableEditing="yes")? tabIndex="-1"
xd:binding_secondary="LEAF_XPATH2" xd:boundPropSecondary="displaytext" xd:boundProp="href"
xd:binding="LEAF_XPATH1" (style="HYPERLINK_BOX_STYLE")?>
    <button class="xdHyperlinkBoxButtonClickable" tabIndex="TAB_INDEX1">
        (<xsl:if test="string-length(LEAF_XPATH1)=0">
            <xsl:attribute name="accessKey">SINGLE_CHARACTER1</xsl:attribute>
        </xsl:if>)?
        <img style="vertical-align:bottom" src="res://infopath.exe/hyperlinkbox.png" />
    </button>
    <span style="width:5px;" />
    <xsl:choose>
        <xsl:when test="string-length(LEAF_XPATH1)!=0">
            <A class="hyperlinkAnchor" tabIndex="TAB_INDEX1" (accessKey="SINGLE_CHARACTER1")?
(style="(STYLE_COLOR;)?")?>
                <xsl:attribute name="title">
                    <xsl:value-of select="LEAF_XPATH1" />
                </xsl:attribute>
```

```
                <xsl:attribute name="href">
                    <xsl:value-of select="LEAF_XPATH1" />
                </xsl:attribute>
                <xsl:choose>
                    <xsl:when test="string-length(normalize-space(LEAF_XPATH2))=0">
                        <xsl:value-of select="substring(normalize-space(LEAF_XPATH1), 0,
256)" />
                    </xsl:when>
                    <xsl:otherwise>
                        <xsl:value-of select="substring(normalize-space(LEAF_XPATH2), 0,
256)" />
                    </xsl:otherwise>
                </xsl:choose>
            </A>
        </xsl:when>
        <xsl:otherwise>ANY_STRING</xsl:otherwise>
    </xsl:choose>
</span>
```

HYPERLINK_BOX_DUAL_BOUND_WITH_CONDITIONAL_FORMATTING:

```
<span HYPERLINK_BOX_COMMON_ATTRIBUTES (xd:disableEditing="yes")? tabIndex="-1"
xd:binding_secondary="LEAF_XPATH2" xd:boundPropSecondary="displaytext" xd:boundProp="href"
xd:binding="LEAF_XPATH1">
    CHECK_FOR_GETDOM_BEGIN1
    HYPERLINK_BOX_CONDITIONAL_FORMATTING_WITH_STYLE
    CHECK_FOR_GETDOM_END1
    <button class="xdHyperlinkBoxButtonClickable" tabIndex="TAB_INDEX1">
        (<xsl:if test="string-length(LEAF_XPATH1)=0">
            <xsl:attribute name="accessKey">SINGLE_CHARACTER1</xsl:attribute>
        </xsl:if>)?
        <img style="vertical-align:bottom" src="res://infopath.exe/hyperlinkbox.png" />
    </button>
    <span style="width:5px;" />
    <xsl:choose>
        <xsl:when test="string-length(LEAF_XPATH1)!=0">
            <A class="hyperlinkAnchor" tabIndex="TAB_INDEX1" (accessKey="SINGLE_CHARACTER1")?
(style="(STYLE_COLOR;)?")?>
                CHECK_FOR_GETDOM_BEGIN1
                HYPERLINK_BOX_ANCHOR_CONDITIONAL_FORMATTING_WITH_STYLE
                CHECK_FOR_GETDOM_END1
                <xsl:attribute name="title">
                    <xsl:value-of select="LEAF_XPATH1" />
                </xsl:attribute>
                <xsl:attribute name="href">
                    <xsl:value-of select="LEAF_XPATH1" />
                </xsl:attribute>
                <xsl:choose>
                    <xsl:when test="string-length(normalize-space(LEAF_XPATH2))=0">
                        <xsl:value-of select="substring(normalize-space(LEAF_XPATH1), 0,
256)" />
                    </xsl:when>
                    <xsl:otherwise>
                        <xsl:value-of select="substring(normalize-space(LEAF_XPATH2), 0,
256)" />
                    </xsl:otherwise>
```

```
                </xsl:choose>
            </A>
        </xsl:when>
        <xsl:otherwise>ANY_STRING</xsl:otherwise>
    </xsl:choose>
</span>
```

HYPERLINK_BOX_SINGLE_BOUND_WITH_COMPLEX_XPATHS:

```
<span HYPERLINK_BOX_COMMON_ATTRIBUTES xd:disableEditing="yes" tabIndex="-1"
xd:boundProp="href" xd:binding="STRING_XPATH_EXPRESSION1" (style="HYPERLINK_BOX_STYLE")?>
    (<xsl:if test="function-available('xdXDocument:GetDOM')" />)?
    <button class="xdHyperlinkBoxButtonClickable" tabIndex="TAB_INDEX1">
        (<xsl:if test="string-length(STRING_XPATH_EXPRESSION1)=0">
            <xsl:attribute name="accessKey">SINGLE_CHARACTER1</xsl:attribute>
        </xsl:if>)?
        <img style="vertical-align:bottom" src="res://infopath.exe/hyperlinkbox.png" />
    </button>
    <span style="width:5px;" />
    <xsl:choose>
        <xsl:when test="string-length(STRING_XPATH_EXPRESSION1)!=0">
            <A class="hyperlinkAnchor" tabIndex="TAB_INDEX1" (accessKey="SINGLE_CHARACTER1")?
(style="(STYLE_COLOR;)?")?>
                (<xsl:if test="function-available('xdXDocument:GetDOM')" />)?
                <xsl:attribute name="title">
                    <xsl:value-of select="STRING_XPATH_EXPRESSION1" />
                </xsl:attribute>
                <xsl:attribute name="href">
                    <xsl:value-of select="STRING_XPATH_EXPRESSION1" />
                </xsl:attribute>
                <xsl:value-of select="substring(normalize-space(STRING_XPATH_EXPRESSION1), 0,
256)" />
            </A>
        </xsl:when>
        <xsl:otherwise>ANY_STRING</xsl:otherwise>
    </xsl:choose>
</span>
```

HYPERLINK_BOX_SINGLE_BOUND_WITH_CONDITIONAL_FORMATTING_AND_COMPLEX_XPATHS:

```
<span HYPERLINK_BOX_COMMON_ATTRIBUTES xd:disableEditing="yes" tabIndex="-1"
xd:boundProp="href" xd:binding="STRING_XPATH_EXPRESSION1">
    CHECK_FOR_GETDOM_BEGIN1
    HYPERLINK_BOX_CONDITIONAL_FORMATTING_WITH_STYLE
    CHECK_FOR_GETDOM_END1
    <button class="xdHyperlinkBoxButtonClickable" tabIndex="TAB_INDEX1">
        (<xsl:if test="string-length(STRING_XPATH_EXPRESSION1)=0">
            <xsl:attribute name="accessKey">SINGLE_CHARACTER1</xsl:attribute>
        </xsl:if>)?
        <img style="vertical-align:bottom" src="res://infopath.exe/hyperlinkbox.png" />
    </button>
    <span style="width:5px;" />
    <xsl:choose>
        <xsl:when test="string-length(STRING_XPATH_EXPRESSION1)!=0">
```

```
            <A class="hyperlinkAnchor" tabIndex="TAB_INDEX1" (accessKey="SINGLE_CHARACTER1")?
(style="(STYLE_COLOR;)?")?>
                CHECK_FOR_GETDOM_BEGIN1
                HYPERLINK_BOX_ANCHOR_CONDITIONAL_FORMATTING_WITH_STYLE
                CHECK_FOR_GETDOM_END1
                <xsl:attribute name="title">
                    <xsl:value-of select="STRING_XPATH_EXPRESSION1" />
                </xsl:attribute>
                <xsl:attribute name="href">
                    <xsl:value-of select="STRING_XPATH_EXPRESSION1" />
                </xsl:attribute>
                <xsl:value-of select="substring(normalize-space(STRING_XPATH_EXPRESSION1), 0,
256)" />
            </A>
        </xsl:when>
        <xsl:otherwise>ANY_STRING</xsl:otherwise>
    </xsl:choose>
</span>
```

HYPERLINK_BOX_DUAL_BOUND_WITH_COMPLEX_XPATHS:

```
<span HYPERLINK_BOX_COMMON_ATTRIBUTES xd:disableEditing="yes" tabIndex="-1"
xd:binding_secondary="LEAF_XPATH2" xd:boundPropSecondary="displaytext" xd:boundProp="href"
xd:binding="STRING_XPATH_EXPRESSION1" (style="HYPERLINK_BOX_STYLE")?>
    (<xsl:if test="function-available('xdXDocument:GetDOM')" />)?
    <button class="xdHyperlinkBoxButtonClickable" tabIndex="TAB_INDEX1">
        (<xsl:if test="string-length(STRING_XPATH_EXPRESSION1)=0">
            <xsl:attribute name="accessKey">SINGLE_CHARACTER1</xsl:attribute>
        </xsl:if>)?
        <img style="vertical-align:bottom" src="res://infopath.exe/hyperlinkbox.png" />
    </button>
    <span style="width:5px;" />
    <xsl:choose>
        <xsl:when test="string-length(STRING_XPATH_EXPRESSION1)!=0">
            <A class="hyperlinkAnchor" tabIndex="TAB_INDEX1" (accessKey="SINGLE_CHARACTER1")?
(style="(STYLE_COLOR;)?")?>
                (<xsl:if test="function-available('xdXDocument:GetDOM')" />)?
                <xsl:attribute name="title">
                    <xsl:value-of select="STRING_XPATH_EXPRESSION1" />
                </xsl:attribute>
                <xsl:attribute name="href">
                    <xsl:value-of select="STRING_XPATH_EXPRESSION1" />
                </xsl:attribute>
                <xsl:choose>
                    <xsl:when test="string-length(normalize-space(LEAF_XPATH2))=0">
                        <xsl:value-of select="substring(normalize-
space(STRING_XPATH_EXPRESSION1), 0, 256)" />
                    </xsl:when>
                    <xsl:otherwise>
                        <xsl:value-of select="substring(normalize-space(LEAF_XPATH2), 0,
256)" />
                    </xsl:otherwise>
                </xsl:choose>
            </A>
        </xsl:when>
        <xsl:otherwise>ANY_STRING</xsl:otherwise>
```

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

HYPERLINK_BOX_DUAL_BOUND_WITH_CONDITIONAL_FORMATTING_AND_COMPLEX_XPATHS:

```
<span HYPERLINK_BOX_COMMON_ATTRIBUTES xd:disableEditing="yes" tabIndex="-1"
xd:binding_secondary="LEAF_XPATH2" xd:boundPropSecondary="displaytext" xd:boundProp="href"
xd:binding="STRING_XPATH_EXPRESSION1">
    CHECK_FOR_GETDOM_BEGIN1
    HYPERLINK_BOX_CONDITIONAL_FORMATTING_WITH_STYLE
    CHECK_FOR_GETDOM_END1
    <button class="xdHyperlinkBoxButtonClickable" tabIndex="TAB_INDEX1">
        (<xsl:if test="string-length(STRING_XPATH_EXPRESSION1)=0">
            <xsl:attribute name="accessKey">SINGLE_CHARACTER1</xsl:attribute>
        </xsl:if>)?
        <img style="vertical-align:bottom" src="res://infopath.exe/hyperlinkbox.png" />
    </button>
    <span style="width:5px;" />
    <xsl:choose>
        <xsl:when test="string-length(STRING_XPATH_EXPRESSION1)!=0">
            <A class="hyperlinkAnchor" tabIndex="TAB_INDEX1" (accessKey="SINGLE_CHARACTER1")?
(style="(STYLE_COLOR;)?")?>
                CHECK_FOR_GETDOM_BEGIN1
                HYPERLINK_BOX_ANCHOR_CONDITIONAL_FORMATTING_WITH_STYLE
                CHECK_FOR_GETDOM_END1
                <xsl:attribute name="title">
                    <xsl:value-of select="STRING_XPATH_EXPRESSION1" />
                </xsl:attribute>
                <xsl:attribute name="href">
                    <xsl:value-of select="STRING_XPATH_EXPRESSION1" />
                </xsl:attribute>
                <xsl:choose>
                    <xsl:when test="string-length(normalize-space(LEAF_XPATH2))=0">
                        <xsl:value-of select="substring(normalize-
space(STRING_XPATH_EXPRESSION1), 0, 256)" />
                    </xsl:when>
                    <xsl:otherwise>
                        <xsl:value-of select="substring(normalize-space(LEAF_XPATH2), 0,
256)" />
                    </xsl:otherwise>
                </xsl:choose>
            </A>
        </xsl:when>
        <xsl:otherwise>ANY_STRING</xsl:otherwise>
    </xsl:choose>
</span>
```

HYPERLINK_BOX_CONDITIONAL_FORMATTING_STYLE: Semicolon-delimited list of (**STYLE_TEXT_DECORATION**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_FONT**?, **STYLE_COLOR**?, **STYLE_CAPTION**)

HYPERLINK_BOX_ANCHOR_CONDITIONAL_FORMATTING_WITH_STYLE:

```
<xsl:attribute name="style">(STYLE_COLOR;)?
```

```
    HYPERLINK_BOX_CONDITIONAL_FORMATTING
</xsl:attribute>
HYPERLINK_BOX_CONDITIONAL_DISABLING
```

HYPERLINK_BOX_CONDITIONAL_FORMATTING_WITH_STYLE:

```
<xsl:attribute name="style">HYPERLINK_BOX_STYLE?
    HYPERLINK_BOX_CONDITIONAL_FORMATTING
</xsl:attribute>
HYPERLINK_BOX_CONDITIONAL_DISABLING
```

HYPERLINK_BOX_CONDITIONAL_FORMATTING:

```
<xsl:choose>
    (<xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_DISPLAY_NONE; STYLE_CAPTION</xsl:when> |
    <xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_CAPTION</xsl:when> |
    <xsl:when test="BOOLEAN_XPATH_EXPRESSION">
        HYPERLINK_BOX_CONDITIONAL_FORMATTING_STYLE
    </xsl:when>)+
</xsl:choose>
```

HYPERLINK_BOX_CONDITIONAL_DISABLING:

```
(<xsl:choose>
    (<xsl:when test="BOOLEAN_XPATH_EXPRESSION"/> |
    <xsl:when test="BOOLEAN_XPATH_EXPRESSION">
        <xsl:attribute name="xd:disableEditing">yes</xsl:attribute>
    </xsl:when>)+
</xsl:choose>)?
```

HYPERLINK_BOX_COMMON_ATTRIBUTES:

```
class="xdHyperlinkBox xdBehavior_Formatting xdHyperlinkBoxClickable" (title="ANY_STRING")?
(tabStop="true")? xd:CtrlId="CONTROL_ID" xd:xctname="HyperlinkBox"
(xd:postbackModel="POSTBACKMODEL")?
```

HYPERLINK_BOX_STYLE: Semicolon-delimited list of (**STYLE_WIDTH**?, **STYLE_FONT**?, **STYLE_HEIGHT**?, **STYLE_BORDER**?, **STYLE_VERTICAL_ALIGN**?, **STYLE_TEXT_DECORATION**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_COLOR**?)

Within this control, any usage of **STYLE_COLOR** MUST yield the same value, except for the **HYPERLINK_BOX_CONDITIONAL_FORMATTING_STYLE** production.

Within this control, any usage of **HYPERLINK_BOX_CONDITIONAL_FORMATTING** MUST yield the same value.

Within this control, any usage of **HYPERLINK_BOX_CONDITIONAL_DISABLING** MUST yield the same value.

Within this control, if any optional unit in a production defines the **accessKey** attribute, all optional units in the production containing the **accessKey** attribute MUST have a non-empty yield.

The following table lists the control-specific attributes used by the hyperlink control.

Attribute	Section
xd:binding	2.4.2.6
xd:binding_secondary	2.4.2.37.2
xd:boundProp	2.4.2.9
xd:boundPropSecondary	2.4.2.37.3
xd:CtrlId	2.4.2.10
xd:disableEditing	2.4.2.12
xd:postbackModel	2.4.2.29
xd:xctname	2.4.2.35

2.4.1.21.7 List Controls (Bulleted List Control, Numbered List Control and Plain List Control)

The list controls allow the user to enter multiple items of text in the form (1). The following table describes the symbols for a list control.

Symbol	Description
LIST_SIMPLE	The list control with no conditional formatting.
LIST_WITH_CONDITIONAL_FORMATTING	The list control with conditional formatting.

LIST_SIMPLE:

```
<span style="LIST_STYLE" class="xdRepeating" title="ANY_STRING" xd:xctname="(BulletedList |
bulletedlist | NumberedList | numberedlist | PlainList | plainlist)"
(xd:postbackModel="POSTBACKMODEL")?>
    <ol style="LIST_STYLE_TYPE MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px">
        <xsl:for-each select="REPEATING_LEAF_XPATH">
            <li>
                <span hideFocus="1" class="xdListItem" xd:xctname="ListItem_Plain"
tabIndex="TAB_INDEX1" xd:binding="." xd:CtrlId="CTRL_ID" (xd:disableEditing="yes" |
contentEditable="true")? (xd:inputScopeId="INPUT_SCOPE_ID")? (xd:allowNonMatching="yes")?
style="LIST_ITEM_STYLE">
                    <xsl:value-of select="." />
                </span>
            </li>
        </xsl:for-each>
    </ol>
</span>
LIST_INSERT_LINK
```

LIST_WITH_CONDITIONAL_FORMATTING:

```
<span style="LIST_STYLE" class="xdRepeating" title="ANY_STRING" xd:xctname="(BulletedList |
bulletedlist | NumberedList | numberedlist | PlainList | plainlist)"
(xd:postbackModel="POSTBACKMODEL")?>
    <ol style="LIST_STYLE_TYPE MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px">
        <xsl:for-each select="REPEATING_LEAF_XPATH">
            <li>
                <span hideFocus="1" class="xdListItem" xd:xctname="ListItem_Plain"
tabIndex="TAB_INDEX1" xd:binding="." xd:CtrlId="CTRL_ID" (xd:disableEditing="yes" |
contentEditable="true")? (xd:inputScopeId="INPUT_SCOPE_ID")? (xd:allowNonMatching="yes")? >
                    CHECK_FOR_GETDOM_BEGIN1
                        LIST_CONDITIONAL_FORMATTING
                    <xsl:value-of select="." />
                    CHECK_FOR_GETDOM_END1
                </span>
            </li>
        </xsl:for-each>
    </ol>
</span>
LIST_INSERT_LINK
```

LIST_CONDITIONAL_FORMATTING:

```
(<xsl:attribute name="style">
    LIST_ITEM_STYLE
    <xsl:choose>
        (<xsl:when test="BOOLEAN_XPATH_EXPRESSIONX">STYLE_CAPTION</xsl:when>|
        <xsl:when test="BOOLEAN_XPATH_EXPRESSIONY">LIST_CONDITIONAL_FORMATTING_STYLE
        </xsl:when>)+
    </xsl:choose>
</xsl:attribute>)?
(<xsl:choose>
    (<xsl:when test="BOOLEAN_XPATH_EXPRESSIONX"/>|
    <xsl:when test="BOOLEAN_XPATH_EXPRESSIONY">
        <xsl:attribute name="contentEditable">false</xsl:attribute>
    </xsl:when>)+
</xsl:choose>)?
```

LIST_INSERT_LINK:

```
(<div class="optionalPlaceholder" xd:xmlToEdit="XML_TO_EDIT_NAME" tabIndex="TAB_INDEX1"
xd:action="xTextList::insert" style="MARGIN-LEFT: 40px; STYLE_WIDTH"> ANY_STRING</div>)?
```

LIST_CONDITIONAL_FORMATTING_STYLE: Semicolon-delimited list of (**STYLE_FONT_WEIGHT**?, **STYLE_TEXT_DECORATION**?, **STYLE_FONT_STYLE**?, **STYLE_COLOR**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_CAPTION**)

LIST_ITEM_STYLE:

```
WIDTH: 100%; (WORD-WRAP: break-word; WHITE-SPACE: normal)?
```

LIST_STYLE: Semicolon-delimited list of (**STYLE_BORDER**?, **STYLE_PADDING**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_FONT_STYLE**?, **STYLE_MARGIN**?, **STYLE_WIDTH**?, **STYLE_HEIGHT**?, **STYLE_FONT_FAMILY**?, **STYLE_COLOR**?, **STYLE_FONT_SIZE**?,

STYLE_VERTICAL_ALIGN?, **STYLE_FONT_WEIGHT**?, **STYLE_TEXT_DECORATION**?, **STYLE_DIRECTION**?)

BULLETED_LIST_STYLE_TYPE: LIST-STYLE-TYPE: (disc or circle or square).

NUMBERED_LIST_STYLE_TYPE: LIST-STYLE-TYPE: (decimal or lower-roman or upper-roman or lower-alpha or upper-alpha).

PLAIN_LIST_STYLE_TYPE: LIST-STYLE-TYPE: none.

LIST_STYLE_TYPE: (**BULLETED_LIST_STYLE_TYPE** or **NUMBERED_LIST_STYLE_TYPE** or **PLAIN_LIST_STYLE_TYPE**);

The value for **LIST_STYLE_TYPE** is determined by the value of the **xd:xctname** attribute and **MUST** be set as specified in the following table.

xd:xctname	LIST_STYLE_TYPE
"BulletedList"	**BULLETED_LIST_STYLE_TYPE**
"bulletedlist"	**BULLETED_LIST_STYLE_TYPE**
"NumberedList"	**NUMBERED_LIST_STYLE_TYPE**
"numberedlist"	**NUMBERED_LIST_STYLE_TYPE**
"PlainList"	**PLAIN_LIST_STYLE_TYPE**
"plainlist"	**PLAIN_LIST_STYLE_TYPE**

The following table lists control-specific attributes used by the list controls.

Attribute	Section
xd:binding	2.4.2.6
xd:CtrlId	2.4.2.10
xd:postbackModel	2.4.2.29
xd:xctname	2.4.2.35
xd:disableEditing	2.4.2.12
xd:allowNonMatching	2.4.2.2
xd:inputscope	2.4.2.20
xd:action	2.4.2.1

2.4.1.21.8 Linked Picture Control

The linked picture control allows a user to specify a URL of a picture that is displayed in the form (1). The user can also specify a description for the picture. The following table describes the symbols for a linked picture control.

Symbol	Description
LINKED_PICTURE_SIMPLE	The linked picture control only allows the user to specify a URL of a picture. This control is bound to a **LEAF_XPATH**.
LINKED_PICTURE_EXPRESSION	The linked picture control only allows the user to specify a URL of a picture. The control is bound to a **STRING_XPATH_EXPRESSION**.
LINKED_PICTURE_SIMPLE_WITH_DESCRIPTION	The linked picture control allows the user to specify a URL of a picture, as well as a description of the picture. The binding for the URL is a **LEAF_XPATH**.
LINKED_PICTURE_EXPRESSION_WITH_DESCRIPTION	The linked picture control allows the user to specify a URL of a picture, as well as a description of the picture. The binding for the URL is a **STRING_XPATH_EXPRESSION**.

LINKED_PICTURE_SIMPLE:

```
<img hideFocus="1" class="xdLinkedPicture" xd:xctname="LinkedImage" xd:boundProp="src"
tabStop="true" alt="ANY_STRING1" displaytext="" (align="ALIGN")? xd:CtrlId="CONTROL_ID"
(xd:binding="LEAF_XPATH1")? (accessKey="SINGLE_CHARACTER")? (title="ANY_STRING2")?
(width="WIDTH" height="HEIGHT")? (tabIndex="TAB_INDEX")? (xd:postbackModel="POSTBACKMODEL")?
(disabled="disabled" xd:disableEditing="yes")? (style="LINKED_PICTURE_STYLE")?>
    CHECK_FOR_GETDOM_BEGIN1
    LINKED_PICTURE_CONDITIONAL_FORMATTING?
    <xsl:attribute name="src">
        <xsl:value-of select="LEAF_XPATH1" />
    </xsl:attribute>
    CHECK_FOR_GETDOM_END1
</img>
```

LINKED_PICTURE_EXPRESSION:

```
<img hideFocus="1" class="xdLinkedPicture" xd:xctname="LinkedImage" xd:boundProp="src"
tabStop="true" displaytext="" disabled="disabled" xd:disableEditing="yes" alt="ANY_STRING1"
xd:CtrlId="CONTROL_ID" xd:binding="STRING_XPATH_EXPRESSION1" (align="ALIGN")?
(accessKey="SINGLE_CHARACTER")? (title="ANY_STRING2")? (height="HEIGHT" width="WIDTH")?
(tabIndex="TAB_INDEX")? (xd:postbackModel="POSTBACKMODEL")? (style="LINKED_PICTURE_STYLE")?>
    CHECK_FOR_GETDOM_BEGIN1
    LINKED_PICTURE_CONDITIONAL_FORMATTING?
    <xsl:attribute name="src">
        <xsl:value-of select="STRING_XPATH_EXPRESSION1" />
    </xsl:attribute>
    CHECK_FOR_GETDOM_END1
</img>
```

LINKED_PICTURE_SIMPLE_WITH_DESCRIPTION:

```
<img hideFocus="1" class="xdLinkedPicture" xd:xctname="LinkedImage" xd:boundProp="src"
xd:boundPropSecondary="displaytext" tabStop="true" xd:CtrlId="CONTROL_ID" (align="ALIGN")?
xd:binding="LEAF_XPATH1" xd:binding_secondary="LEAF_XPATH2" (accessKey="SINGLE_CHARACTER")?
```

```
(disabled="disabled" xd:disableEditing="yes")? (height="HEIGHT" width="WIDTH")?
(tabIndex="TAB_INDEX")? (xd:postbackModel="POSTBACKMODEL")? (style="LINKED_PICTURE_STYLE")?>
    CHECK_FOR_GETDOM_BEGIN1
    LINKED_PICTURE_CONDITIONAL_FORMATTING?
    <xsl:attribute name="src">
        <xsl:value-of select=" LEAF_XPATH1" />
    </xsl:attribute>
    <xsl:attribute name="displaytext">
        <xsl:value-of select="LEAF_XPATH2" />
    </xsl:attribute>
    <xsl:attribute name="alt">
        <xsl:choose>
            <xsl:when test="string-length(LEAF_XPATH1) &gt; 0">
                <xsl:value-of select="LEAF_XPATH2" />
            </xsl:when>
            (<xsl:otherwise/> | <xsl:otherwise>ANY_STRING1</xsl:otherwise>)
        </xsl:choose>
    </xsl:attribute>
    (<xsl:if test="string-length(LEAF_XPATH1) = 0">
        <xsl:attribute name="title">ANY_STRING2</xsl:attribute>
    </xsl:if>)?
    CHECK_FOR_GETDOM_END1
</img>
```

LINKED_PICTURE_EXPRESSION_WITH_DESCRIPTION:

```
<img hideFocus="1" class="xdLinkedPicture" xd:xctname="LinkedImage" xd:boundProp="src"
xd:boundPropSecondary="displaytext" tabStop="true" xd:CtrlId="CONTROL_ID"
xd:binding="STRING_XPATH_EXPRESSION1" xd:binding_secondary=" LEAF_XPATH1"  (align="ALIGN")?
(accessKey="SINGLE_CHARACTER")? disabled="disabled" xd:disableEditing="yes" (height="HEIGHT"
width="WIDTH")? (tabIndex="TAB_INDEX")? (xd:postbackModel="POSTBACKMODEL")?
(style="LINKED_PICTURE_STYLE")?>
    CHECK_FOR_GETDOM_BEGIN1
    LINKED_PICTURE_CONDITIONAL_FORMATTING?
    <xsl:attribute name="src">
        <xsl:value-of select="STRING_XPATH_EXPRESSION1" />
    </xsl:attribute>
    <xsl:attribute name="displaytext">
        <xsl:value-of select="LEAF_XPATH1" />
    </xsl:attribute>
    <xsl:attribute name="alt">
        <xsl:choose>
            <xsl:when test="string-length(STRING_XPATH_EXPRESSION1) &gt; 0">
                <xsl:value-of select="LEAF_XPATH1" />
            </xsl:when>
            (<xsl:otherwise/>)
        </xsl:choose>
    </xsl:attribute>
    (<xsl:if test="string-length(STRING_XPATH_EXPRESSION1) = 0">
        <xsl:attribute name="title">ANY_STRING2</xsl:attribute>
    </xsl:if>)?
    CHECK_FOR_GETDOM_END1
</img>
```

LINKED_PICTURE_CONDITIONAL_FORMATTING:

```
(<xsl:attribute name="style">LINKED_PICTURE_STYLE
    <xsl:choose>
        (<xsl:when test="BOOLEAN_XPATH_EXPRESSIONX">STYLE_DISPLAY_NONE;
STYLE_CAPTION</xsl:when>|
        <xsl:when test="BOOLEAN_XPATH_EXPRESSIONY">STYLE_CAPTION</xsl:when>|
        <xsl:when
test="BOOLEAN_XPATH_EXPRESSIONZ">LINKED_PICTURE_CONDITIONAL_FORMATTING_STYLE</xsl:when>)+
    </xsl:choose>
</xsl:attribute>)?
(<xsl:choose>
    (<xsl:when test="BOOLEAN_XPATH_EXPRESSIONX"/>|
    <xsl:when test="BOOLEAN_XPATH_EXPRESSIONY">
        <xsl:attribute name="disabled">true</xsl:attribute>
    </xsl:when>)+
</xsl:choose>)?
```

LINKED_PICTURE_STYLE: Semicolon-delimited list of (**STYLE_BORDER**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_COLOR**?, **STYLE_WIDTH**?, **STYLE_HEIGHT**?)

LINKED_PICTURE_CONDITIONAL_FORMATTING_STYLE: STYLE_DISPLAY_NONE or Semicolon-delimited list of (**STYLE_BACKGROUND_COLOR**?, **STYLE_COLOR**?, **STYLE_CAPTION**)

The following table lists control-specific attributes used by the linked picture controls.

Attribute	Section
xd:binding	2.4.2.6
xd:binding_secondary	2.4.2.37.2
xd:boundProp	2.4.2.9
xd:boundPropSecondary	2.4.2.37.3
xd:CtrlId	2.4.2.10
xd:postbackModel	2.4.2.29
xd:xctname	2.4.2.35
xd:disableEditing	2.4.2.12

2.4.1.21.9 Multiple-Selection List Box Control

The multiple-selection list box control enables the user to select multiple values from a list of options that can be specified manually by the form template designer, or is populated from a data source (2). The user is also able to enter custom values in the control. The following table describes the symbols for a multiple-selection list box control.

Symbol	Description
MSLB_WITH_STATIC_CHOICES	A multiple-selection list box is a control that allows the user to select multiple values from a list of options. The list of options is stored in the XSL.

Symbol	Description
MSLB_WITH_STATIC_CHOICES_AND_CUSTOM_ITEMS	Similar to **MSLB_WITH_STATIC_CHOICES**, but in addition to being able to select options, the user is able to enter custom values.
MSLB_WITH_STATIC_CHOICES_AND_CUSTOM_ITEMS_AND_DATA_FORMAT	Similar to **MSLB_WITH_STATIC_CHOICES_AND_CUSTOM_ITEMS**, but with data formatting.
MSLB_WITH_DYNAMIC_CHOICES	Similar to **MSLB_WITH_STATIC_CHOICES**, with the exception that the list of options is populated from another location within the form's data source (2).
MSLB_WITH_DYNAMIC_CHOICES_AND_CUSTOM_ITEMS	Similar to **MSLB_WITH_DYNAMIC_CHOICES,** but in addition to being able to select options, the user is able to enter custom values.
MSLB_WITH_DYNAMIC_CHOICES_AND_CUSTOM_ITEMS_DATA_FORMAT	Similar to **MSLB_WITH_MSLB_WITH_DYNAMIC_CHOICES_AND_CUSTOM_ITEMS**, but with data formatting.

In any production defined in this section, any use of a **REPEATING_LEAF_XPATH** with the same subscript MUST have an identical yield.

In any production defined in this section, any use of a **MSLB_STYLES** with the same subscript MUST have an identical yield.

In any production defined in this section, any use of a **TAB_INDEX** with the same subscript MUST have an identical yield.

For each instance of **ANY_STRING**$_X$ in **MSLB_FOR_EACH_SELECT** there MUST be an identical instance of **ANY_STRING**$_X$ in **MSLB_STATIC_OPTIONS**.

MSLB_FOR_EACH_SELECT:

```
REPEATING_LEAF_XPATH1[(.!=&quot;&quot;) | ("and" separated list of
(.!=&quot;ANY_STRINGX&quot;)+)]
```

MSLB_WITH_STATIC_CHOICES:

```
<xsl:choose>
    <xsl:when test="function-available('ipApp:GetMajorVersion') and ipApp:GetMajorVersion()
&gt;= 12">
        <span style="MSLB_STYLES1" class="xdMultiSelectList" title="ANY_STRING1"
(xd:postbackModel="POSTBACKMODEL")? xd:CtrlId="CONTROL_ID" xd:xctname="multiselectlistbox"
(xd:boundProp="value")? tabIndex="-1" xd:ref="REPEATING_LEAF_XPATH1" (DISABLED="true")?>
            MSLB_CONDITIONAL_FORMATTING1
            MSLB_CONDITIONAL_DISABLING1
            MSLB_STATIC_OPTIONS
            <xsl:for-each select="MSLB_FOR_EACH_SELECT1">
                <xsl:if test="normalize-space(.)!=''">
                    <span class="xdMultiSelectListItem">
                        MSLB_CHECKBOX
                        <xsl:value-of select="."/>
                    </span>
```

```
                </xsl:if>
            </xsl:for-each>
        </span>
    </xsl:when>
    <xsl:otherwise>
        <span>
            LIST_MSLB
        </span>
    </xsl:otherwise>
</xsl:choose>
```

MSLB_WITH_STATIC_CHOICES_AND_CUSTOM_ITEMS:

```
<xsl:choose>
    <xsl:when test="function-available('ipApp:GetMajorVersion') and ipApp:GetMajorVersion()
&gt;= 12">
        <span style="MSLB_STYLES1" class="xdMultiSelectList" title="ANY_STRING1"
(xd:postbackModel="POSTBACKMODEL")? xd:CtrlId="CONTROL_ID" xd:xctname="multiselectlistbox"
(xd:boundProp="value")? tabIndex="-1" xd:ref="REPEATING_LEAF_XPATH1">
            MSLB_CONDITIONAL_FORMATTING
            MSLB_CONDITIONAL_DISABLING1
            MSLB_STATIC_OPTIONS
            <xsl:for-each select="MSLB_FOR_EACH_SELECT1">
                <span class="xdMultiSelectListItem">
                    MSLB_CHECKBOX
                    <span hideFocus="1" contentEditable="true" xd:binding="."
xd:xctname="PlainText" style="WIDTH: 70%;" tabIndex="TAB_INDEX1"
(xd:inputScopeId="INPUT_SCOPE_ID") ? (xd:allowNonMatching="yes")?
class="xdMultiSelectFillIn">
                        <xsl:attribute name="title">
                            <xsl:value-of select="." />
                        </xsl:attribute>
                        MSLB_CONDITIONAL_DISABLING2
                        <xsl:value-of select="." />
                    </span>
                </span>
            </xsl:for-each>
            <xsl:if test="not(MSLB_FOR_EACH_SELECT1[1])">
                <span class="xdMultiSelectListItem">
                    MSLB_CUSTOM_VALUE
                    <span title="" hideFocus="1" xd:xctname="PlainText"
xd:binding="REPEATING_LEAF_XPATH1[.='']1]" style="WIDTH: 70%;"  tabIndex="TAB_INDEX1"
class="xdMultiSelectFillIn">
                    </span>
                </span>
            </xsl:if>
        </span>
    </xsl:when>
    <xsl:otherwise>
        <span>
            LIST_MSLB
        </span>
    </xsl:otherwise>
</xsl:choose>
```

MSLB_WITH_STATIC_CHOICES_AND_CUSTOM_ITEMS_AND_DATA_FORMAT:

```
<xsl:choose>
    <xsl:when test="function-available('ipApp:GetMajorVersion') and ipApp:GetMajorVersion()
&gt;= 12">
        <span style="MSLB_STYLES1" class="xdMultiSelectList" title="ANY_STRING1"
(xd:postbackModel="POSTBACKMODEL")? xd:CtrlId="CONTROL_ID" xd:xctname="multiselectlistbox"
tabIndex="-1" xd:ref="REPEATING_LEAF_XPATH1" xd:boundProp="xd:num"
xd:datafmt="DATA_FMT_CTRL_MSLB1">
            MSLB_CONDITIONAL_FORMATTING
            MSLB_CONDITIONAL_DISABLING1
            MSLB_STATIC_OPTIONS
            <xsl:for-each select="MSLB_FOR_EACH_SELECT1">
                <span class="xdMultiSelectListItem">
                    MSLB_CHECKBOX
                    <span hideFocus="1" contentEditable="true" xd:binding="."
xd:xctname="PlainText" style="WIDTH: 70%;" tabIndex="TAB_INDEX1" class="xdMultiSelectFillIn
xdBehavior_Formatting" (xd:inputScopeId="INPUT_SCOPE_ID")? (xd:allowNonMatching="yes")?
xd:boundProp="xd:num" xd:datafmt="DATA_FMT_CTRL_MSLB1">
                        <xsl:attribute name="xd:num">
                            <xsl:value-of select="." />
                        </xsl:attribute>
                        <xsl:attribute name="title">
                            <xsl:value-of select="." />
                        </xsl:attribute>
                        MSLB_CONDITIONAL_DISABLING2
                        MSLB_DATA_FORMATTING1
                    </span>
                </span>
            </xsl:for-each>
            <xsl:if test="not(MSLB_FOR_EACH_SELECT1[1])">
                <span class="xdMultiSelectListItem">
                    MSLB_CUSTOM_VALUE
                    <span title="" hideFocus="1" xd:xctname="PlainText"
xd:binding="REPEATING_LEAF_XPATH1[.='']" style="WIDTH: 70%;"  tabIndex="TAB_INDEX1"
class="xdMultiSelectFillIn" >
                    </span>
                </span>
            </xsl:if>
        </span>
    </xsl:when>
    <xsl:otherwise>
        <span>
            LIST_MSLB
        </span>
    </xsl:otherwise>
</xsl:choose>
```

MSLB_WITH_DYNAMIC_OPTIONS:

```
<xsl:choose>
    <xsl:when test="function-available('ipApp:GetMajorVersion') and ipApp:GetMajorVersion()
&gt;= 12">
        <span style="MSLB_STYLES1" class="xdMultiSelectList" title="ANY_STRING1"
xd:xctname="multiselectlistbox" xd:CtrlId="CONTROL_ID" xd:boundProp="value"
(xd:postbackModel="POSTBACKMODEL")? tabIndex="-1" xd:ref="REPEATING_LEAF_XPATH1">
            MSLB_CONDITIONAL_FORMATTING
            MSLB_CONDITIONAL_DISABLING
            <xsl:variable name="values" select="REPEATING_LEAF_XPATH1" />
```

```
            (MSLB_DYNAMIC_OPTIONS_GROUP | MSLB_DYNAMIC_OPTIONS_REPEATING_FIELD)
            <xsl:for-each select="REPEATING_LEAF_XPATH1[not(.=$options)] |
([PREDICATE_XPATH1])?">
                <xsl:if test="normalize-space(.)!=''">
                    <span class="xdMultiSelectListItem">
                        MSLB_CHECKBOX
                        <xsl:value-of select="." />
                    </span>
                </xsl:if>
            </xsl:for-each>
        </span>
    </xsl:when>
    <xsl:otherwise>
        <span>
            LIST_MSLB
        </span>
    </xsl:otherwise>
</xsl:choose>
```

MSLB_WITH_DYNAMIC_OPTIONS_AND_CUSTOM_OPTIONS:

```
<xsl:choose>
    <xsl:when test="function-available('ipApp:GetMajorVersion') and ipApp:GetMajorVersion()
&gt;= 12">
        <span style="MSLB_STYLES1" class="xdMultiSelectList" title="ANY_STRING1"
xd:xctname="multiselectlistbox" xd:CtrlId="CONTROL_ID" xd:boundProp="value"
(xd:postbackModel="POSTBACKMODEL")? tabIndex="-1" xd:ref="REPEATING_LEAF_XPATH1">
            MSLB_CONDITIONAL_FORMATTING
            MSLB_CONDITIONAL_DISABLING1
            <xsl:variable name="values" select="REPEATING_LEAF_XPATH1" />
            (MSLB_DYNAMIC_OPTIONS_GROUP | MSLB_DYNAMIC_OPTIONS_REPEATING_FIELD)
            <xsl:for-each select="REPEATING_LEAF_XPATH1[not(.=$options)]
|([PREDICATE_XPATH1])?">
                <span class="xdMultiSelectListItem">
                    MSLB_CHECKBOX
                    <span hideFocus="1" contentEditable="true" xd:binding="."
xd:xctname="PlainText" style="WIDTH: 70%;" tabIndex="TAB_INDEX1" class="xdMultiSelectFillIn"
(xd:inputScopeId="INPUT_SCOPE_ID")? (xd:allowNonMatching="yes")?>
                        <xsl:attribute name="title">
                            <xsl:value-of select="." />
                        </xsl:attribute>
                        MSLB_CONDITIONAL_DISABLING1
                        <xsl:value-of select="." />
                    </span>
                </span>
            </xsl:for-each>
            <xsl:if test="not(REPEATING_LEAF_XPATH1[not(.=$options)])">
                <span class="xdMultiSelectListItem">
                    MSLB_CUSTOM_VALUE
                     <span title="" hideFocus="1" xd:xctname="PlainText"
xd:binding="REPEATING_LEAF_XPATH1[.=''][1]" style="WIDTH: 70%;"  tabIndex="TAB_INDEX1"
class="xdMultiSelectFillIn" >
                    </span>
                </span>
            </xsl:if>
        </span>
    </xsl:when>
```

```
    <xsl:otherwise>
        <span>
            LIST_MSLB
        </span>
    </xsl:otherwise>
</xsl:choose>
```

MSLB_WITH_DYNAMIC_OPTIONS_AND_CUSTOM_OPTIONS_AND_DATA_FORMATTING:

```
<xsl:choose>
    <xsl:when test="function-available('ipApp:GetMajorVersion') and ipApp:GetMajorVersion()
&gt;= 12">
        <span style="MSLB_STYLES1" class="xdMultiSelectList" title="ANY_STRING1"
xd:xctname="multiselectlistbox" xd:CtrlId="CONTROL_ID" xd:boundProp="value"
(xd:postbackModel="POSTBACKMODEL")? tabIndex="-1" xd:ref="REPEATING_LEAF_XPATH1"
xd:datafmt="DATA_FMT_CTRL_MSLB1">
            MSLB_CONDITIONAL_FORMATTING
            MSLB_CONDITIONAL_DISABLING1
            <xsl:variable name="values" select="REPEATING_LEAF_XPATH1" />
            (MSLB_DYNAMIC_OPTIONS_GROUP | MSLB_DYNAMIC_OPTIONS_REPEATING_FIELD)
            <xsl:for-each select="REPEATING_LEAF_XPATH1[not(.=$options)]
|([PREDICATE_XPATH1])?">
                <span class="xdMultiSelectListItem">
                    MSLB_CHECKBOX
                    <span hideFocus="1" contentEditable="true" xd:binding="."
xd:xctname="PlainText" style="WIDTH: 70%;" tabIndex="TAB_INDEX1" class="xdMultiSelectFillIn
xdBehavior_Formatting" (xd:inputScopeId="INPUT_SCOPE_ID")? (xd:allowNonMatching="yes")?
xd:boundProp="xd:num" xd:datafmt="DATA_FMT_CTRL_MSLB1">
                        <xsl:attribute name="xd:num">
                            <xsl:value-of select="." />
                        </xsl:attribute>
                        <xsl:attribute name="title">
                            <xsl:value-of select="." />
                        </xsl:attribute>
                        MSLB_CONDITIONAL_DISABLING2
                        MSLB_DATA_FORMATTING1
                    </span>
                </span>
            </xsl:for-each>
            <xsl:if test="not(REPEATING_LEAF_XPATH1[not(.=$options)])">
                <span class="xdMultiSelectListItem">
                    MSLB_CUSTOM_VALUE
                    <span title="" hideFocus="1" xd:xctname="PlainText"
xd:binding="REPEATING_LEAF_XPATH1[.='']" style="WIDTH: 70%;"  tabIndex="TAB_INDEX1"
class="xdMultiSelectFillIn" >
                    </span>
                </span>
            </xsl:if>
        </span>
    </xsl:when>
    <xsl:otherwise>
        <span>
            LIST_MSLB
        </span>
    </xsl:otherwise>
</xsl:choose>
```

MSLB_DATA_FORMATTING:

```
<xsl:choose>
    <xsl:when test="function-available('xdFormatting:formatString')">
        <xsl:value-of select="xdFormatting:formatString(.,DATA_FMT_CTRL_MSLB1)" />
    </xsl:when>
    <xsl:otherwise>
        <xsl:value-of select="." />
    </xsl:otherwise>
</xsl:choose>
```

MSLB_CUSTOM_VALUE:

```
<input type="checkbox" title="" xd:onValue="" xd:boundProp="xd:value"
xd:binding="REPEATING_LEAF_XPATH1[.=''][1]" xd:xctname="CheckBox" tabIndex="TAB_INDEX1">
    <xsl:attribute name="xd:value">
        <xsl:value-of select="REPEATING_LEAF_XPATH1[.=''][1]" />
    </xsl:attribute>
    <xsl:if test="REPEATING_LEAF_XPATH1=''">
        <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
    </xsl:if>
</input>
```

MSLB_DYNAMIC_OPTIONS_GROUP:

```
((<xsl:variable name="items">
    <xsl:copy-of select="REPEATING_GROUP_XPATH1([PREDICATE_XPATH])?" />
</xsl:variable>
<xsl:variable name="uniqueItems" select="msxsl:node-set($items)/*[not(RELATIVE_LEAF_XPATH3 =
preceding::REPEATING_GROUP_XPATH1/RELATIVE_LEAF_XPATH3)]" />
<xsl:for-each select="$uniqueItems">)
| <xsl:for-each select="REPEATING_GROUP_XPATH1 ([PREDICATE_XPATH1])?"/>)
    <span class="xdMultiSelectListItem">
        <input type="checkbox" xd:boundProp="xd:value" xd:binding="." xd:xctname="CheckBox"
tabIndex="TAB_INDEX1">
            <xsl:attribute name="xd:value">
                <xsl:value-of select="." />
            </xsl:attribute>
            <xsl:attribute name="xd:onValue">
                <xsl:value-of select="RELATIVE_LEAF_XPATH2" />
            </xsl:attribute>
            <xsl:attribute name="title">
                <xsl:value-of select="RELATIVE_LEAF_XPATH3" />
            </xsl:attribute>
            <xsl:if test="RELATIVE_LEAF_XPATH2=$values">
                <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
            </xsl:if>
        </input>
        <xsl:value-of select="RELATIVE_LEAF_XPATH3" />
    </span>
</xsl:for-each>
<xsl:variable name="options"
select="REPEATING_GROUP_XPATH1([PREDICATE_XPATH1])?/RELATIVE_LEAF_XPATH2" /
```

MSLB_DYNAMIC_OPTIONS_REPEATING_FIELD:

```
((<xsl:variable name="items">
    <xsl:copy-of select="REPEATING_LEAF_XPATH2([PREDICATE_XPATH1])?" />
</xsl:variable>
<xsl:variable name="uniqueItems" select="msxsl:node-set($items)/*[not(.=
preceding::REPEATING_LEAF_XPATH2)]" />
<xsl:for-each select="$uniqueItems">)
| <xsl:for-each select="REPEATING_LEAF_XPATH2([PREDICATE_XPATH1])?/>)
    <span class="xdMultiSelectListItem">
        <input type="checkbox" xd:boundProp="xd:value" xd:binding="." xd:xctname="CheckBox"
tabIndex=" TAB_INDEX1">
            <xsl:attribute name="xd:value">
                <xsl:value-of select="." />
            </xsl:attribute>
            <xsl:attribute name="xd:onValue">
                <xsl:value-of select="." />
            </xsl:attribute>
            <xsl:attribute name="title">
                <xsl:value-of select="." />
            </xsl:attribute>
            <xsl:if test=".=$values">
                <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
            </xsl:if>
        </input>
        <xsl:value-of select="." />
    </span>
</xsl:for-each>
<xsl:variable name="options" select="REPEATING_LEAF_XPATH2([PREDICATE_XPATH1])?/." />
```

MSLB_STATIC_OPTIONS:

```
(<span class="xdMultiSelectListItem">
    <input type="checkbox" title="ANY_STRINGX" xd:onValue="ANY_STRINGY"
xd:boundProp="xd:value" xd:xctname="CheckBox" tabIndex="TAB_INDEX1">
        <xsl:attribute name="xd:value">
            <xsl:value-of select="REPEATING_LEAF_XPATH1[.=&amp;quot;ANY_STRINGY&amp;quot;][1]"/>
        </xsl:attribute>
        <xsl:attribute name="xd:binding">
            <xsl:value-of select="REPEATING_LEAF_XPATH1[.=&amp;quot;ANY_STRINGY&amp;quot;][1]"/>
        </xsl:attribute>
        <xsl:if test="REPEATING_LEAF_XPATH1=&amp;quot;ANY_STRINGY&amp;quot;">
            <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
        </xsl:if>
    </input>ANY_STRINGX</span>)*
```

MSLB_CHECKBOX:

```
<input type="checkbox" CHECKED="CHECKED" xd:onValue="{.}" xd:boundProp="xd:value"
xd:binding="." xd:xctname="CheckBox" tabIndex="TAB_INDEX1">
    <xsl:attribute name="xd:value">
        <xsl:value-of select="."/>
    </xsl:attribute>
    <xsl:attribute name="title">
        <xsl:value-of select="."/>
```

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

MSLB_CONDITIONAL_FORMATTING:

```
(<xsl:attribute name="style">
    MSLB_STYLES1
    <xsl:choose>
        (<xsl:when
test="BOOLEAN_XPATH_EXPRESSIONX">MSLB_CONDITIONAL_FORMATTING_STYLES</xsl:when>)*
    </xsl:choose>
</xsl:attribute>)?
```

MSLB_CONDITIONAL_DISABLING:

```
(<xsl:choose>
    (<xsl:when test="BOOLEAN_XPATH_EXPRESSIONY"/>
    |<xsl:when test="BOOLEAN_XPATH_EXPRESSIONY">
        <xsl:attribute name="disabled">true</xsl:attribute>
    </xsl:when)+
</xsl:choose>)?
```

LIST_MSLB: MUST be an HTML 4.1 fragment, as specified in [HTML], valid under the **SPAN** element that is also a valid XML 1.0 fragment, as specified in [W3C-XML]. This fragment MUST be ignored.

MSLB_STYLES: Semicolon-delimited list of (**STYLE_BORDER**?, **STYLE_PADDING**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_COLOR**?, **STYLE_MARGIN**?, **STYLE_FONT_STYLE**?, **STYLE_WIDTH**?, **STYLE_HEIGHT**?, **STYLE_FONT_FAMILY**?, **STYLE_VERTICAL_ALIGN**?, **STYLE_FONT_SIZE**?, **STYLE_FONT_WEIGHT**?, **STYLE_TEXT_DECORATION**?, **STYLE_TEXT_ALIGN**?)

MSLB_CONDITIONAL_FORMATTING_STYLES: Semicolon-delimited list of (**STYLE_DISPLAY_NONE**, **STYLE_CAPTION**) or semicolon-delimited list of (**STYLE_BACKGROUND_COLOR**?, **STYLE_COLOR**?, **STYLE_FONT_WEIGHT**?, **STYLE_FONT_STYLE**?, **STYLE_TEXT_DECORATION**?, **STYLE_CAPTION**)

The following table lists control-specific attributes used by the multiple-selection list box control.

Attribute	Section
xd:allowNonMatching	2.4.2.2
xd:binding	2.4.2.6
xd:boundProp	2.4.2.9
xd:CtrlId	2.4.2.10
xd:datafmt	2.4.2.11
xd:inputScopeId	2.4.2.21

Attribute	Section
xd:num	2.4.2.26
xd:onValue	2.4.2.28
xd:postbackModel	2.4.2.29
xd:ref	2.4.2.30
xd:value	2.4.2.34
xd:xctname	2.4.2.35

2.4.1.21.10 Picture Button Control

The picture button control is an unbound control that displays an image and executes actions (submit, query, new, and refresh), rules (1), or custom code when clicked.

Symbol	Description
PICTURE_BUTTON_RULES_AND_CUSTOM_CODE_WITH_CONDITIONAL_FORMATTING	The picture button executes rules (1) and custom code when clicked, and supports conditional formatting.
PICTURE_BUTTON_UPDATE_FORM_WITH_CONDITIONAL_FORMATTING	The picture button updates the form content when clicked, and supports **conditional formatting**.
PICTURE_BUTTON_ACTION_WITH_CONDITIONAL_FORMATTING	The picture button executes actions (submit, query, new, and refresh) when clicked, and supports conditional formatting.

PICTURE_BUTTON_ACTION_TYPE: "submit" or "query" or "new" or "refresh".

PICTURE_BUTTON_HIDE_IN_PRINT_VIEW: "true" or "false".

PICTURE_BUTTON_IMAGE: The value MUST be an image resource file, as specified in section 2.1, contained in the form template, and MUST conform to the specifications of a value for the **SRC** attribute of an **IMG** element, as specified in [HTML] section 13.2.

PICTURE_BUTTON_PRIMARY_IMAGE:

```
res://infopath.exe/picbuttonplaceholder.png | PICTURE_BUTTON_IMAGE
```

PICTURE_BUTTON_STYLE: Semicolon-delimited list of (**STYLE_WIDTH**, **STYLE_HEIGHT**, **BORDER-RIGHT**: medium none; **BORDER-TOP**: medium none; **BORDER-LEFT**: medium none; **BORDER-BOTTOM**: medium none, **STYLE_BACKGROUND_COLOR**?, **STYLE_MARGIN**?, **STYLE_DIRECTION**?)

PICTURE_BUTTON_IMAGE_STYLE: Semicolon-delimited list of (**WIDTH**: 100%, **STYLE_HEIGHT**, **POSITION**: (static|relative)).

The yield of **STYLE_HEIGHT** in the **PICTURE_BUTTON_STYLE** production MUST be the same as the yield of **STYLE_HEIGHT** in the **PICTURE_BUTTON_IMAGE_STYLE** production.

PICTURE_BUTTON_ACTION_STYLE: Semicolon-delimited list of (**BEHAVIOR**: url(#default#ActionButton) url(#default#PictureButton), **PICTURE_BUTTON_STYLE**).

PICTURE_BUTTON_CONDITIONAL_FORMATTING_STYLE: Semicolon-delimited list of (**STYLE_BACKGROUND_COLOR**?, **STYLE_CAPTION**).

PICTURE_BUTTON_CONDITIONAL_FORMATTING:

```
(<xsl:attribute name="style">PICTURE_BUTTON_STYLE<xsl:choose>
        (<xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_DISPLAY_NONE;
STYLE_CAPTION</xsl:when>|
        <xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_CAPTION</xsl:when>|
        <xsl:when
test="BOOLEAN_XPATH_EXPRESSION">PICTURE_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>)?
```

PICTURE_BUTTON_ACTION_CONDITIONAL_FORMATTING:

```
(<xsl:attribute name="style">PICTURE_BUTTON_ACTION_STYLE<xsl:choose>
        (<xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_DISPLAY_NONE;
STYLE_CAPTION</xsl:when>|
        <xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_CAPTION</xsl:when>|
        <xsl:when
test="BOOLEAN_XPATH_EXPRESSION">PICTURE_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>)?
```

PICTURE_BUTTON_UPDATE_FORM_CONDITIONAL_FORMATTING:

```
<xsl:attribute name="style">PICTURE_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;
STYLE_CAPTION</xsl:when>|
        <xsl:when test="BOOLEAN_XPATH_EXPRESSION">STYLE_CAPTION</xsl:when>|
        <xsl:when
test="BOOLEAN_XPATH_EXPRESSION">PICTURE_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>)?
```

PICTURE_BUTTON_RULES_AND_CUSTOM_CODE_WITH_CONDITIONAL_FORMATTING:

```
<button class="xdPictureButton" contentEditable="false" style="PICTURE_BUTTON_STYLE"
xd:CtrlId="CONTROL_ID" xd:xctname="PictureButton"
xd:HideInPrintView="PICTURE_BUTTON_HIDE_IN_PRINT_VIEW" (xd:auxDom="AUX_DOM_SOURCE_NAME")?
(tabIndex="TAB_INDEX")? (xd:postbackModel="BUTTON_POSTBACKMODEL")?
(accessKey="SINGLE_CHARACTER")?>
    PICTURE_BUTTON_CONDITIONAL_FORMATTING
    <img style="PICTURE_BUTTON_IMAGE_STYLE" title="ANY_STRING1" alt="ANY_STRING1"
src="PICTURE_BUTTON_PRIMARY_IMAGE" xd:HoverSRC="PICTURE_BUTTON_IMAGE" (Linked="true")?/>
</button>
```

PICTURE_BUTTON_UPDATE_FORM_WITH_CONDITIONAL_FORMATTING:

```
<button class="xdPictureButton" contentEditable="false" style="PICTURE_BUTTON_ACTION_STYLE"
xd:CtrlId="CONTROL_ID" xd:xctname="PictureButton"
xd:HideInPrintView="PICTURE_BUTTON_HIDE_IN_PRINT_VIEW" xd:action="updateForm"
(xd:auxDom="AUX_DOM_SOURCE_NAME")? (tabIndex="TAB_INDEX")?
(xd:postbackModel="BUTTON_POSTBACKMODEL")? (accessKey="SINGLE_CHARACTER")?>
    PICTURE_BUTTON_UPDATE_FORM_CONDITIONAL_FORMATTING
    <img style="PICTURE_BUTTON_IMAGE_STYLE" title="ANY_STRING1" alt="ANY_STRING1"
src="PICTURE_BUTTON_PRIMARY_IMAGE" xd:HoverSRC="PICTURE_BUTTON_IMAGE" (Linked="true")?/>
</button>
```

PICTURE_BUTTON_ACTION_WITH_CONDITIONAL_FORMATTING:

```
<button class="xdPictureButton" contentEditable="false" style="PICTURE_BUTTON_ACTION_STYLE"
xd:CtrlId="CONTROL_ID" xd:xctname="PictureButton"
xd:HideInPrintView="PICTURE_BUTTON_HIDE_IN_PRINT_VIEW" (xd:action="BUTTON_ACTION_TYPE")?
(xd:auxDom="AUX_DOM_SOURCE_NAME")? (tabIndex="TAB_INDEX")?
(xd:postbackModel="BUTTON_POSTBACKMODEL")? (accessKey="SINGLE_CHARACTER")?>
    PICTURE_BUTTON_ACTION_CONDITIONAL_FORMATTING
    <img style="PICTURE_BUTTON_IMAGE_STYLE" title="ANY_STRING1" alt="ANY_STRING1"
src="PICTURE_BUTTON_PRIMARY_IMAGE" xd:HoverSRC="PICTURE_BUTTON_IMAGE" (Linked="true")?/>
</button>
```

The following table lists control-specific attributes used by the picture button control.

Attribute	Section
xd:action	2.4.2.1
xd:auxDom	2.4.2.4
xd:CtrlId	2.4.2.10
xd:HideInPrintView	2.4.2.37.5
xd:HoverSRC	2.4.2.37.6
xd:postbackModel	2.4.2.29
xd:xctname	2.4.2.35

2.4.1.21.11 SharePoint File Attachment Control

The SharePoint file attachment control enables users to attach files to a form (1).

SHAREPOINT_FILE_ATTACHMENT:

```
<span class="xdSharePointFileAttachment" style="SHAREPOINT_FILE_ATTACHMENT_STYLE"
xd:xctname="SharePointFileAttachment" xd:CtrlId="CONTROL_ID" xd:binding="GROUP_XPATH"
xd:disableEditing="yes" title="ANY_STRING" (accessKey="SINGLE_CHARACTER")?
(tabindex="TAB_INDEX")? >
    <SPAN tabindex="0"/>
    <div>
        <xsl:for-each select="REPEATING_LEAF_XPATH">
            <div class="xdAttachItem" title="" xd:xctname="SharePointAttachItem"
xd:disableEditing="yes">
                <SPAN style="width:32px" xd:binding="." xd:disableEditing="yes"
tabindex="0"/>
                <a>
                    SHAREPOINT_FILE_ATTACHMENT_URL_LINK
                    <xsl:value-of select="xdXDocument:GetNamedNodeProperty(.,
'FileAttachURL', '')"/>
                </a>
            </div>
        </xsl:for-each>
    </div>
</span>
```

SHAREPOINT_FILE_ATTACHMENT_STYLE: Semicolon-delimited list of (**STYLE_WIDTH**, **STYLE_HEIGHT**?, **STYLE_SIZE**?, **STYLE_TEXT_DECORATION**?, **STYLE_BACKGROUND_COLOR**?, **STYLE_BORDER**?, **STYLE_FONT**?, **STYLE_COLOR**?, **STYLE_VERTICAL_ALIGN**?, **STYLE_DIRECTION**?)

SHAREPOINT_FILE_ATTACHMENT_URL_LINK:

```
<xsl:variable name="IsLocal" select="xdXDocument:GetNamedNodeProperty(., 'IsLocal', '')"/>
<xsl:if test="$IsLocal='true'">
    <xsl:attribute name="href">
        <xsl:value-of select="."/>
    </xsl:attribute>
```

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

The following table lists control-specific attributes used by the SharePoint file attachment control.

Attribute	Section
xd:binding	2.4.2.6
xd:CtrlId	2.4.2.10
xd:disableEditing	2.4.2.12
xd:xctname	2.4.2.35

The **xdXDocument:GetNamedNodeProperty** XSL function extension, as specified in section 2.4.3.9.3, is used by the SharePoint file attachment control.

2.4.1.22 Ignored Controls

XSL files SHOULD contain valid controls but MAY contain controls that are not recognized by the form server and are, therefore, ignored. The XSL fragment that maps to an ignored control is passed directly by the form server to the **user agent**. Ignored controls have no mechanism for persisting information to the form server, nor are they able to manipulate form data.

Following is a list of ignored controls and how to identify them.

Controls	Identifying Characteristic
Choice Group	**xctName** attribute = "ChoiceGroup"

2.4.1.23 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"
- "layoutregion"
- "choicegrouprepeating"
- "choicetermrepeating"
- "signatureline"

XHTML elements MUST NOT have an **xd:xctName** attribute (section 2.4.2.35) of the following form, where *clsid* is a GUID, except if it represents a contact selector control with *clsid* equal to "{{61e40d31-993d-4777-8fa0-19ca59b6d0bb}}":

- {{*clsid*}}

Every XHTML element that contains an **xctName** attribute (section 2.4.2.35) MUST be specified in the controls section of this document.

Expression Box (section 2.4.1.10) MUST NOT contain the writing-mode style in its style block.

Repeating Section (section 2.4.1.15) MUST NOT be encased in a **SPAN** HTML element instead of a **DIV** or **TABLE** element.

A control MUST NOT have an **XmlToEdit** element (section 2.2.1.2.105) with a recursive item XPATH.

Repeating Section (section 2.4.1.15) MUST NOT contain the **linkedToMaster** attribute.

A form definition (.xsf) file that contains an **editWith** element (section 2.2.1.2.89) with the **component** attribute set to "xImage" or "xReplace" MUST NOT be present.

2.4.1.24 Invalid Constructs

An XSLT file that contains an **xsl:template** element with the **mode** attribute set to the value "xd:preserve" MUST NOT be present.

2.4.2 Control-Specific Attributes

This section specifies the use of attributes in the XSLT file, as specified in section 2.4.1.

Examples of the use of the attributes specified in this section can be found in section 3.4.3.

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	Section	Description
action	2.4.2.1	The action executed when a control is clicked.
AllowMultiple	2.4.2.37.1	A **Boolean** value that specifies whether selection of more than one entity is allowed in a contact selector control.
allownonmatching	2.4.2.2	Not in use.
autoAdvance	2.4.2.3	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	2.4.2.4	The **name** (section 2.2.1.2.17) of the secondary data connection associated with the "refresh" **action** (section 2.4.2.1) of a button control.
backgroundPicture	2.4.2.5	Not in use.
binding	2.4.2.6	The XML field (3) from which the control reads and writes data.
binding_secondary	2.4.2.37.2	A secondary XML field (3) from which the control reads and writes data.

Attribute	Section	Description
bindingProperty	2.4.2.7	The name of the **ActiveX control** property used by the contact selector (section 2.4.1.7), and external item picker (section 2.4.1.21.4) controls to read and write data from and to the XML field (3) associated with the control.
bindingType	2.4.2.8	The format of the data that the control reads and writes.
boundProp	2.4.2.9	The name of the XSLT attribute that contains the XML field (3) from which the control reads and writes data.
boundPropSecondary	2.4.2.37.3	Not in use.
caption	2.4.2.37.11	The name of a formatting rule (1) applied to a control.
CtrlId	2.4.2.10	The unique identifier of a control in the form (1).
datafmt	2.4.2.11	The data formatting the control uses to display the data in the associated XML field (3) in the form (1).
datafmt2	2.4.2.37.4	The data formatting the control uses to display the data in the associated XML field (3) in the form (1).
disableEditing	2.4.2.12	A **Boolean** value that specifies whether a control is specified as read-only.
enabledProperty	2.4.2.13	Not in use.
enabledValue	2.4.2.14	Not in use.
ghosted	2.4.2.15	Not in use.
HideInPrintView	2.4.2.37.5	A **Boolean** value that specifies whether a picture button control (section 2.4.1.21.10) is hidden when a form (1) is printed or exported to HTML, PDF, or XPS.
HoverSrc	2.4.2.37.6	The image to be used when hovering over a picture button control.
ictID	2.4.2.16	Not in use.
ictVersion	2.4.2.17	Not in use.
inline	2.4.2.18	Not in use.
innerCtrl	2.4.2.19	A type of component inside the date picker control (section 2.4.1.8).
inputscope	2.4.2.20	Not in use.
inputScopeId	2.4.2.21	Not in use.
layoutText	2.4.2.22	Not in use.

Attribute	Section	Description
linkedToMaster	2.4.2.23	Not in use.
masterID	2.4.2.24	Not in use.
masterName	2.4.2.25	Not in use.
num	2.4.2.26	Indicates that the **xd:num** XSLT attribute contains the XML field (3) from which the control reads and writes data.
offValue	2.4.2.27	The XML field value for a check box control (section 2.4.1.6) when that check box control is cleared.
onValue	2.4.2.28	The XML field value for a control when that control is selected.
postbackModel	2.4.2.29	Specifies whether to trigger a postback when the XML field value displayed in the control is changed.
ref	2.4.2.30	Not in use.
SearchPeopleOnly	2.4.2.37.7	A **Boolean** value that specifies whether a contact selector control (section 2.4.1.7) is allowed to search only for people.
server	2.4.2.37.8	Not in use.
SharePointGroup	2.4.2.37.9	The group (1) that a contact selector control (section 2.4.1.7) uses to filter the results.
SignatureBlock	2.4.2.31	The name for the signed data block for a control that allows XML digital signatures (1).
SignedSectionDisplaySignatures	2.4.2.32	A Boolean value that specifies whether to display XML digital signatures inline in the form (1) for a control that allows XML digital signatures (1).
SignedSectionName	2.4.2.33	The **name** for the signed data block (section 2.2.1.2.107) for a control that allows XML digital signatures (1).
value	2.4.2.34	Indicates the **xd:value** XSLT attribute contains the XML field (3) from which the control reads and writes data.
widgetIndex	2.4.2.37.10	The tab index for the control menu of several controls.
xctname	2.4.2.35	The type of the control.
xmlToEdit	2.4.2.36	The additional editing properties of the control.

2.4.2.1 action

Button (section 2.4.1.5), picture button (section 2.4.1.21.10), repeating section (section 2.4.1.15), repeating table (section 2.4.1.16), and list (section 2.4.1.21.7) controls can contain the **action** attribute in their respective XSLT representations.

All other controls MUST NOT contain this attribute in their XSLT representations.

For button controls, the value of this attribute MUST be one of the following:

- **delete:** This action is associated with a client-only feature and MUST NOT be present.
- **new:** Creates a new record associated with an ADO (section 2.2.1.2.19) data connection (1). This action MUST be set only if the main data connection is an ADO 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 the following:

- **xCollection::insert**: Inserts the XML editing component associated with the repeating section and repeating table controls.

For list controls, the value of this attribute MUST be set to the following:

- **xTextList::insert**: Inserts the XML editing component associated with the list controls.

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

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

2.4.2.2 allownonmatching

The **allownonmatching** attribute is associated with a client-only feature and MUST be ignored by the form server.

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

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

2.4.2.3 autoAdvance

Text box controls (section 2.4.1.20) can contain the **autoAdvance** attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of this attribute MUST be 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

Button controls (section 2.4.1.5) and picture button controls (section 2.4.1.21.10) can contain this attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of this attribute MUST be equal to the **name** section 2.2.1.2.17) of a secondary data connection that exists in the form (1), and the value MUST be 1 to 255 characters in length.

If this attribute is unspecified, all of the secondary data sources that can query MUST be refreshed as part of the "refresh" **action** (section 2.4.2.1).

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

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

2.4.2.5 backgroundPicture

The **backgroundPicture** attribute is associated with a client-only feature and MUST be ignored by the form server.

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

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

2.4.2.6 binding

The following controls MUST contain the **binding** attribute in their XSLT representation:

- Check box (section 2.4.1.6)

- Combo box (section 2.4.1.21.2)
- Contact selector (section 2.4.1.7)
- Date picker (section 2.4.1.8)
- Drop-down list (section 2.4.1.9)
- Embedded picture (section 2.4.1.21.5)
- External item picker (section 2.4.1.21.4)
- File attachment (section 2.4.1.11)
- Hyperlink input (section 2.3.2.6)
- List (section 2.4.1.21.7)
- List box (section 2.3.1.9)
- Linked picture (section 2.3.2.8)
- Multiple-selection list box
- Option button (section 2.3.1.10)
- Rich text box (section 2.3.1.13)
- SharePoint file attachment (section 2.3.2.11)
- Text box (section 2.3.1.16)

The expression box control (section 2.3.1.6) can contain this attribute in its XSLT representation.

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **binding** attribute for the following controls MUST be set to a **LEAF_XPATH**, as specified in section 2.4.1.1:

- Check box
- Combo Box
- Date picker
- Drop-down list box
- Expression box
- Embedded picture
- File attachment
- List box
- Option button
- Rich text box

- Text box

The value of this attribute for the following controls MUST be set to a **LEAF_XPATH** or as a **STRING_XPATH_EXPRESSION,** as specified in section 2.4.1.1:

- Expression box
- Hyperlink input
- Linked Picture

The value of this attribute for list controls MUST be set as specified in the following table.

Value	Description
.	Specifies the **RELATIVE_LEAF_XPATH** that the control is bound to.

The value of this attribute for the multiple-selection list box control MUST be set to one of the values in the following table, as specified in section 2.4.1.1.

Value	Description
.	Specifies the **RELATIVE_LEAF_XPATH** that the control is bound to.
REPEATING_LEAF_XPATH[.=''][1]	Specifies the first instance of a **REPEATING_LEAF_XPATH** that has no value.
REPEATING_LEAF_XPATH[.="**ANY_STRING**"][1]	Specifies the first instance of a **REPEATING_LEAF_XPATH** that has a value equal to **ANY_STRING**.

The value of this attribute for the SharePoint file attachment control MUST be set to one of the values in the following table, as specified in section 2.4.1.1.

Value	Description
.	Specifies the **RELATIVE_LEAF_XPATH** that the control is bound to.
GROUP_XPATH	Specifies the **GROUP_XPATH** that the control is bound to.

For contact selector (section 2.4.1.7) and external item picker (section 2.4.1.21.4) controls, the value of the **binding** attribute MUST be set to a **GROUP_XPATH**, as specified in section 2.4.1.1.

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

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

2.4.2.7 bindingProperty

Contact selector controls (section 2.3.1.3) and external item picker controls (section 2.3.2.4) MUST contain the **bindingProperty** attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their respective XSLT representations.

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

Contact selector controls (section 2.3.1.3) and external item picker controls (section 2.3.2.4) MUST contain the **bindingType** attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their XSLT representations.

Contact selector and external item picker controls use the **bindingType** attribute to determine how to read and write data in and out of the control.

The value of this attribute MUST be the following:

- **xmlnode:** Specifies an XML node data format.

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

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

2.4.2.9 boundProp

The following controls MUST contain the **boundProp** attribute in their XSLT representation:

- Check box (section 2.3.1.2)
- Combo box (section 2.3.2.2)
- Contact selector (section 2.3.1.3)
- Drop-down list (section 2.3.1.5)
- Embedded picture (section 2.3.2.5)
- External item picker (section 2.3.2.4)
- File attachment (section 2.3.1.7)
- Hyperlink input (section 2.3.2.6)
- List box (section 2.3.1.9)
- Linked picture (section 2.3.2.8)

- Multiple-selection list box (section 2.3.2.9)
- Option button (section 2.3.1.10)

The following controls can contain the **boundProp** attribute in their XSLT representation:

- Date picker (section 2.3.1.4)
- Text box section 2.3.1.16)

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **boundProp** attribute for the multiple-selection list box control MUST be set as specified in the following table.

xctname value	Constraints	Value	Description
"CheckBox"	None	**xd:value**	Indicates the **xd:value** XSLT attribute contains the XML field (3) from which the control reads and writes data.
"PlainText"	The control supports custom values and data formatting.	**xd:num**	Indicates the **xd:num** XSLT attribute contains the XML field (3) from which the control reads and writes data.
"multiselectlistbox"	The control populates options from a data source (2).	**value**	Indicates the **value** XSLT attribute contains the XML field (3) from which the control reads and writes data.
	The options are specified by the form template designer and the control supports custom values and data formatting.	**xd:num**	Indicates the **xd:num** XSLT attribute contains the XML field (3) from which the control reads and writes data.

For all other cases, the multiple-selection list box control MUST NOT contain this attribute.

The value of the **boundProp** attribute for all other controls MUST be set as specified in the following table.

Value	Description	Controls
empty string (1)	This value is associated with a client-only feature and MUST be ignored by the form server.	Embedded picture
"href"	This value is associated with a client-only feature and MUST be ignored by the form server.	Hyperlink input
"src"	This value is associated with a client-only feature and MUST be ignored by the form server.	Linked picture
"value"	Indicates the **value** XSLT attribute contains the XML field (3) from which the control reads and writes data.	Drop-down list box List box Combo box

Value	Description	Controls
"xdInkData"	This value is associated with a client-only feature and MUST be ignored by the form server.	
"xd:inline"	Indicates that the XML field (3) from which the control reads and writes data is inline in the control's XSLT.	Contact selector External item picker File attachment
"xd:num"	Indicates that the **xd:num** XSLT attribute contains the XML field (3) from which the control reads and writes data.	Date picker Text box
"xd:value"	Indicates that the **xd:value** XSLT attribute contains the XML field (3) from which the control reads and writes data.	Check box Option button

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

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

2.4.2.10 CtrlId

The following controls MUST contain the **CtrlId** attribute in their XSLT representation:

- Button (section 2.4.1.5)
- Check box (section 2.4.1.6)
- Choice Section (section 2.4.1.21.1)
- Combo Box (section 2.4.1.21.2)
- Contact selector (section 2.4.1.7)
- Date picker (section 2.3.1.4)
- Drop-down list box (section 2.4.1.9)
- Embedded Picture (section 2.4.1.21.5)
- External item picker (section 2.4.1.21.4)
- Expression box (section 2.4.1.10)
- File attachment (section 2.4.1.11)
- Hyperlink input (section 2.4.1.21.6)
- List (section 2.4.1.21.7)
- List box (section 2.4.1.13)
- Linked picture (section 2.4.1.21.8)

- Multiple-selection list box (section 2.4.1.21.9)
- Option button (section 2.4.1.14)
- Optional section (section 2.4.1.18)
- Picture button (section 2.4.1.21.10)
- 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)
- SharePoint file attachment (section 2.4.1.21.11)
- Text box (section 2.4.1.20)

Hyperlink controls (section 2.4.1.12) can contain this attribute in their XSLT representation.

The value of the **CrtlId** 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 and picture 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

The following controls can contain the **datafmt** attribute in their XSLT representation:

- Combo box (section 2.3.2.2)
- Date picker (section 2.3.1.4)
- Expression box (section 2.3.1.6)
- Multiple-selection list box (section 2.3.2.9)
- Text box (section 2.3.1.16)

All other controls MUST NOT contain this attribute in their XSLT representations.

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 (3) be formatted as type **currency** when displayed in the corresponding control.
- **date:** Specifies that the data from the XML field (3) be formatted as type **date** when displayed in the corresponding control.
- **datetime:** Specifies that the data from the XML field (3) be formatted as type **datetime** when displayed in the corresponding control.
- **number:** Specifies that the data from the XML field (3) be formatted as type **number** when displayed in the corresponding control.
- **percentage:** Specifies that the data from the XML field (3) be formatted as type **percentage** when displayed in the corresponding control.
- **string:** Specifies that the data from the XML field (3) be formatted as type **string** when displayed in the corresponding control.
- **time:** Specifies that the data from the XML field (3) 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 defines which format specifications MUST be specified for each format category.

FormatCategory	Supported FormatSpecification
currency	**locale, numDigits, grouping, decimalSep, thousandSep, negativeOrder, positiveOrder, currencyLocale**
date	**locale, dateFormat, useAltCalendar, englishStringsAlways**
datetime	**locale, dateFormat, timeFormat, useAltCalendar, englishStringsAlways, noSeconds**
number	**locale, numDigits, grouping, decimalSep, thousandSep, negativeOrder**
percentage	**locale, numDigits, grouping, decimalSep, thousandSep, negativeOrder**
string	**plainMultiline**
time	**locale, timeFormat, noSeconds**

FormatSpecification MUST be a semicolon-delimited list of one or more specification items. Each specification item MUST have the proper structure according to the following table.

Name of specification item	Structure of specification item
plainMultiline	*"itemName"*
Any other name	*"itemName:itemValue"*

The format specification item names MUST be set to the following and adhere to the specified value requirements:

- **currencyLocale:** The LCID, as specified in [MS-LCID]. Used to map the currency symbol **string** determined by the currency locale.
- **dateFormat:** The date format pattern to use when displaying date and time values.
  - The value MUST be one of the following:
    - **Short Date**: A short date format, as specified in [MC-NLSIP].
    - **Long Date**: A long date format, as specified in [MC-NLSIP].
    - **Year Month**: A year month format, as specified in [MC-NLSIP].
    - **none**: No date format pattern applied.
    - A date format pattern specified in [ISO-8601].
- **decimalSep:** The **string** to display as the decimal separator in numeric values.
  - The value MUST be one of the following:
    - '.' (period)
    - ',' (comma)
    - ' ' (space)
    - ' ' (nonbreaking space)
    - ''' (single quote)
    - '٫ ' (Arabic comma)
    - Empty (no separator)
  - If unspecified, the decimal separator is determined based on the form server settings.
- **englishStringsAlways:** A **Boolean** value that specifies the ability to always use English as the LCID, as specified in [MS-LCID], when displaying date and date time values.
  - The value MUST be one of the following:
    - "0": Use the locale specified by the LCID.
    - "1": Use English as the LCID.
  - If unspecified, the behavior MUST be the same as a specification item value of zero ("0").
- **grouping:** The digit grouping pattern for digits to the left of the decimal.
  - The value MUST be one of the following:
    - Range from zero ("0"), meaning no grouping, through "9".
    - "32"

- If unspecified, the grouping is determined based on the form server settings.

- **locale**<4>**:** The LCID, as specified in [MS-LCID]. If unspecified, the locale is determined based on the locale of the protocol server.
- **negativeOrder:** The format pattern for negative numeric values.
  - If the value is "-1", the negative order is determined by the default pattern associated with the form server's locale identifier, as specified in [MS-LCID].
  - If unspecified, the negative order is determined based on the form server settings.
- **noSeconds:** A **Boolean** value that specifies whether to display seconds in time formatting.
  - The value MUST be set to one of the following:
    - "0": Display seconds for time formatting.
    - "1": Do not display seconds for time formatting.
  - If unspecified, the behavior MUST be the same as a specification item value of zero ("0").
- **numDigits:** The number of fractional digits to display after the decimal separator.
  - The value MUST be set to one of the following:
    - Range from zero ("0") through "9".
    - "Auto": Specifies the general numeric format string as implemented by the form server.
  - If 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 MUST be set to an empty string (1).
- **positiveOrder:** The format pattern for positive currency values.
  - If the value is "-1", the positive order is determined by the default pattern associated with the form server's locale identifier, as specified in [MS-LCID].
  - If unspecified, the positive order is determined based on the form server settings.
- **thousandSep:** The **string** that separates groups of digits to the left of the decimal in numerical values.
  - The value MUST be one of the following:
    - '.' (period)
    - ',' (comma)
    - ' ' (space)
    - ' ' (nonbreaking space)
    - ''' (single quote)
    - '٬' (Arabic comma)

        - Empty (no separator)
    - If unspecified, the thousand separator is determined based on the form server settings.
- **timeFormat:** The time format pattern to use when displaying time values.
    - The value MUST be one of the following:
        - **Short Time**: A short time format, as specified in [MC-NLSIP].
        - **Long Time**: A long time format, as specified in [MC-NLSIP].
        - **none**: No time format pattern applied.
        - A time format pattern specified in [ISO-8601].
    - If 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-WSSFO2] section 2.2.3.3.
    - The value MUST be one of the following:
        - "0": Use the calendar type associated with the locale identifier, as specified in [MS-LCID].
        - "1": Use the alternate calendar type for the locale identifier, as specified in [MS-LCID].
    - If unspecified, the behavior MUST be the same as a specification item value of zero ("0").

If the **format** category is "datetime", at least one of the specification items **timeFormat** and **dateFormat** MUST have the value "none".

The combinations of locale and date and time format in the following table 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"

Locale	Date and time format
1052	"h:mm:ss.tt"
1053	"'den 'd MMMM"
1054	"MMMM dd"
1062	"d. MMMM"
1063	"MMMM d 'd.'"
1066	"dd MMMM yyyy"
1066	"dd MMMM"
1066	"MMMM yyyy"
1069	"MMMM dd"
1078	"dd MMMM"
1079	"yyyy '\u10EC\u10DA\u10D8\u10E1' dd MM, dddd"
1104	"d MMMM"
1125	"MMMM dd"
2052	"M'\u6708'd'\u65E5'"
2060	"dd-MMM-yy"
2060	"H' h 'mm"
2060	"H' h 'm' min '"
2060	"H' h 'm' min 's' s '"
2067	"H.mm' u.'"
2070	"HH'H'mm'm'"
3079	"d.MMMyyyy"
3082	"HH'H'mm'\'"
3084	"d MMMM"
3084	"H' h 'mm"
5132	"HH' h 'mm"
6156	"HH' h 'mm"
7177	"dd MMMM"

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

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

2.4.2.12 disableEditing

The following controls can contain the **disableEditing** attribute in their XSLT representation:

- Contact selector (section 2.4.1.7)
- Embedded picture (section 2.4.1.21.5)
- External item picker (section 2.4.1.21.4)
- File attachment (section 2.4.1.11)
- Hyperlink input (section 2.4.1.21.6)
- List (section 2.4.1.21.7)
- Linked picture (section 2.4.1.21.8)
- Rich text box (section 2.4.1.17)
- Text box (section 2.4.1.20)

The following controls MUST contain the **disableEditing** attribute in their XSLT representation:

- Expression box (section 2.4.1.10)
- Hyperlink (section 2.3.1.8)
- Section and optional section (section 2.4.1.18) with XML digital signatures (1)
- SharePoint file attachment (section 2.4.1.21.11)

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **disableEditing** attribute MUST be 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

All controls MUST NOT contain the **enabledProperty** attribute in their XSLT representations.

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

All controls MUST NOT contain the **enabledValue** attribute in their XSLT representations.

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

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

2.4.2.15 ghosted

The **ghosted** attribute is associated with a client-only feature and MUST be ignored by the form server.

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

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

2.4.2.16 ictID

The **ictID** attribute is associated with a client-only feature and MUST be ignored by the form server.

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

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

2.4.2.17 ictVersion

The **ictVersion** attribute is associated with a client-only feature and MUST be ignored by the form server.

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

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

2.4.2.18 inline

The **inline** attribute is associated with a client-only feature and MUST be ignored by the form server.

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

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

2.4.2.19 innerCtrl

Date picker controls (section 2.4.1.8) MUST contain the **innerCtrl** attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their XSLT representations.

The **innerCtrl** attribute MUST be set to one of the following values, which specify the respective component types:

- **_DTButton:** A date picker button.
- **_DTText:** An editable text field (3).

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

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

2.4.2.20 inputscope

The **inputscope** attribute is associated with a client-only feature and MUST be ignored by the form server.

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

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

2.4.2.21 inputScopeId

The **inputScopeId** attribute is associated with a client-only feature and MUST be ignored by the form server.

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

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

2.4.2.22 layoutText

The **layoutText** attribute is associated with a client-only feature and MUST be ignored by the form server.

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

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

2.4.2.23 linkedToMaster

The **linkedToMaster** attribute is associated with a client-only feature and MUST NOT be present.

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

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

2.4.2.24 masterID

The **masterID** attribute is associated with a client-only feature and MUST be ignored by the form server.

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

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

2.4.2.25 masterName

The **masterName** attribute is associated with a client-only feature and MUST NOT be present.

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

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

2.4.2.26 num

The following controls can contain the **num** attribute in their XSLT representation:

- Combo box (section 2.4.1.21.2)
- Date picker (section 2.4.1.8)
- Expression box (section 2.4.1.10)
- Multiple-selection list box (section 2.4.1.21.9)
- Text box (section 2.4.1.20)

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of this attribute MUST be set as specified in section 2.4.2.9.

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

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

2.4.2.27 offValue

Check box controls (section 2.4.1.6) MUST contain the **offValue** attribute in their XSLT representation if the **onValue** attribute (section 2.4.2.28) is unspecified.

Check box controls can contain the **offValue** attribute in their XSLT representation if the **onValue** attribute is specified.

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **offValue** attribute MUST be a value that 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

Check box controls (section 2.4.1.6) MUST contain the **onValue** attribute in their XSLT representation if the **offValue** attribute (section 2.4.2.27) is unspecified.

Check box controls can contain the **onValue** attribute in their XSLT representation if the **offValue** attribute is specified.

The following controls can contain this attribute in their XSLT representation:

- Multiple-selection list box (section 2.4.1.21.9)
- Option button (section 2.4.1.14)

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of this attribute MUST be a value that is valid for the XML field's data type.

For the check box control, the value of this attribute MUST be a different **string** value than the **string** value specified for the **offValue** attribute.

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

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

2.4.2.29 postbackModel

The following controls can contain the **postbackModel** attribute in their XSLT representation:

- Button (section 2.4.1.5)
- Check box (section 2.4.1.6)
- Choice Section (section 2.4.1.21.1)
- Combo box (section 2.4.1.21.2)
- Date picker (section 2.4.1.8)
- Drop-down list box (section 2.4.1.9)
- Hyperlink input (section 2.4.1.21.6)
- List (section 2.4.1.21.7)
- List box (section 2.4.1.13)
- Linked picture (section 2.4.1.21.8)
- Multiple-selection list box (section 2.4.1.21.9)
- Option button (section 2.4.1.14)
- Optional section (section 2.4.1.18)
- Picture button (section 2.4.1.21.10)
- Repeating section (section 2.4.1.15)
- Repeating table (section 2.4.1.16)
- Rich text box (section 2.4.1.17)
- Section (section 2.4.1.18)
- Text box (section 2.4.1.20)

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of this attribute MUST be one of the following:

- **always:** Always send data to the form server when the XML field value in the control is changed.
- **auto:** Dependant 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 the **postbackModel** attribute is unspecified, the behavior MUST be the same as an attribute value of "auto".

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

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

2.4.2.30 ref

The following controls can contain the **ref** attribute in their **XSLT** representation:

- Choice group (section 2.4.1.21.1)
- Multiple-selection list box (section 2.4.1.21.9)

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **ref** attribute for the multiple-selection list box control MUST be set to a **LEAF_XPATH**, as specified in section 2.4.1.1.

The value of the **ref** attribute for the choice group control MUST be set to a **GROUP_XPATH,** as specified in section 2.4.1.1.

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

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

2.4.2.31 SignatureBlock

Section and optional section controls (section 2.4.1.18) can contain the **SignatureBlock** attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **SignatureBlock** attribute MUST be equal to the signed data block name (section 2.2.1.2.107) that exists in the form (1).

The value of this attribute MUST be equal to the value specified for the **SignedSectionName** attribute (section 2.4.2.33).

The value of this attribute MUST begin with an alphabetic or underscore character, and MUST contain only alphanumeric, underscore, hyphen, and period characters.

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

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

2.4.2.32 SignedSectionDisplaySignatures

Section and optional section controls (section 2.4.1.18) can contain the **SignedSectionDisplaySignatures** attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **SignedSectionDisplaySignatures** 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

Section and optional section controls (section 2.4.1.18) can contain the **SignedSectionName** attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **SignedSectionName** attribute MUST be equal to the **signedDataBlock name** (section 2.2.1.2.107) that exists in the form (1).

The value of the **SignedSectionName** attribute MUST be equal to the value specified for the **SignatureBlock** attribute (section 2.4.2.31).

The value of this attribute MUST begin with an alphabetic or underscore character, and MUST contain only alphanumeric, underscore, hyphen, and period characters.

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

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

2.4.2.34 value

The following controls MUST contain the **value** attribute in their XSLT representation:

- Check box (section 2.4.1.6)
- Multiple-selection list box (section 2.4.1.21.9)
- Option button (section 2.4.1.14)

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **value** attribute MUST be set as specified in the **boundProp** attribute (section 2.4.2.9).

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

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

2.4.2.35 xctname

All controls MUST contain the **xctname** attribute in their XSLT representation.

The value of this attribute MUST be set to one included in the following table for **built-in control**.

Control	xctname value
Bulleted list (section 2.4.1.21.7)	"BulletedList" "bulletedlist"
Button (section 2.4.1.5)	"Button"
Check box (section 3.4.1.1)	"CheckBox"
Choice Group (section 2.4.1.21.1)	"Choice Group: choicegroup" to specify a container for choice sections. "Choice Section: choiceterm"
Combo box (section 2.4.1.21.2)	"combobox"
Contact selector (section 2.4.1.7)	"{{61e40d31-993d-4777-8fa0-19ca59b6d0bb}}"
Date picker (section 2.4.1.8)	"DTPicker" to specify the date picker control. "DTPicker_DTButton" to specify the date picker button in the date picker control. "DTPicker_DTText" to specify the editable text field (3) in the date picker control.
Drop-down list box (section 2.4.1.9)	"DropDown"
Embedded picture (section 2.4.1.21.5)	"InlineImage"
External Item Picker (section 2.4.1.21.4)	"entitypicker"
Expression box (section 2.4.1.12)	"ExpressionBox"
File attachment (section 2.4.1.11)	"FileAttachment"
Hyperlink (section 2.4.1.12)	"Hyperlink"
Hyperlink input (section 2.4.1.21.6)	"HyperlinkBox"
List box (section 2.4.1.13)	"ListBox"
Linked picture (section 2.4.1.21.8)	"LinkedImage"
Multiple-selection list box (section 2.4.1.21.9)	"multiselectlistbox" to specify the multiple-selection list box control. "PlainText" to specify a custom value text box in the multiple-selection list box control. "CheckBox" to specify a check box in the multiple-selection list box control.
Numbered list (section 2.4.1.21.7)	"NumberedList" "numberedlist"

Control	xctname value
Option button (section 3.4.1.10)	"OptionButton"
Optional section (section 2.4.1.18)	"Section"
Picture button (section 2.4.1.21.10)	"PictureButton"
Plain list (section 2.4.1.21.7)	"PlainList" "plainlist"
Repeating section (section 2.4.1.15)	"RepeatingSection"
Repeating table (section 2.4.1.16)	"RepeatingTable"
Rich text box (section 2.4.1.17)	"RichText"
Section (section 3.4.1.14)	"Section"
SharePoint file attachment (section 2.4.1.21.11)	"SharePointFileAttachment" to specify the SharePoint file attachment control. "SharePointAttachItem" to specify the individual attached file in the SharePoint file attachment control.
Text box (section 2.3.1.16)	"PlainText"

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

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

2.4.2.36 xmlToEdit

The following controls can contain the **xmlToEdit** attribute in their XSLT representation:

- Optional section (section 2.4.1.18)
- Repeating section (section 2.4.1.15)
- Repeating table (section 2.4.1.16)
- Section (section 2.4.1.18)

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **xmlToEdit** attribute MUST be equal to the name of an **xmlToEdit** element (section 2.2.1.2.105) 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"/>
```

2.4.2.37 Control-Specific Attributes Introduced in Version 2 of the Structure Specification

2.4.2.37.1 AllowMultiple

Contact selector controls (section 2.4.1.7) can contain the **AllowMultiple** attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **AllowMultiple** attribute MUST be one of the following:

- **true:** Allow the selection of more than one entity in the contact selector control.
- **false:** Allow the selection of, at most, one entity in the contact selector control.

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

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

2.4.2.37.2 binding_secondary

The following controls can contain the **binding_secondary** attribute in their XSLT representation:

- Hyperlink input (section 2.4.1.21.6)
- Linked picture (section 2.4.1.21.8)

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **binding_secondary** attribute for the following controls MUST be a **LEAF_XPATH,** as specified in section 2.4.1.1:

- Hyperlink input
- Linked picture

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

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

2.4.2.37.3 boundPropSecondary

The following controls can contain the **boundPropSecondary** attribute in their XSLT representation:

- Hyperlink input (section 2.4.1.21.6)

- Linked picture (section 2.4.1.21.8)

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **boundPropSecondary** attribute MUST be set as specified in the following table.

Value	Description	Controls
"displaytext"	This value is associated with a client-only feature and MUST be ignored by the form server.	Hyperlink input Linked picture

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

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

2.4.2.37.4 datafmt2

The following controls can contain the **datafmt2** attribute in their XSLT representation:

- Date picker (section 2.4.1.8)
- Expression box (section 2.4.1.10)
- Text box (section 2.4.1.20)

If the control contains the **datafmt2** attribute, it MUST contain the **datafmt** attribute (section 2.4.2.11).

All other controls MUST NOT contain this attribute in their XSLT representations.

The **datafmt2** attribute extends **datafmt** for specifying additional format options. The value of this attribute MUST be a semicolon-delimited list of zero or more specification items and each specification item MUST follow this structure:

- *"itemName:itemValue"*

The format specification item names MUST be set to the following and adhere to the specified value requirements:

- **calendar:** The calendar type to use when displaying date values. This can be used with one of the following format categories specified in **datafmt**:
  - **date**
  - **datetime**
- All other format categories specified in **datafmt** MUST NOT use this format specification item.
- The value of this specification MUST be set to one of the following:
  - A calendar identifier specified in [MSDN-CALID].

- **-1:** No calendar type specified.

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

```
<xsd:element name="datafmt2" type="xsd:string"/>
```

2.4.2.37.5 HideInPrintView

Picture button controls (section 2.4.1.21.10) MUST contain the **HideInPrintView** attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **HideInPrintView** attribute MUST be one of the following:

- **true:** Always hide the picture button control when a form (1) is printed or exported to HTML, PDF, or XPS.
- **false:** Always show the picture button control when a form (1) is printed or exported to HTML, PDF, or XPS.

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

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

2.4.2.37.6 HoverSrc

Picture button controls (section 2.4.1.21.10) MUST contain the **HoverSrc** attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **HoverSrc** attribute MUST be one of the following:

- An image resource file (section 2.1) contained in the form template. The value MUST conform to the specifications of a value for the **SRC** attribute of an **IMG** element, as specified in [HTML] section 13.2.
- An empty string (1).

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

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

2.4.2.37.7 SearchPeopleOnly

Contact selector controls (section 2.3.1.3) can contain the **SearchPeopleOnly** attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **SearchPeopleOnly** attribute MUST be set to one of the following:

- **true:** Corresponds to the "PeopleOnly" value of **UserSelectionMode**, as specified in [MS-WSSFO2] section 2.2.8.3.3.1.
- **false:** Corresponds to the "PeopleAndGroups" value of **UserSelectionMode**, as specified in [MS-WSSFO2] section 2.2.8.3.3.1.

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

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

2.4.2.37.8 server

The **server** 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="server" type="xsd:string"/>
```

2.4.2.37.9 SharePointGroup

Contact selector controls (section 2.3.1.3) and external item picker controls (section 2.3.2.4) can contain the **SharePointGroup** attribute in their XSLT representation.

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **SharePointGroup** attribute MUST a value specified by **UserSelectionScope**, as specified in [MS-WSSFO2] section 2.2.8.3.3.1.

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

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

2.4.2.37.10 widgetIndex

The following controls can contain the **widgetIndex** attribute in their XSLT representation:

- Choice section (section 2.4.1.21.1)
- Optional section (section 2.4.1.18)
- Repeating section (section 2.4.1.15)
- Repeating table (section 2.4.1.16)

- Section (section 2.4.1.18)

All other controls MUST NOT contain this attribute in their XSLT representations.

The value of the **widgetIndex** attribute MUST be set to one of the following:

- **-1:** The control menu is not tab stoppable.
- The value MUST<5> conform to the specifications of a value for the **tabindex** attribute, as specified in [HTML] section 17.11.1.

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

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

2.4.2.37.11 caption_

The following controls can contain multiple instances of the **caption_** attribute in their XSLT representation:

- Repeating (section 2.3.1.11)
- Repeating table (section 2.3.1.12)
- Section (section 2.3.1.14)
- Optional section (section 2.3.1.14)
- Choice section (section 2.3.2.1)

All other controls MUST NOT contain this attribute in their XSLT representations.

An **integer** MUST be appended to the end of the name of the **caption_** attribute. The first instance of this attribute, if it exists, MUST be named "**xd:caption_0**", and each subsequent instance of this attribute MUST have the next **integer**.

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

```
<xsd:element name="caption_" 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	Section	Description
msxsl	2.4.3.1	A string comparison function.
xdDate	2.4.3.2	A set of date- and time-related functions.

Namespace	Section	Description
xdEnvironment	2.4.3.3	A set of functions that are related to the environment in which the form (1) is being filled out.
xdFormatting	2.4.3.4	A set of string formatting functions.
xdImage	2.4.3.5	A set of image-related functions.
xdMath	2.4.3.6	A set of mathematical functions.
xdUser	2.4.3.7	A set of functions that are related to the user who is filling out the form (1).
xdUtil	2.4.3.8	Generic helper tool functions.
xdXDocument	2.4.3.9	A set of functions that are related to the data of the form (1) being filled out.
xdServerInfo	2.4.3.10	A set of functions that are related to the server where the form template is located.
ipApp	2.4.3.11	A set of functions related to the application.

2.4.3.1 msxsl

Microsoft XPath Extensions, as described in [MSDN-XPATH], include a number of functions, one of which is supported by msxsl.

2.4.3.1.1 string-compare

Function signature: **number** msxsl:string-compare(*First String, Second String*)

The **string-compare** 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 zero if they are equivalent **strings**, "1" if the second **string** comes before the first in lexicographical order, and "-1" if the first **string** comes before the second in lexicographical order. **Number** is specified in [XPATH] section 3.5.

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*)

The **AddDays** function MUST take two parameters:

- **date:** MUST be either a **string** or an XPath expression, as specified in [XPATH], which returns a node. The **string** or the value of the node MUST be a date in ISO 8601 format, as specified in [ISO-8601], to be a valid parameter.
- **days:** 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 "#ERR?" if either of the parameters have a value that is not valid. **String** is specified in [XPATH] section 3.6.

2.4.3.2.2 AddSeconds

Function Signature: **string** xdDate:addSeconds(*time, seconds*)

The **AddSeconds** function MUST take two parameters:

- **Time**: 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.
- **Seconds**: 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 "#ERR?" if either of the parameters have a value that is not valid. **String** is specified in [XPATH] section 3.6.

2.4.3.2.3 Now

Function Signature: **string** xdDate:now()

The **Now** 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()

The **Today** 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 in [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()

The **IsBrowser** 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()

The **IsMobile** 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*)

The **avg** function MUST take one parameter:

- **XPath Expression**: MUST be an XPath expression that returns a **node-set**. XPath expression is specified in [XPATH]. **Node-set** is specified in [XPATH] section 3.3.

The **string** value of every node in the **node-set** MUST be calculated using the **string** function specified in [XPATH] section 4.2. The output of the **string** function MUST be converted to a **number** using the **number** function specified in [XPATH] section 4.4. If the **number** function returns "NAN" for any node of the **node-set**, the **avg** function MUST return "**NAN**". The output of the **number** function MUST be used as the numerical value of the node.

The **avg** function MUST return the average value of all of the numerical values in the given **node-set**. **Number** and **NAN** are specified in [XPATH] section 3.5.

2.4.3.6.2 Eval

Function Signature: **node-set** xdMath:eval(*XPath Expression, XSLT Expression*)

This function MUST take two parameters:

- **XPath Expression**: MUST be an XPath expression that returns a **node-set**. XPath expression is specified in [XPATH]. **Node-set** is specified in [XPATH] section 3.3.
- **XSLT Expression**: MUST be a valid XSLT expression, as specified in [W3C-XSLT] section 4.

The function MUST apply the XSLT expression to every node in the **node-set** and return the resulting **node-set**.

2.4.3.6.3 Max

Function Signature: **number** xdMath:max(*XPath Expression*)

The **max** function MUST take one parameter:

- **XPath Expression**: MUST be an XPath expression that returns a **node-set**. XPath expression is specified in [XPATH]. **Node-set** is specified in [XPATH] section 3.3.

The **string** value of every node of the **node-set** MUST be calculated using the **string** function specified in [XPATH] section 4.2. The output of the **string** function MUST be converted to a **number** using the **number** function specified in [XPATH] section 4.4. If the **number** function returns "NAN" for any node of the **node-set**, the **max** function MUST return "NAN". The output of the **number** function MUST be used as the numerical value of the node.

The **max** function MUST return the numerical value that is greater than or equal to the value of every other item in the node set. **Number** and **NAN** are specified in [XPATH] section 3.5.

2.4.3.6.4 Min

Function Signature: **number** xdMath:min(*XPath Expression*)

The **min** function MUST take one parameter:

- **XPath Expression:** This parameter MUST be an XPath expression that returns a **node-set**. XPath expression is specified in [XPATH]. **Node-set** is specified in [XPATH] section 3.3.

The **string** value of every node of the **node-set** MUST be calculated using the **string** function specified in [XPATH] section 4.2. The output of the **string** function MUST be converted to a **number** using the **number** function specified in [XPATH] section 4.4. If the **number** function returns "NAN" for any node of the **node-set**, the **min** function MUST return "NAN". The output of the **number** function MUST be used as the numerical value of the node.

The **min** function MUST return the numerical value that is smaller than or equal to the value of every other item in the given node set. **Number** and **NAN** are specified in [XPATH] section 3.5.

2.4.3.6.5 Nz

Function Signature: **node-set** xdMath:nz(*XPath Expression*)

This function MUST take the following parameter:

- **XPath Expression:** This parameter MUST be an XPath expression that returns a **node-set**. XPath expression is specified in [XPATH]. **Node-set** is specified in [XPATH] section 3.3.

The function MUST return a **node-set** that is identical to the given **node-set** with the exception that empty nodes are given the value zero ("0").

2.4.3.7 xdUser

xdUser contains a set of functions that are related to the user who is filling out the form (1).

2.4.3.7.1 get-UserName

Function Signature: **string** xdUser:get-UserName()

The **get-UserName** 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 utility functions.

2.4.3.8.1 Match

Function Signature: **boolean** xdUtil:match(*string, Regular Expression*)

This function MUST take the following two parameters:

- **String:** This parameter MUST be a **string**.
- **Regular Expression:** This 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 that 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()

The **get-dom** function MUST NOT take any parameters. It MUST return a **node-set** that contains the main data source.

2.4.3.9.2 getDOM

Function Signature: **node-set** xdXDocument:getDOM(*Name*)

This function MUST take one parameter:

- **Name:** This parameter MUST be a **string**. It MUST be the name of a secondary data source (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 three parameters:

- **MainDOMNode**: 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**: MUST be a **string**. It specifies the name of the property whose value is to be returned.
- **DefaultValue**: MUST be a **string**. It specifies the default value to be returned if the property has not been set.

This function provides a mechanism to retrieve **string** data that is stored on the non-attribute nodes of the main data source. The protocol only defines a mechanism to retrieve the data. The protocol server SHOULD provide a mechanism to store **string** data.

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.

2.4.3.10 xdServerInfo

xdServerInfo contains a set of functions that are related to the server where the form template is located.

2.4.3.10.1 get-SharePointListUrl

Function Signature: **string** xdServerInfo:get-SharePointListUrl()

The **get-SharePointListUrl** function MUST NOT take any parameters. It MUST return the URL of the list (1) where the form template is hosted. If the form template is not located within a list (1), it MUST return an empty string (1).

2.4.3.10.2 get-SharePointSiteUrl

Function Signature: **string** xdServerInfo:get-SharePointSiteUrl()

The **get-SharePointSiteUrl** function MUST NOT take any parameters. It MUST return the URL of the site where the form template is hosted.

2.4.3.10.3 get-SharePointServerRootUrl

Function Signature: **string** xdServerInfo:get-SharePointServerRootUrl()

This function MUST NOT take any parameters. It MUST return the URL of the root of the server where the form template is hosted.

2.4.3.10.4 get-SharePointSiteCollectionUrl

Function Signature: **string** xdServerInfo:get-SharePointSiteCollectionUrl()

The **get-SharePointSiteCollectionUrl** function MUST NOT take any parameters. It MUST return the URL of the site collection where the form template is hosted.

2.4.3.11 ipApp

ipApp contains a set of application related functions. It is not supported and MUST be ignored.

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 a **view** element (section 2.2.1.2.104) in the form definition (.xsf) file. See section 3.5 for an example.

2.6 Submit Files (XML) Specification

For each **input** element (section 2.2.1.2.24) inside of a **webServiceAdapter** element (section 2.2.1.2.20) 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". with the number increasing by one for each additional file. Each of these files MUST be referenced at the **input** element inside the form definition (.xsf) file. All of these files MUST be contained inside the form template.

Each Submit.xml file MUST contain only the following types:

- **myFields**
- **dataFields**, including the Web service method template specified in section 2.6.2.

2.6.1 myFields

The **myFields** element MUST be the top-level element in this XML file. It MUST do the following:

- Specify 'xmlns:dfs="http://schemas.microsoft.com/office/infopath/2003/dataFormSolution"' as a namespace.
- Specify any additional namespaces required for the Web service method template specified in section 2.6.2.
- Have a single child **dataFields** element.

Child Elements
dataFields

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

```
<xs:element name="myFields" type="dfs:MyFieldType"/>
<xs:complexType name="MyFieldType">
  <xs:sequence>
    <xs:element ref="dfs:dataFields" minOccurs="1" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>
```

2.6.2 dataFields

The **dataFields** element MUST have exactly one child node that is the Web service method template. This template specifies the method and parameter field 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
myFields

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

```
<xs:element name="dataFields" type="dfs:DataFieldType"/>
<xs:complexType name="DataFieldType">
  <xs:sequence>
    <!-- Web Service Template -->
    <xs:any processContents="skip" minOccurs="1" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>
```

2.7 Template.XML Specification

The template.xml file MUST be an instance of the XML schema document, as specified in section 2.3, and MUST be a valid form file, as specified in [MS-IPFFX] section 2.1.

Initial values for the fields (3) in a new form file MUST be stored in and loaded from this file. If a pre-existing form file is opened, contents of this file MUST be ignored.

2.8 Upgrade.XSL Specification

The upgrade.xsl file is an XSLT that MUST conform to the XSLT specification 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 section 2.8.1.

2.8.1 MSXSL:Node-Set()

The **msxsl:node-set** function converts a result tree fragment into an XML node set. The resulting XML node set always contains a single XML node and is the root XML node of the tree. It MUST take one argument, *$var*, which is the result tree fragment to be converted.

3 Structure Examples

The section contains examples for the following structures:

- The InfoPath form template format (section 3.1)
- Form definition files (XSF) (section 3.2)
- XML schema files (XSD) (section 3.3)
- Form view files (XSL) (section 3.4)
- Print view files (XSLT) (section 3.5)
- Submit files (XML) (section 3.6)
- Template.XML (section 3.7)
- Upgrade.XSL (section 3.8)

3.1 The InfoPath Form Template Format

3.1.1 Simple Form Template

The following example describes a simple form template (.xsn) file.

Simple.xsn

- manifest.xsf
- myschema.xsd
- template.xml
- sampledata.xml
- view1.xsl

The following five 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. At least one such file is required. It represents how the form is displayed, including which fields (3) appear and in what order.

3.1.2 Complex Form Template

The following example describes a slightly more complex form template (.xsn) file than the one contained in the previous section.

Complex.xsn

- manifest.xsf
- myschema.xsd
- template.xml
- sampledata.xml
- view1.xsl
- view2.xsl
- IPTemplate_bkgd.gif
- 741C3E77.gif
- upgrade.xsl
- 70482F6B.gif

The following file list includes 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.
- 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 section contains form definition (.xsf) file examples that demonstrate different modes, as specified in section 1.3.1, and features.

3.2.1 Form Definition File (XSF) Examples: Browser-Compatible Form

This example form definition (.xsf) file illustrates the use of the following features:

- The **dataObject** element (section 2.2.1.2.17) specifies that it contains a data adapter that queries a list (1) located at http://www.someserver.com with **sharePointListID**="{EFD22576-5D4F-40F9-80A5-DD53930974C4}", and that the **ID**, **Modified** and **Created** fields (3) are read.
- The **documentSchemas** element (section 2.2.1.2.41) specifies that there are two XML schema documents that are used to verify the form file.

- The **server** element (section 2.2.2.2.2) specifies that the form template is not compatible with mobile Web browsers and that the form (1) does not postback to the protocol server before submitting the form file.
- The **customValidation** element (section 2.2.3.2.17) specifies that /my:myFields/my:field2 cannot be blank.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
This file is automatically created and modified by Microsoft 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="3.0.0.0"
solutionVersion="1.0.0.8" productVersion="14.0.0"
publishUrl="http://www.someserver.com" name="urn:schemas-microsoft-
com:office:infopath:browser:-myXSD-2009-05-05T20-26-38"
xmlns:xsf="http://schemas.microsoft.com/office/infopath/2003/solutionDefinition"
xmlns:xsf2="http://schemas.microsoft.com/office/infopath/2006/solutionDefinition/exten
sions" 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="http://schemas.microsoft.com/office/infopath/2003/xslt/extension"
xmlns:xdEnvironment="http://schemas.microsoft.com/office/infopath/2006/xslt/environmen
t" xmlns:xdUser="http://schemas.microsoft.com/office/infopath/2006/xslt/User"
xmlns:xdServerInfo="http://schemas.microsoft.com/office/infopath/2009/xslt/ServerInfo"
xmlns:my="http://schemas.microsoft.com/office/infopath/2003/myXSD/2009-05-05T20:26:38"
xmlns:pc="http://schemas.microsoft.com/office/infopath/2007/PartnerControls"
xmlns:ma="http://schemas.microsoft.com/office/2009/metadata/properties/metaAttributes"
xmlns:d="http://schemas.microsoft.com/office/infopath/2009/WSSList/dataFields"
xmlns:q="http://schemas.microsoft.com/office/infopath/2009/WSSList/queryFields"
xmlns:dfs="http://schemas.microsoft.com/office/infopath/2003/dataFormSolution"
xmlns:dms="http://schemas.microsoft.com/office/2009/documentManagement/types"
xmlns:xhtml="http://www.w3.org/1999/xhtml">
  <xsf:package>
    <xsf:files>
      <xsf:file name="myschema.xsd">
        <xsf:fileProperties>
          <xsf:property name="namespace" type="string"
value="http://schemas.microsoft.com/office/infopath/2003/myXSD/2009-05-
05T20:26:38"></xsf:property>
          <xsf:property name="editability" type="string" value="full"></xsf:property>
          <xsf:property name="rootElement" type="string"
value="myFields"></xsf:property>
          <xsf:property name="useOnDemandAlgorithm" type="string"
value="yes"></xsf:property>
        </xsf:fileProperties>
      </xsf:file>
      <xsf:file name="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="componentId" type="string" value="7"></xsf:property>
          <xsf:property name="xmlToEditName" type="string" value="2"></xsf:property>
        </xsf:fileProperties>
      </xsf:file>
      <xsf:file name="Links.xsd">
        <xsf:fileProperties>
          <xsf:property name="namespace" type="string"
value="http://schemas.microsoft.com/office/infopath/2003/dataFormSolution"></xsf:prope
rty>
          <xsf:property name="dataObject" type="string" value="Links"></xsf:property>
          <xsf:property name="rootElement" type="string"
value="myFields"></xsf:property>
          <xsf:property name="useOnDemandAlgorithm" type="string"
value="yes"></xsf:property>
        </xsf:fileProperties>
      </xsf:file>
      <xsf:file name="Links1.xsd">
        <xsf:fileProperties>
          <xsf:property name="namespace" type="string"
value="http://schemas.microsoft.com/office/infopath/2007/PartnerControls"></xsf:proper
ty>
          <xsf:property name="dataObject" type="string" value="Links"></xsf:property>
        </xsf:fileProperties>
      </xsf:file>
      <xsf:file name="Links2.xsd">
        <xsf:fileProperties>
          <xsf:property name="namespace" type="string"
value="http://schemas.microsoft.com/office/2009/documentManagement/types"></xsf:proper
ty>
          <xsf:property name="dataObject" type="string" value="Links"></xsf:property>
        </xsf:fileProperties>
      </xsf:file>
      <xsf:file name="Links3.xsd">
        <xsf:fileProperties>
          <xsf:property name="namespace" type="string"
value="http://schemas.microsoft.com/office/infopath/2009/WSSList/dataFields"></xsf:pro
perty>
          <xsf:property name="dataObject" type="string" value="Links"></xsf:property>
        </xsf:fileProperties>
      </xsf:file>
      <xsf:file name="Links4.xsd">
        <xsf:fileProperties>
          <xsf:property name="namespace" type="string"
value="http://schemas.microsoft.com/office/infopath/2009/WSSList/queryFields"></xsf:pr
operty>
          <xsf:property name="dataObject" type="string" value="Links"></xsf:property>
        </xsf:fileProperties>
      </xsf:file>
      <xsf:file name="upgrade.xsl"></xsf:file>
      <xsf:file name="gdotkota.xsd">
        <xsf:fileProperties>
          <xsf:property name="dataObject" type="string"
value="gdotkota"></xsf:property>
          <xsf:property name="rootElement" type="string"
value="gergely"></xsf:property>
```

```
          <xsf:property name="useOnDemandAlgorithm" type="string"
value="yes"></xsf:property>
        </xsf:fileProperties>
      </xsf:file>
      <xsf:file name="gdotkota">
        <xsf:fileProperties>
          <xsf:property name="fileType" type="string" value="resource"></xsf:property>
        </xsf:fileProperties>
      </xsf:file>
      <xsf:file name="BuiltInActiveXControls.xsd">
        <xsf:fileProperties>
          <xsf:property name="namespace" type="string"
value="http://schemas.microsoft.com/office/infopath/2007/PartnerControls"></xsf:proper
ty>
          <xsf:property name="editability" type="string" value="none"></xsf:property>
        </xsf:fileProperties>
      </xsf:file>
    </xsf:files>
  </xsf:package>
  <xsf:importParameters enabled="yes"></xsf:importParameters>
  <xsf:documentVersionUpgrade>
    <xsf:useTransform transform="upgrade.xsl" minVersionToUpgrade="0.0.0.0"
maxVersionToUpgrade="1.0.0.6"></xsf:useTransform>
  </xsf:documentVersionUpgrade>
  <xsf:extensions>
    <xsf:extension name="SolutionDefinitionExtensions">
      <xsf2:solutionDefinition runtimeCompatibility="client server"
runtimeCompatibilityURL="http://www.someserver.com/_vti_bin/FormsServices.asmx"
verifyOnServer="yes">
        <xsf2:offline openIfQueryFails="yes" cacheQueries="yes"></xsf2:offline>
        <xsf2:server formLocale="en-US" isPreSubmitPostBackEnabled="no"
isMobileEnabled="no"></xsf2:server>
        <xsf2:viewsExtension>
          <xsf2:viewExtension ref="View 1" designMode="normal">
            <xsf2:xmlToEditExtension ref="field2_2" allowLinkedImages="yes"
excludeEmbeddedImages="yes"></xsf2:xmlToEditExtension>
          </xsf2:viewExtension>
        </xsf2:viewsExtension>
      </xsf2:solutionDefinition>
      <xsf3:solutionDefinition
xmlns:xsf3="http://schemas.microsoft.com/office/infopath/2009/solutionDefinition/exten
sions">
        <xsf3:customValidation>
          <xsf3:errorBlank match="/my:myFields/my:field2" expressionContext="."
expression="(. = &quot;&quot; or . = &quot; &quot;) and not(descendant-or-
self::node()/*[name() = &quot;img&quot;])"></xsf3:errorBlank>
        </xsf3:customValidation>
      </xsf3: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="field2_2" item="/my:myFields/my:field2">
          <xsf:editWith type="rich" maxLength="1" component="xField"></xsf:editWith>
        </xsf:xmlToEdit>
      </xsf:editing>
```

```
    </xsf:view>
  </xsf:views>
  <xsf:applicationParameters application="InfoPath Design Mode">
    <xsf:solutionProperties
fullyEditableNamespace="http://schemas.microsoft.com/office/infopath/2003/myXSD/2009-
05-05T20:26:38" lastOpenView="view1.xsl"
lastVersionNeedingTransform="1.0.0.6"></xsf:solutionProperties>
  </xsf:applicationParameters>
  <xsf:documentSchemas>
    <xsf:documentSchema rootSchema="yes"
location="http://schemas.microsoft.com/office/infopath/2003/myXSD/2009-05-05T20:26:38
myschema.xsd"></xsf:documentSchema>
    <xsf:documentSchema
location="http://schemas.microsoft.com/office/infopath/2007/PartnerControls
BuiltInActiveXControls.xsd"></xsf:documentSchema>
  </xsf:documentSchemas>
  <xsf:fileNew>
    <xsf:initialXmlDocument caption="browser"
href="template.xml"></xsf:initialXmlDocument>
  </xsf:fileNew>
  <xsf:dataObjects>
    <xsf:dataObject name="Links" schema="Links.xsd" initOnLoad="yes">
      <xsf:query>
        <xsf:sharepointListAdapterRW queryAllowed="yes" submitAllowed="no"
siteURL="http://www.someserver.com" sharePointListID="{EFD22576-5D4F-40F9-80A5-
DD53930974C4}" name="Links" contentTypeID="" sortBy="ID" sortAscending="yes"
relativeListUrl="Lists/Links">
          <xsf:field internalName="ID" required="no" type="Counter"></xsf:field>
          <xsf:field internalName="Modified" required="no"
type="DateTime"></xsf:field>
          <xsf:field internalName="Created" required="no" type="DateTime"></xsf:field>
        </xsf:sharepointListAdapterRW>
      </xsf:query>
    </xsf:dataObject>
    <xsf:dataObject name="gdotkota" schema="gdotkota.xsd" initOnLoad="yes">
      <xsf:query>
        <xsf:xmlFileAdapter fileUrl="x-soln:///gdotkota"
name="gdotkota"></xsf:xmlFileAdapter>
      </xsf:query>
    </xsf:dataObject>
  </xsf:dataObjects>
  <xsf:customValidation></xsf:customValidation>
  <xsf:permissions>
    <xsf:allowedControl clsid="{61E40D31-993D-4777-8FA0-
19CA59B6D0BB}"></xsf:allowedControl>
  </xsf:permissions>
</xsf:xDocumentClass>
```

3.2.2 Form Definition File (XSF) Examples: List Form

This example form definition (.xsf) file specifies that this is a list form template published to http://www.someserver.com/Lists/Links. The **sharepointListAdapterRW** element (section 2.2.1.2.132) describes the list (1) with which this form template (.xsn) file is associated.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
This file is automatically created and modified by Microsoft InfoPath.
```

```
Changes made to the file outside of InfoPath might be lost if the form template is modified
in InfoPath.
-->
<xsf:xDocumentClass solutionFormatVersion="3.0.0.0" solutionVersion="1.0.0.2"
productVersion="14.0.0" publishUrl="http://www.someserver.com/Lists/Links" name="urn:schemas-
microsoft-com:office:infopath:list:-AutoGen-2009-05-05T19:03:07:350Z"
xmlns:xsf="http://schemas.microsoft.com/office/infopath/2003/solutionDefinition"
xmlns:xsf2="http://schemas.microsoft.com/office/infopath/2006/solutionDefinition/extensions"
xmlns:xsf3="http://schemas.microsoft.com/office/infopath/2009/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="http://schemas.microsoft.com/office/infopath/2003/xslt/extension"
xmlns:xdEnvironment="http://schemas.microsoft.com/office/infopath/2006/xslt/environment"
xmlns:xdUser="http://schemas.microsoft.com/office/infopath/2006/xslt/User"
xmlns:my="http://schemas.microsoft.com/office/infopath/2009/WSSList/cmeDataFields"
xmlns:dfs="http://schemas.microsoft.com/office/infopath/2003/dataFormSolution"
xmlns:d="http://schemas.microsoft.com/office/infopath/2009/WSSList/dataFields"
xmlns:xdServerInfo="http://schemas.microsoft.com/office/infopath/2009/xslt/ServerInfo"
xmlns:pc="http://schemas.microsoft.com/office/infopath/2007/PartnerControls"
xmlns:ma="http://schemas.microsoft.com/office/2009/metadata/properties/metaAttributes"
xmlns:q="http://schemas.microsoft.com/office/infopath/2009/WSSList/queryFields"
xmlns:dms="http://schemas.microsoft.com/office/2009/documentManagement/types">
  <xsf:package>
    <xsf:files>
      <xsf:file name="rootschema.xsd">
        <xsf:fileProperties>
          <xsf:property name="namespace" type="string"
value="http://schemas.microsoft.com/office/infopath/2003/dataFormSolution"></xsf:property>
          <xsf:property name="editability" type="string" value="none"></xsf:property>
          <xsf:property name="rootElement" type="string" value="myFields"></xsf:property>
          <xsf:property name="useOnDemandAlgorithm" type="string" value="yes"></xsf:property>
        </xsf:fileProperties>
      </xsf:file>
      <xsf:file name="qfschema.xsd">
        <xsf:fileProperties>
          <xsf:property name="namespace" type="string"
value="http://schemas.microsoft.com/office/infopath/2009/WSSList/queryFields"></xsf:property>
          <xsf:property name="editability" type="string" value="none"></xsf:property>
        </xsf:fileProperties>
      </xsf:file>
      <xsf:file name="dfschema.xsd">
        <xsf:fileProperties>
          <xsf:property name="namespace" type="string"
value="http://schemas.microsoft.com/office/infopath/2009/WSSList/cmeDataFields"></xsf:propert
y>
          <xsf:property name="editability" type="string" value="full"></xsf:property>
        </xsf:fileProperties>
      </xsf:file>
      <xsf:file name="typeschema.xsd">
        <xsf:fileProperties>
          <xsf:property name="namespace" type="string"
value="http://schemas.microsoft.com/office/2009/documentManagement/types"></xsf:property>
          <xsf:property name="editability" type="string" value="none"></xsf:property>
        </xsf:fileProperties>
      </xsf:file>
```

```
      <xsf:file name="builtincontrolsschema.xsd">
        <xsf:fileProperties>
          <xsf:property name="namespace" type="string"
value="http://schemas.microsoft.com/office/infopath/2007/PartnerControls"></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="mode" type="string" value="1"></xsf:property>
          <xsf:property name="xmlToEditName" type="string" value="1"></xsf:property>
        </xsf:fileProperties>
      </xsf:file>
    </xsf:files>
  </xsf:package>
  <xsf:importParameters enabled="yes"></xsf:importParameters>
  <xsf:documentVersionUpgrade>
    <xsf:useTransform transform="" minVersionToUpgrade="0.0.0.0"></xsf:useTransform>
  </xsf:documentVersionUpgrade>
  <xsf:extensions>
    <xsf:extension name="SolutionDefinitionExtensions">
      <xsf2:solutionDefinition runtimeCompatibility="client server"
runtimeCompatibilityURL="http://www.someserver.com/_vti_bin/FormsServices.asmx"
verifyOnServer="yes" allowClientOnlyCode="no">
        <xsf2:offline openIfQueryFails="yes" cacheQueries="yes"></xsf2:offline>
        <xsf2:server isPreSubmitPostBackEnabled="no" isMobileEnabled="no" formLocale="en-
US"></xsf2:server>
        <xsf2:solutionPropertiesExtension branch="list">
          <xsf2:list path="http://www.someserver.com/"></xsf2:list>
        </xsf2:solutionPropertiesExtension>
        <xsf2:viewsExtension>
          <xsf2:viewExtension ref="Edit item" clientOnly="no"></xsf2:viewExtension>
        </xsf2:viewsExtension>
      </xsf2:solutionDefinition>
      <xsf3:solutionPropertiesExtension2009>
        <xsf3:solutionMode mode="list"
originalPublishUrl="http://www.someserver.com/"></xsf3:solutionMode>
      </xsf3:solutionPropertiesExtension2009>
      <xsf3:solutionDefinition>
        <xsf3:customValidation></xsf3:customValidation>
        <xsf3:viewsExtension>
          <xsf3:viewExtension ref="Edit item"></xsf3:viewExtension>
        </xsf3:viewsExtension>
      </xsf3:solutionDefinition>
    </xsf:extension>
  </xsf:extensions>
  <xsf:applicationParameters application="InfoPath Design Mode">
    <xsf:solutionProperties
fullyEditableNamespace="http://schemas.microsoft.com/office/infopath/2009/WSSList/cmeDataFiel
ds" lastOpenView="view1.xsl"></xsf:solutionProperties>
  </xsf:applicationParameters>
```

```
  <xsf:documentSchemas>
    <xsf:documentSchema rootSchema="yes"
location="http://schemas.microsoft.com/office/infopath/2003/dataFormSolution
rootschema.xsd"></xsf:documentSchema>
    <xsf:documentSchema rootSchema="no"
location="http://schemas.microsoft.com/office/infopath/2009/WSSList/queryFields
qfschema.xsd"></xsf:documentSchema>
    <xsf:documentSchema rootSchema="no"
location="http://schemas.microsoft.com/office/infopath/2009/WSSList/cmeDataFields
dfschema.xsd"></xsf:documentSchema>
    <xsf:documentSchema rootSchema="no"
location="http://schemas.microsoft.com/office/2009/documentManagement/types
typeschema.xsd"></xsf:documentSchema>
    <xsf:documentSchema rootSchema="no"
location="http://schemas.microsoft.com/office/infopath/2007/PartnerControls
builtincontrolsschema.xsd"></xsf:documentSchema>
  </xsf:documentSchemas>
  <xsf:fileNew>
    <xsf:initialXmlDocument caption="Template" href="template.xml"></xsf:initialXmlDocument>
  </xsf:fileNew>
  <xsf:calculations></xsf:calculations>
  <xsf:views default="Edit item">
    <xsf:view name="Edit item" caption="Edit item">
      <xsf:mainpane transform="view1.xsl"></xsf:mainpane>
      <xsf:editing>
        <xsf:xmlToEdit name="xImage_URL"
item="/dfs:myFields/dfs:dataFields/my:SharePointListItem_RW/my:URL">
          <xsf:editWith component="xImage"></xsf:editWith>
        </xsf:xmlToEdit>
        <xsf:xmlToEdit name="Comments_1"
item="/dfs:myFields/dfs:dataFields/my:SharePointListItem_RW/my:Comments">
          <xsf:editWith type="plainMultiline" component="xField"></xsf:editWith>
        </xsf:xmlToEdit>
        <xsf:xmlToEdit name="URL_1"
item="/dfs:myFields/dfs:dataFields/my:SharePointListItem_RW/my:URL">
          <xsf:editWith component="xImage"></xsf:editWith>
        </xsf:xmlToEdit>
      </xsf:editing>
    </xsf:view>
  </xsf:views>
  <xsf:query>
    <xsf:sharepointListAdapterRW siteURL="http://www.someserver.com/"
sharePointListID="{EFD22576-5D4F-40F9-80A5-DD53930974C4}"
contentTypeID="0x01050057E75A325225FF499779D9968DF3275F" relativeListUrl="Lists/Links"
queryAllowed="yes" submitAllowed="yes" name="Main Data Connection" queryOneItemOnly="yes"
version="88e33c9e1d7f63bc">
      <xsf:field internalName="ID" type="Counter" required="no" appendOnly="no"></xsf:field>
      <xsf:field internalName="Title" type="Text" required="no" appendOnly="no"></xsf:field>
      <xsf:field internalName="Author" type="User" required="no" appendOnly="no"></xsf:field>
      <xsf:field internalName="Editor" type="User" required="no" appendOnly="no"></xsf:field>
      <xsf:field internalName="Modified" type="DateTime" required="no"
appendOnly="no"></xsf:field>
      <xsf:field internalName="Created" type="DateTime" required="no"
appendOnly="no"></xsf:field>
      <xsf:field internalName="URL" type="URL" required="yes" appendOnly="no"></xsf:field>
      <xsf:field internalName="Comments" type="Plain" required="no"
appendOnly="no"></xsf:field>
    </xsf:sharepointListAdapterRW>
  </xsf:query>
```

```
  <xsf:submit onAfterSubmit="close" showStatusDialog="no">
    <xsf:errorMessage>The form cannot be submitted because of an error.</xsf:errorMessage>
    <xsf:useQueryAdapter></xsf:useQueryAdapter>
  </xsf:submit>
</xsf:xDocumentClass>
```

The following example illustrates the algorithm specified in section 2.2.1.2.132 to calculate the **version** attribute of the **sharepointListAdapterRW** element.

The first step is to obtain the CAML that specifies the content type for the list (1) that this form template (.xsn) file is associated with, as shown in the following example.

```
<ContentType ID="0x010500E61FA3E964317F4F90FCC5FD106A3265" Name="Link" Group="List Content
Types" Description="Create a new link to a Web page or other resource."
V2ListTemplateName="favorite" Version="0" FeatureId="{695b6570-a48b-4a8e-8ea5-26ea7fc1d162}"
xmlns="http://schemas.microsoft.com/sharepoint/soap/">
  <Fields>
    <Field ID="{c042a256-787d-4a6f-8a8a-cf6ab767f12d}" Type="Computed" DisplayName="Content
Type" ReadOnly="TRUE" Name="ContentType" DisplaceOnUpgrade="TRUE"
RenderXMLUsingPattern="TRUE" SourceID="http://schemas.microsoft.com/sharepoint/v3"
StaticName="ContentType" Group="_Hidden" PITarget="MicrosoftWindowsSharePointServices"
PIAttribute="ContentTypeID" FromBaseType="TRUE">
      ...
    </Field>
    <Field ID="{fa564e0f-0c70-4ab9-b863-0177e6ddd247}" Type="Text" Name="Title"
DisplayName="Title" Required="FALSE" SourceID="http://schemas.microsoft.com/sharepoint/v3"
StaticName="Title" FromBaseType="TRUE" ReadOnly="TRUE" Hidden="TRUE" ColName="nvarchar1" />
    <Field ID="{82642ec8-ef9b-478f-acf9-31f7d45fbc31}" ReadOnly="TRUE" Type="Computed"
Name="LinkTitle" DisplayName="Title" DisplayNameSrcField="Title" ClassInfo="Menu"
AuthoringInfo="(linked to item with edit menu)" ListItemMenuAllowed="Required"
LinkToItemAllowed="Prohibited" SourceID="http://schemas.microsoft.com/sharepoint/v3"
StaticName="LinkTitle" FromBaseType="TRUE" Hidden="TRUE">
    ...
    </Field>
    <Field ID="{bc91a437-52e7-49e1-8c4e-4698904b2b6d}" ReadOnly="TRUE" Type="Computed"
Name="LinkTitleNoMenu" DisplayName="Title" Dir="" DisplayNameSrcField="Title"
AuthoringInfo="(linked to item)" EnableLookup="TRUE" ListItemMenuAllowed="Prohibited"
LinkToItemAllowed="Prohibited" SourceID="http://schemas.microsoft.com/sharepoint/v3"
StaticName="LinkTitleNoMenu" FromBaseType="TRUE" Hidden="TRUE">
    ...
    </Field>
    <Field ID="{c29e077d-f466-4d8e-8bbe-72b66c5f205c}" Type="URL" Name="URL"
DisplayName="URL" Required="TRUE" FromBaseType="TRUE" ShowInViewForms="FALSE"
SourceID="http://schemas.microsoft.com/sharepoint/v3" StaticName="URL" ColName="nvarchar3"
ColName2="nvarchar4" />
    <Field ID="{9da97a8a-1da5-4a77-98d3-4bc10456e700}" Type="Note" Name="Comments"
DisplayName="Notes" Sortable="FALSE" SourceID="http://schemas.microsoft.com/sharepoint/v3"
StaticName="Comments" ColName="ntext2" />
    <Field ID="{2a9ab6d3-268a-4c1c-9897-e5f018f87e64}" ReadOnly="TRUE" Filterable="FALSE"
Type="Computed" Name="URLwMenu" DisplayName="URL" DisplayNameSrcField="URL" ClassInfo="Menu"
AuthoringInfo="(URL with edit menu)" ListItemMenuAllowed="Required"
LinkToItemAllowed="Prohibited" SourceID="http://schemas.microsoft.com/sharepoint/v3"
StaticName="URLwMenu">
    ...
    </Field>
    <Field ID="{aeaf07ee-d2fb-448b-a7a3-cf7e062d6c2a}" DisplaceOnUpgrade="TRUE"
ReadOnly="TRUE" Filterable="FALSE" Type="Computed" Name="URLNoMenu" DisplayName="URL"
```

```
DisplayNameSrcField="URL" SourceID="http://schemas.microsoft.com/sharepoint/v3"
StaticName="URLNoMenu">
    ...
    </Field>
  </Fields>
  <XmlDocuments>
    <XmlDocument NamespaceURI="http://schemas.microsoft.com/sharepoint/v3/contenttype/forms">
      <FormTemplates xmlns="http://schemas.microsoft.com/sharepoint/v3/contenttype/forms">
        <Display>ListForm</Display>
        <Edit>ListForm</Edit>
        <New>ListForm</New>
      </FormTemplates>
    </XmlDocument>
  </XmlDocuments>
  <Folder TargetName="Link" />
</ContentType>
```

Then a **string** that summarizes all the properties for every field (3) in the content type is calculated from the CAML, as shown in the following example.

```
"Name:ContentType
Type:Computed
ReadOnly:TRUE



Name:Title
Type:Text
Required:FALSE
ReadOnly:TRUE



Name:LinkTitle
Type:Computed
ReadOnly:TRUE



Name:LinkTitleNoMenu
Type:Computed
ReadOnly:TRUE



Name:URL
Type:URL
Required:TRUE


Name:Comments
Type:Note
```

```
Name:URLwMenu
Type:Computed
ReadOnly:TRUE



Name:URLNoMenu
Type:Computed
ReadOnly:TRUE
```

Finally, a hash of this string is calculated using the SHA-1 hash algorithm. This generates the value of the **version** attribute of the **sharepointListAdapterRW** element, "88e33c9e1d7f63bc".

3.3 XML Schema Files (XSD) Examples

Section 2.3 provides example XSD constructs for supported controls.

Following is an example XML schema document.

```
<?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 schema document is **myFields**, which is the root element for all the other elements that represent a control in the schema document. **myFields** contains a reference to **my:field1**, **my:group1**, and **my:field3**.

- **my:field1** represents a text box control in the XML schema document that can have a **string** content.
- **my:group1** contains another group (1), **my:group2**. **my:group2** is a repeating element. This is used to represent repeating controls, such as repeating section controls.
- **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 example XSL fragments for all of the controls specified in section 2.4.1. Each fragment provides an example of how a control can be structured with features such as conditional formatting, data formatting, or retrieving selection options from a data source (2).

3.4.1.1 Button Control

The following XSL examples are for button controls, as specified in section 2.3.1.1.

The following example is a button control with conditional formatting. The **name** attribute is set to the value of **my:field1**. This means that the button's display text is the value of **my:field1**. Conditional formatting is set such that if the value of **my:field3** is "true", the control is hidden.

```
<input class="langFont" title="" type="button" xd:xctname="Button" xd:CtrlId="CTRL1_5"
tabIndex="0">
    <xsl:attribute name="style">
        <xsl:choose>
            <xsl:when test="my:field3 = string(true())">DISPLAY: none; caption: Rule
1</xsl:when>
        </xsl:choose>
    </xsl:attribute>
    <xsl:attribute name="value">
        <xsl:value-of select="my:field1"/>
    </xsl:attribute>
</input>
```

The following example is a button control that is used to update the form content in the Web browser. The button display text is the value of **my:field1**. Conditional formatting is set such that if the value of **my:field2** is "Red", the control has a red background color.

```
<input class="langFont" title="" style="BEHAVIOR: url(#default#ActionButton)" type="button"
xd:xctname="Button" xd:CtrlId="CTRL1_5" xd:action="updateForm" tabIndex="0">
    <xsl:attribute name="style">BEHAVIOR: url(#default#ActionButton);<xsl:choose>
            <xsl:when test="not(xdEnvironment:IsBrowser())">DISPLAY: none</xsl:when>
```

```
            <xsl:when test="my:field2 = &quot;Red&quot;">BACKGROUND-COLOR: #ff0000; caption:
Rule 1</xsl:when>
        </xsl:choose>
    </xsl:attribute>
    <xsl:attribute name="value">
        <xsl:value-of select="my:field1"/>
    </xsl:attribute>
</input>
```

The following example is a button control that is used to submit the form data. The button display text is statically set to "Submit". This control has two conditional formatting settings. If the value of **my:field1** is "1", the control is disabled and has a yellow background color. If the first conditional formatting condition is "false" and the value of **my:field2** is "abc", the button display text is bold and the control has an orange background.

```
<input class="langFont" title="Press to submit this form" style="BEHAVIOR:
url(#default#ActionButton)" accessKey="S" type="button" value="Submit" xd:xctname="Button"
xd:CtrlId="CTRL1_5" xd:action="submit" xd:postbackModel="always" tabIndex="0">
    <xsl:attribute name="style">BEHAVIOR: url(#default#ActionButton);<xsl:choose>
            <xsl:when test="my:field1 = 1">BACKGROUND-COLOR: #ffff00; caption: Rule
1</xsl:when>
            <xsl:when test="my:field2 = &quot;abc&quot;">FONT-WEIGHT: bold; COLOR: #ff6600;
caption: Rule 2</xsl:when>
        </xsl:choose>
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="my:field1 = 1">
            <xsl:attribute name="disabled">true</xsl:attribute>
        </xsl:when>
        <xsl:when test="my:field2 = &quot;abc&quot;"/>
    </xsl:choose>
</input>
```

The following example is a button control that is used to refresh the content of a secondary data source (2). The button display text is statically set to "Refresh". Conditional formatting is set such that if the value of **my:field1** is "1", the control is disabled and has a yellow background color.

```
<input class="langFont" title="" style="BEHAVIOR: url(#default#ActionButton)" type="button"
value="Refresh" xd:xctname="Button" xd:CtrlId="CTRL1_5" xd:action="refresh"
xd:auxDom="UserNameList" tabIndex="0">
    <xsl:attribute name="style">BEHAVIOR: url(#default#ActionButton);<xsl:choose>
            <xsl:when test="my:field1 = 1">BACKGROUND-COLOR: #ffff00; caption: Rule
1</xsl:when>
        </xsl:choose>
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="my:field1 = 1">
            <xsl:attribute name="disabled">true</xsl:attribute>
        </xsl:when>
    </xsl:choose>
</input>
```

3.4.1.2 Check Box Control

The following XSL examples are for check box controls, as specified in section 3.4.1.2.

The following example is a check box control with the value "1" if the control is cleared, and zero ("0") if the control is selected. When the user hovers over the control with the cursor, it displays the message "this is a checkbox".

```
<input class="xdBehavior_Boolean" title="this is a checkbox" type="checkbox"
xd:binding="my:field1" xd:boundProp="xd:value" xd:offValue="1" xd:onValue="0" tabIndex="0"
xd:xctname="CheckBox" xd:CtrlId="CTRL1">
    <xsl:attribute name="xd:value">
        <xsl:value-of select="my:field1" />
    </xsl:attribute>
    <xsl:if test="my:field1=&quot;true&quot;">
        <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
    </xsl:if>
</input>
```

The following example is a check box control with the value "false" if the control is cleared, and "true" if the control is selected. Conditional formatting is set such that if the control is selected, the control is disabled.

```
<input class="xdBehavior_Boolean" title="" type="checkbox" xd:binding="my:field2"
xd:boundProp="xd:value" xd:offValue="false" xd:onValue="true" tabIndex="0"
xd:xctname="CheckBox" xd:CtrlId="CTRL2">
    <xsl:attribute name="style">
        <xsl:choose>
            <xsl:when test="my:field2 = string(true())">caption: Rule 1</xsl:when>
        </xsl:choose>
    </xsl:attribute>
    <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=&quot;true&quot;">
        <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
    </xsl:if>
</input>
```

3.4.1.3 Contact Selector Control

The following XSL example is a contact selector control, as specified in section 3.4.1.3, with conditional formatting. Conditional formatting is set such that if the value of **my:field1** is "false", the control is disabled.

```
<object class="xdActiveX" hideFocus="1" style="WIDTH: 288px; HEIGHT: 23px"
classid="clsid:61e40d31-993d-4777-8fa0-19ca59b6d0bb" tabIndex="0" tabStop="true"
xd:xctname="{{61e40d31-993d-4777-8fa0-19ca59b6d0bb}}" xd:CtrlId="CTRL1"
xd:server="http://server" xd:bindingType="xmlNode" xd:bindingProperty="Value"
```

```
xd:boundProp="xd:inline" xd:AllowMultiple="true" xd:SearchPeopleOnly="true"
xd:SharePointGroup="0" 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:attribute name="style">WIDTH: 288px; HEIGHT: 23px;<xsl:choose>
            <xsl:when test="my:field1 = string(false())">caption: Rule 1</xsl:when>
        </xsl:choose>
    </xsl:attribute>
    <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="Calibri,11,0,400,0,0,0"/>
    <param NAME="ButtonText" VALUE=""/>
    <param NAME="DisplayNameXPath" VALUE="pc:DisplayName"/>
    <param NAME="ObjectIdXPath" VALUE="pc:AccountId"/>
    <param NAME="ObjectTypeXPath" VALUE="pc:AccountType"/>
    <param NAME="SiteUrlXPath" VALUE="/Context/@siteUrl"/>
    <param NAME="SiteUrlDataSource" VALUE="Context"/>
    <param NAME="NewNodeTemplate"
VALUE="&lt;pc:Person&gt;&#xA;&lt;pc:DisplayName&gt;&lt;/pc:DisplayName&gt;&#xA;&lt;pc:Account
Id&gt;&lt;/pc:AccountId&gt;&#xA;&lt;pc:AccountType&gt;&lt;/pc:AccountType&gt;&#xA;&lt;/pc:Per
son&gt;"/>
    <param NAME="BackgroundColor" VALUE="2147483653"/>
    <param NAME="MaxLines" VALUE="4"/>
    <param NAME="Direction" VALUE="1"/>
</object>
```

3.4.1.4 Date Picker Control

The following XSL examples are for date picker controls, as specified in section 3.4.1.4.

The following example is a date picker control where **xd:datafmt** is equal to ""date","dateFormat:Short Date;"". This formats the value of **my:field1** to be a short date.

```
<div class="xdDTPicker" title="" style="WIDTH: 130px" noWrap="1" xd:CtrlId="CTRL1"
xd:xctname="DTPicker">
    <span class="xdDTText xdBehavior_FormattingNoBUI" hideFocus="1" contentEditable="true"
xd:xctname="DTPicker_DTText" xd:datafmt="&amp;quot;date&amp;quot;,&amp;quot;dateFormat:Short Date;&amp;quot;"
xd:boundProp="xd:num" xd:binding="my:field1" tabIndex="0" xd:innerCtrl="_DTText">
        <xsl:attribute name="xd:num">
            <xsl:value-of select="my:field1" />
        </xsl:attribute>
        <xsl:choose>
            <xsl:when test="function-available('xdFormatting:formatString')">
                <xsl:value-of
select="xdFormatting:formatString(my:field1,&amp;quot;date&amp;quot;,&amp;quot;dateFormat:Short
Date;&amp;quot;)" />
            </xsl:when>
            <xsl:otherwise>
                <xsl:value-of select="my:field1" />
            </xsl:otherwise>
        </xsl:choose>
```

```
    </span>
    <button class="xdDTButton" xd:xctname="DTPicker_DTButton" xd:innerCtrl="_DTButton"
tabIndex="0">
        <img src="res://infopath.exe/calendar.gif" />
    </button>
</div>
```

The following example is a date picker control where **xd:datafmt** is equal to ""date","locale:1041;dateFormat:y'年'M'月'd'日';useAltCalendar:1;"" and **xd:datafmt2** is equal to "calendar:3;". This formats the value of **my:field1** as a date in the Japanese Emperor calendar (formatted as y'年'M'月'd'日').

```
<div class="xdDTPicker" title="" style="WIDTH: 130px" noWrap="1" xd:CtrlId="CTRL1"
xd:xctname="DTPicker">
    <span class="xdDTText xdBehavior_FormattingNoBUI" hideFocus="1" contentEditable="true"
xd:xctname="DTPicker_DTText"
xd:datafmt="&amp;quot;date&amp;quot;,&amp;quot;locale:1041;dateFormat:y'年M'月'd'日';useAltCalendar:1;&amp;quot
;" xd:boundProp="xd:num" xd:binding="my:field1" tabIndex="0" xd:innerCtrl="_DTText"
xd:datafmt2="calendar:3;">
        <xsl:attribute name="xd:num">
            <xsl:value-of select="my:field1"/>
        </xsl:attribute>
        <xsl:choose>
            <xsl:when test="function-available('xdFormatting:formatString2')">
                <xsl:value-of
select="xdFormatting:formatString2(my:field1,&amp;quot;date&amp;quot;,&amp;quot;locale:1041;dateFormat:y'
年M'月'd'日';useAltCalendar:1;&amp;quot;,'calendar:3;')"/>
            </xsl:when>
            <xsl:when test="function-available('xdFormatting:formatString')">
                <xsl:value-of
select="xdFormatting:formatString(my:field1,&amp;quot;date&amp;quot;,&amp;quot;locale:1041;dateFormat:y'年
'M'月'd'日';useAltCalendar:1;&amp;quot;)"/>
            </xsl:when>
            <xsl:otherwise>
                <xsl:value-of select="my:field1"/>
            </xsl:otherwise>
        </xsl:choose>
    </span>
    <button class="xdDTButton" xd:xctname="DTPicker_DTButton" xd:innerCtrl="_DTButton"
tabIndex="0">
        <img src="res://infopath.exe/calendar.gif"/>
    </button>
</div>
```

The following example is a date picker control with conditional formatting. Conditional formatting is set such that if the value of **my:field3** is "1900-01-01", the text for the control is bold and strikethrough.

```
<div class="xdDTPicker" title="" style="WIDTH: 130px" noWrap="1" xd:CtrlId="CTRL3"
xd:xctname="DTPicker">
    <span class="xdDTText xdBehavior_FormattingNoBUI" hideFocus="1" contentEditable="true"
xd:xctname="DTPicker_DTText" xd:datafmt="&amp;quot;date&amp;quot;,&amp;quot;dateFormat:Short Date;&amp;quot;"
xd:boundProp="xd:num" xd:binding="my:field3" tabIndex="0" xd:innerCtrl="_DTText">
        <xsl:attribute name="style">
            <xsl:choose>
```

```
                <xsl:when test="my:field3 = &quot;1900-01-01&quot;">FONT-WEIGHT: bold; TEXT-
DECORATION: line-through; caption: Rule 1</xsl:when>
            </xsl:choose>
        </xsl:attribute>
        <xsl:attribute name="xd:num">
            <xsl:value-of select="my:field3" />
        </xsl:attribute>
        <xsl:choose>
            <xsl:when test="function-available('xdFormatting:formatString')">
                <xsl:value-of
select="xdFormatting:formatString(my:field3,&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="0">
        <img src="res://infopath.exe/calendar.gif" />
    </button>
</div>
```

3.4.1.5 Drop-Down List Control

The following XSL examples are for drop-down list controls, as specified in section 3.4.1.5.

The following example is a drop-down list control with the static values "Select...", "1", "2", and "3".

```
<select class="xdComboBox xdBehavior_Select" title="" size="1" xd:binding="my:field1"
xd:boundProp="value" xd:xctname="dropdown" tabIndex="0" xd:CtrlId="CTRL1" style="WIDTH:
130px">
    <xsl:attribute name="value">
        <xsl:value-of select="my:field1" />
    </xsl:attribute>
    <option>
        <xsl:if test="my:field1=&quot;&quot;">
            <xsl:attribute name="selected">selected</xsl:attribute>
        </xsl:if>
        Select...
    </option>
    <option value="1">
        <xsl:if test="my:field1=&quot;1&quot;">
            <xsl:attribute name="selected">selected</xsl:attribute>
        </xsl:if>
        1
    </option>
    <option value="2">
        <xsl:if test="my:field1=&quot;2&quot;">
            <xsl:attribute name="selected">selected</xsl:attribute>
        </xsl:if>
        2
    </option>
    <option value="3">
        <xsl:if test="my:field1=&quot;3&quot;">
```

```
            <xsl:attribute name="selected">selected</xsl:attribute>
        </xsl:if>
        3
    </option>
</select>
```

The following example is a drop-down list control with values that are dynamically generated from an external data source (2) called "sample" and a filter that only allows data that meets the filter criteria to be displayed.

```
<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
[contains(., &quot;value&quot;)] [.=$val] or $val='')">
                <option selected="selected">
                    <xsl:attribute name="value">
                        <xsl:value-of select="$val" />
                    </xsl:attribute>
                    <xsl:value-of select="$val" />
                </option>
            </xsl:if>
            <xsl:for-each select="xdXDocument:GetDOM(&quot;sample&quot;)/main/small/big/name
[contains(., &quot;value&quot;)]">
                <option>
                    <xsl:attribute name="value">
                        <xsl:value-of select="." />
                    </xsl:attribute>
                    <xsl:if test="$val=.">
                        <xsl:attribute name="selected">selected</xsl:attribute>
                    </xsl:if>
                    <xsl:value-of select="." />
                </option>
            </xsl:for-each>
        </xsl:when>
        <xsl:otherwise>
            <option>
                <xsl:value-of select="my:field2" />
            </option>
        </xsl:otherwise>
    </xsl:choose>
</select>
```

The following example is a drop-down list control with values that are dynamically generated from an external data source (2) called "sample", displaying only unique entries.

```
<select class="xdComboBox xdBehavior_Select" title="" style="WIDTH: 130px" size="1"
xd:binding="my:field2" xd:boundProp="value" value="" xd:xctname="dropdown" xd:CtrlId="CTRL2"
tabIndex="0">
    <xsl:attribute name="value">
        <xsl:value-of select="my:field2" />
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="function-available('xdXDocument:GetDOM')">
            <option />
            <xsl:variable name="val" select="my:field2" />
            <xsl:if
test="not(xdXDocument:GetDOM(&quot;sample&quot;)/main/small/big/name[.=$val] or $val='')">
                <option selected="selected">
                    <xsl:attribute name="value">
                        <xsl:value-of select="$val" />
                    </xsl:attribute>
                    <xsl:value-of select="$val" />
                </option>
            </xsl:if>
            <xsl:variable name="items">
                <xsl:copy-of
select="xdXDocument:GetDOM(&quot;sample&quot;)/main/small/big/name" />
            </xsl:variable>
            <xsl:variable name="uniqueItems" select="msxsl:node-set($items)/*[not(. =
preceding::name)]" />
            <xsl:for-each select="$uniqueItems">
                <option>
                    <xsl:attribute name="value">
                        <xsl:value-of select="." />
                    </xsl:attribute>
                    <xsl:if test="$val=.">
                        <xsl:attribute name="selected">selected</xsl:attribute>
                    </xsl:if>
                    <xsl:value-of select="." />
                </option>
            </xsl:for-each>
        </xsl:when>
        <xsl:otherwise>
            <option>
                <xsl:value-of select="my:field2" />
            </option>
        </xsl:otherwise>
    </xsl:choose>
</select>
```

The following example is a drop-down list control with conditional formatting and values that are dynamically generated from an external data source (2) called "sample", displaying only unique entries.

```
<select class="xdComboBox xdBehavior_Select" title="" style="WIDTH: 130px" size="1"
xd:binding="my:field2" xd:boundProp="value" value="" xd:xctname="dropdown" xd:CtrlId="CTRL2"
tabIndex="0">
    <xsl:attribute name="style">
        WIDTH: 130px;
        <xsl:choose>
```

```
            <xsl:when test="my:field2 = &quot;bob&quot;">FONT-WEIGHT: bold; COLOR: #808000;
FONT-STYLE: italic; BACKGROUND-COLOR: #800000; TEXT-DECORATION: underline line-through;
caption: Rule 1</xsl:when>
            <xsl:when test="my:field2 = &quot;theodore&quot;">caption: Rule 2</xsl:when>
        </xsl:choose>
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="my:field2 = &quot;bob&quot;" />
        <xsl:when test="my:field2 = &quot;theodore&quot;">
            <xsl:attribute name="disabled">true</xsl:attribute>
        </xsl:when>
    </xsl:choose>
    <xsl:attribute name="value">
        <xsl:value-of select="my:field2" />
        </xsl:attribute>
            <xsl:choose>
        <xsl:when test="function-available('xdXDocument:GetDOM')">
            <option />
            <xsl:variable name="val" select="my:field2" />
            <xsl:if
test="not(xdXDocument:GetDOM(&quot;sample&quot;)/main/small/big/name[.=$val] or $val='')">
                <option selected="selected">
                    <xsl:attribute name="value">
                        <xsl:value-of select="$val" />
                    </xsl:attribute>
                    <xsl:value-of select="$val" />
                </option>
            </xsl:if>
            <xsl:variable name="items">
                <xsl:copy-of
select="xdXDocument:GetDOM(&quot;sample&quot;)/main/small/big/name" />
            </xsl:variable>
            <xsl:variable name="uniqueItems" select="msxsl:node-set($items)/*[not(. =
preceding::name)]" />
            <xsl:for-each select="$uniqueItems">
                <option>
                    <xsl:attribute name="value">
                        <xsl:value-of select="." />
                    </xsl:attribute>
                    <xsl:if test="$val=.">
                        <xsl:attribute name="selected">selected</xsl:attribute>
                    </xsl:if>
                    <xsl:value-of select="." />
                </option>
            </xsl:for-each>
        </xsl:when>
        <xsl:otherwise>
            <option>
                <xsl:value-of select="my:field2" />
            </option>
        </xsl:otherwise>
    </xsl:choose>
</select>
```

3.4.1.6 Expression Box Control

The following XSL examples are for expression box controls, as specified in section 3.4.1.6.

The following example is an expression box control that is displaying the value of **my:field1**.

```
<span class="xdExpressionBox xdDataBindingUI" title="" xd:xctname="ExpressionBox" tabIndex="-
1" xd:CtrlId="CTRL3" xd:disableEditing="yes" style="WIDTH: 145px">
    <xsl:value-of select="my:field1" />
</span>
```

The following example is an expression box control with conditional formatting that is displaying the value of **my:field1**. **xd:datafmt** is ""datetime","dateFormat:Short Date;timeFormat:none;"". This formats the value of **my:field1** to be a short date.

```
<span class="xdExpressionBox xdDataBindingUI xdBehavior_Formatting" title="texas"
xd:binding="my:field1" xd:xctname="ExpressionBox" tabIndex="-1" xd:CtrlId="CTRL4"
xd:disableEditing="yes" xd:datafmt="&amp;quot;datetime&amp;quot;,&amp;quot;dateFormat:Short
Date;timeFormat:none;&amp;quot;" xd:num="">
    <xsl:attribute name="style">
        WIDTH: 145px;
        <xsl:choose>
            <xsl:when test="my:field1 = &amp;quot;1&amp;quot;">DISPLAY: none; caption: Rule
1</xsl:when>
        </xsl:choose>
    </xsl:attribute>
    <xsl:attribute name="xd:num">
        <xsl:value-of select="my:field1" />
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="function-available('xdFormatting:formatString')">
            <xsl:value-of
select="xdFormatting:formatString(my:field1,&amp;quot;datetime&amp;quot;,&amp;quot;dateFormat:Short
Date;timeFormat:none;&amp;quot;)" />
        </xsl:when>
        <xsl:otherwise>
            <xsl:value-of select="my:field1" />
        </xsl:otherwise>
    </xsl:choose>
</span>
```

3.4.1.7 File Attachment Control

The following XSL example is for a file attachment control, as specified in section 3.4.1.7.

```
<span class="xdFileAttachment" hideFocus="1" style="WIDTH: 161px; HEIGHT: 30px"
tabStop="true" xd:binding="my:field1" xd:boundProp="xd:inline" tabIndex="0"
xd:xctname="FileAttachment" xd:CtrlId="CTRL1">
    <xsl:if test="function-available('xdImage:getImageUrl')">
        <xsl:attribute name="src"><xsl:value-of
select="xdImage:getImageUrl(my:field1)"/></xsl:attribute>
    </xsl:if>
</span>
```

3.4.1.8 Hyperlink Control

The following XSL examples are for hyperlink controls, as specified in section 3.4.1.8.

The following example is a static hyperlink control.

```
<a href="http://example.com" xd:disableEditing="yes">http://example.com</a>
```

The following example is a hyperlink control that dynamically changes its target as well as its display text. The target of the hyperlink is the value of **my:field1** and the display text is the value of **my:field2**. This control also contains **border formatting** and shading formatting.

```
<span class="xdHyperlink" hideFocus="1" title="" style="BORDER-RIGHT: #cbd8eb 4.5pt dotted;
BORDER-TOP: #cbd8eb 4.5pt dotted; OVERFLOW: visible; BORDER-LEFT: #cbd8eb 4.5pt dotted;
WIDTH: 130px; BORDER-BOTTOM: #cbd8eb 4.5pt dotted; BACKGROUND-COLOR: #ffff00; TEXT-ALIGN:
left" xd:xctname="hyperlink">
    <a class="xdDataBindingUI" xd:CtrlId="CTRL5" xd:disableEditing="yes">
        <xsl:attribute name="href">
            <xsl:value-of select="my:field1" />
        </xsl:attribute>
        <xsl:value-of select="my:field2" />
    </a>
</span>
```

3.4.1.9 List Box Control

The following XSL examples are for list box controls, as specified in section 3.4.1.9.

The following example is a list box control with three selection entries.

```
<select class="xdListBox xdBehavior_Select" title="" size="3" xd:binding="my:field1"
xd:boundProp="value" tabIndex="0" xd:xctname="ListBox" xd:CtrlId="CTRL1" style="WIDTH:
130px">
    <xsl:attribute name="value">
        <xsl:value-of select="my:field1"/>
    </xsl:attribute>
    <option value="a">
        <xsl:if test="my:field1=&quot;a&quot;">
            <xsl:attribute name="selected">selected</xsl:attribute>
        </xsl:if>A</option>
    <option value="b">
        <xsl:if test="my:field1=&quot;b&quot;">
            <xsl:attribute name="selected">selected</xsl:attribute>
        </xsl:if>B</option>
    <option value="c">
        <xsl:if test="my:field1=&quot;c&quot;">
            <xsl:attribute name="selected">selected</xsl:attribute>
        </xsl:if>C</option>
</select>
```

The following example is a list box control that looks up the selection options from a repeating group within the main data source (2). The control only displays unique selection options. Conditional formatting is set such that if the value of **my:field1** is "a", the control has a red background color.

```
<select class="xdListBox xdBehavior_Select" title="" style="WIDTH: 130px" size="3"
xd:binding="my:field1" xd:boundProp="value" value="a" xd:xctname="ListBox" xd:CtrlId="CTRL1"
tabIndex="0">
    <xsl:attribute name="style">WIDTH: 130px;<xsl:choose>
            <xsl:when test="my:field1 = &quot;a&quot;">BACKGROUND-COLOR: #ff0000; caption:
Rule 1</xsl:when>
        </xsl:choose>
    </xsl:attribute>
    <xsl:attribute name="value">
        <xsl:value-of select="my:field1"/>
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="function-available('xdXDocument:GetDOM')">
        <option/>
            <xsl:variable name="val" select="my:field1"/>
                <xsl:if test="not(my:group1/my:group2[my:field2=$val] or $val='')">
                    <option selected="selected">
                        <xsl:attribute name="value">
                            <xsl:value-of select="$val"/>
                        </xsl:attribute>
                        <xsl:value-of select="$val"/>
                    </option>
                </xsl:if>
            <xsl:variable name="items">
                <xsl:copy-of select="my:group1/my:group2"/>
            </xsl:variable>
            <xsl:variable name="uniqueItems" select="msxsl:node-set($items)/*[not(my:field3 =
preceding::my:group2/my:field3)]"/>
            <xsl:for-each select="$uniqueItems">
                <option>
                    <xsl:attribute name="value">
                        <xsl:value-of select="my:field2"/>
                    </xsl:attribute>
                    <xsl:if test="$val=my:field2">
                        <xsl:attribute name="selected">selected</xsl:attribute>
                    </xsl:if>
                    <xsl:value-of select="my:field3"/>
                </option>
            </xsl:for-each>
        </xsl:when>
        <xsl:otherwise>
            <option>
                <xsl:value-of select="my:field1"/>
            </option>
        </xsl:otherwise>
    </xsl:choose>
</select>
```

The following example is a list box control that looks up the selection options from a repeating group in a secondary data source called "UserNameList". Conditional formatting is set such that if the value of **my:field1** is "a", the control has a red background color.

```
<select class="xdListBox xdBehavior_Select" title="" style="WIDTH: 130px" size="3"
xd:binding="my:field1" xd:boundProp="value" value="a" xd:xctname="ListBox" xd:CtrlId="CTRL1"
tabIndex="0">
    <xsl:attribute name="style">WIDTH: 130px;<xsl:choose>
```

```
            <xsl:when test="my:field1 = &quot;a&quot;">BACKGROUND-COLOR: #ff0000; caption:
Rule 1</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(&quot;UserNameList&quot;)/dfs:myFields/dfs:dataFields/dfs:UserNa
meList[@E-mail_Address=$val] or $val='')">
                <option selected="selected">
                    <xsl:attribute name="value">
                        <xsl:value-of select="$val"/>
                    </xsl:attribute>
                    <xsl:value-of select="$val"/>
                </option>
            </xsl:if>
            <xsl:for-each
select="xdXDocument:GetDOM(&quot;UserNameList&quot;)/dfs:myFields/dfs:dataFields/dfs:UserName
List">
                <option>
                    <xsl:attribute name="value">
                        <xsl:value-of select="@E-mail_Address"/>
                    </xsl:attribute>
                    <xsl:if test="$val=@E-mail_Address">
                        <xsl:attribute name="selected">selected</xsl:attribute>
                    </xsl:if>
                    <xsl:value-of select="@Last_Name"/>
                </option>
            </xsl:for-each>
        </xsl:when>
        <xsl:otherwise>
            <option>
                <xsl:value-of select="my:field1"/>
            </option>
        </xsl:otherwise>
    </xsl:choose>
</select>
```

3.4.1.10 Option Button Control

The following XSL examples are for option button controls, as specified in section 3.4.1.10.

The following example is an option button control with three option buttons.

```
<div>
    <input class="xdBehavior_Boolean" title="" type="radio" name="{generate-id(my:field3)}"
xd:binding="my:field3" xd:boundProp="xd:value" xd:xctname="OptionButton" tabIndex="0"
xd:CtrlId="CTRL6" xd:onValue="1">
        <xsl:attribute name="xd:value">
            <xsl:value-of select="my:field3" />
        </xsl:attribute>
```

```
        <xsl:if test="my:field3=&amp;quot;1&amp;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=&amp;quot;2quot;">
            <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
        </xsl:if>
    </input>
    2
</div>
<div>
    <input class="xdBehavior_Boolean" title="" type="radio" name="{generate-id(my:field3)}"
xd:binding="my:field3" xd:boundProp="xd:value" xd:xctname="OptionButton" tabIndex="0"
xd:CtrlId="CTRL8" xd:onValue="3">
        <xsl:attribute name="xd:value">
            <xsl:value-of select="my:field3" />
        </xsl:attribute>
        <xsl:if test="my:field3=quot;3quot;">
            <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
        </xsl:if>
    </input>
    3
</div>
```

The following example is an option button control with conditional formatting. Conditional formatting is set such that if the value of **my:field3** is "2", the control is disabled.

```
<input class="xdBehavior_Boolean" title="" type="radio" name="{generate-id(my:field3)}"
xd:binding="my:field3" xd:boundProp="xd:value" xd:xctname="OptionButton" tabIndex="0"
xd:CtrlId="CTRL7" xd:onValue="2">
    <xsl:attribute name="style">
        <xsl:choose>
            <xsl:when test="my:field3 = 2">caption: Rule 1</xsl:when>
        </xsl:choose>
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="my:field3 = 2">
            <xsl:attribute name="disabled">true</xsl:attribute>
        </xsl:when>
    </xsl:choose>
    <xsl:attribute name="xd:value">
        <xsl:value-of select="my:field3" />
    </xsl:attribute>
    <xsl:if test="my:field3=&amp;quot;2&amp;quot;">
        <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
    </xsl:if>
```

```
</input>
```

3.4.1.11 Repeating Section Control

The following XSL example is a repeating section controls, as specified in 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 "true", the control is disabled.

The following example is the repeating section call.

```
<span>
    <xsl:attribute name="style">
        <xsl:if test="my:field1 = string(true())">msos-xCollection-group2_1-
editing:disabled;</xsl:if>
    </xsl:attribute>
    <div><xsl:apply-templates select="my:group1/my:group2" mode="_1"/>
        <div class="optionalPlaceholder" xd:xmlToEdit="group2_1" tabIndex="0"
xd:action="xCollection::insert" align="left" style="WIDTH: 651px">Insert item</div>
    </div>
    <div> </div>
    <div> </div>
</span>
```

The following example is the 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"
xd:widgetIndex="0">
        <xsl:attribute name="style">MARGIN-BOTTOM: 6px; WIDTH: 651px;<xsl:choose>
                <xsl:when test="../../my:field1 = string(true())">caption: Rule 1</xsl:when>
            </xsl:choose>
        </xsl:attribute>
        <div> </div>
        <div><xsl:apply-templates select="my:group3" mode="_2"/>
        </div>
        <div> </div>
    </div>
</xsl:template>
```

3.4.1.12 Repeating Table Control

The following XSL examples are for repeating table controls, as specified in section 3.4.1.12.

The following example is a repeating table control that 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" xd:widgetIndex="0">
        <colgroup>
            <col style="WIDTH: 210px" />
            <col style="WIDTH: 211px" />
            <col style="WIDTH: 230px" />
        </colgroup>
        <tbody class="xdTableHeader">
            <tr>
                <td>
                    <div>
                        <strong />
                    </div>
                </td>
                <td>
                    <div>
                        <strong />
                    </div>
                </td>
                <td>
                    <div>
                        <strong />
                    </div>
                </td>
            </tr>
        </tbody>
        <tbody xd:xctname="RepeatingTable">
            <xsl:for-each select="my:group1/my:group2">
                <tr>
                    <td>
                        <span class="xdTextBox" hideFocus="1" title="" xd:binding="my:field5"
xd:xctname="PlainText" tabIndex="0" xd:CtrlId="CTRL13" style="WIDTH: 100%">
                            <xsl:value-of select="my:field5" />
                        </span>
                    </td>
                    <td>
                        <span class="xdTextBox" hideFocus="1" title="" xd:binding="my:field6"
xd:xctname="PlainText" tabIndex="0" xd:CtrlId="CTRL14" style="WIDTH: 100%">
                            <xsl:value-of select="my:field6" />
                        </span>
                    </td>
                    <td>
                        <span class="xdTextBox" hideFocus="1" title="" xd:binding="my:field7"
xd:xctname="PlainText" tabIndex="0" xd:CtrlId="CTRL15" style="WIDTH: 100%">
                            <xsl:value-of select="my:field7" />
                        </span>
                    </td>
                </tr>
            </xsl:for-each>
        </tbody>
</table>
<div class="optionalPlaceholder" xd:xmlToEdit="group2_8" tabIndex="0"
xd:action="xCollection::insert" style="WIDTH: 651px">Insert item</div>
</div>
```

The following example is a repeating table control that has three columns (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 "2", the control has a different background color. This control also causes a postback whenever a table row is inserted or removed.

```
<div>
    <table class="xdRepeatingTable msoUcTable" title="" style="TABLE-LAYOUT: fixed; WIDTH:
651px; BORDER-TOP-STYLE: none; BORDER-RIGHT-STYLE: none; BORDER-LEFT-STYLE: none; BORDER-
COLLAPSE: collapse; WORD-WRAP: break-word; BORDER-BOTTOM-STYLE: none" border="1"
xd:CtrlId="CTRL16" xd:widgetIndex="0" xd:postbackModel="always">
        <colgroup>
            <col style="WIDTH: 210px" />
            <col style="WIDTH: 211px" />
            <col style="WIDTH: 230px" />
        </colgroup>
        <tbody class="xdTableHeader">
            <tr>
                <td>
                    <div>
                        <strong />
                    </div>
                </td>
                <td>
                    <div>
                        <strong />
                    </div>
                </td>
                <td>
                    <div>
                        <strong />
                    </div>
                </td>
            </tr>
        </tbody>
        <tbody xd:xctname="RepeatingTable">
            <xsl:for-each select="my:group3/my:group4">
                <xsl:if test="not((my:field8 = quot;1quot;))">
                    <tr xd:caption_0="Rule 1">
                        <xsl:attribute name="style">
                            <xsl:choose>
                                <xsl:when test="my:field8 = 2">BACKGROUND-COLOR: #ff00ff;
caption: Rule 2</xsl:when>
                            </xsl:choose>
                        </xsl:attribute>
                        <td>
                            <span class="xdTextBox" hideFocus="1" title=""
xd:binding="my:field8" xd:xctname="PlainText" tabIndex="0" xd:CtrlId="CTRL17" style="WIDTH:
100%">
                                <xsl:value-of select="my:field8" />
                            </span>
                        </td>
                        <td>
                            <span class="xdTextBox" hideFocus="1" title=""
xd:binding="my:field9" xd:xctname="PlainText" tabIndex="0" xd:CtrlId="CTRL18" style="WIDTH:
100%">
                                <xsl:value-of select="my:field9" />
                            </span>
                        </td>
                        <td>
```

```
                            <span class="xdTextBox" hideFocus="1" title=""
xd:binding="my:field10" xd:xctname="PlainText" tabIndex="0" xd:CtrlId="CTRL19" style="WIDTH:
100%">
                                <xsl:value-of select="my:field10" />
                            </span>
                        </td>
                    </tr>
                </xsl:if>
            </xsl:for-each>
        </tbody>
        <tbody class="xdTableFooter">
            <tr>
                <td>
                    <div> </div>
                </td>
                <td>
                    <div> </div>
                </td>
                <td>
                    <div> </div>
                </td>
            </tr>
        </tbody>
    </table>
</div>
```

The following example is a repeating table control that has one column (2) with a text box inside. Conditional formatting is set such that if the value of **my:field1** is "1", the control does not allow the user to insert or delete rows from the table. Note that this conditional formatting is placed outside the repeating table element.

```
<span>
    <xsl:attribute name="style">
        <xsl:if test="my:field1 = &quot;1&quot;">msos-xCollection-group8_16-
editing:disabled;</xsl:if>
    </xsl:attribute>
    <div>
        <table class="xdRepeatingTable msoUcTable" title="" style="TABLE-LAYOUT: fixed;
WIDTH: 651px; BORDER-TOP-STYLE: none; BORDER-RIGHT-STYLE: none; BORDER-LEFT-STYLE: none;
BORDER-COLLAPSE: collapse; WORD-WRAP: break-word; BORDER-BOTTOM-STYLE: none" border="1"
xd:CtrlId="CTRL24" xd:widgetIndex="0">
            <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>
```

```
                    <xsl:attribute name="style">
                        <xsl:choose>
                            <xsl:when test="../../my:field1 = &quot;1&quot;">caption: Rule
1</xsl:when>
                        </xsl:choose>
                    </xsl:attribute>
                    <td>
                        <span class="xdTextBox" hideFocus="1" title=""
xd:binding="my:field14" xd:xctname="PlainText" tabIndex="0" xd:CtrlId="CTRL25" style="WIDTH:
100%">
                            <xsl:value-of select="my:field14" />
                        </span>
                    </td>
                </tr>
            </xsl:for-each>
        </tbody>
    </table>
    <div class="optionalPlaceholder" xd:xmlToEdit="group8_16" tabIndex="0"
xd:action="xCollection::insert" style="WIDTH: 651px">Insert item</div>
    </div>
</span>
```

3.4.1.13 Rich Text Box Control

The following XSL example is a rich text box control, as specified in section 3.4.1.13, with conditional formatting set such that if the value of **my:field2** is "false", the control is disabled.

```
<span class="xdRichTextBox" hideFocus="1" title="" xd:binding="my:field1" tabIndex="0"
xd:xctname="RichText" xd:CtrlId="CTRL1" style="WIDTH: 651px; HEIGHT: 50px">
    <xsl:attribute name="style">WIDTH: 651px; HEIGHT: 50px;<xsl:choose>
            <xsl:when test="my:field2 = string(true())">caption: Rule 1</xsl:when>
        </xsl:choose>
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="my:field2 = string(true())">
            <xsl:attribute name="contentEditable">false</xsl:attribute>
        </xsl:when>
    </xsl:choose>
    <xsl:copy-of select="my:field1/node()"/>
</span>
```

3.4.1.14 Section Control and Optional Section Control

The following XSL examples are for section and optional section controls, as specified in section 2.4.1.18.

The first example is a section control that can be digitally signed. This control contains a text box control.

The following example is the section call.

```
<xsl:apply-templates select="my:group1" mode="_1"/>
```

The following example is the 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" xd:widgetIndex="0">
        <div> </div>
        <div><span class="xdTextBox" hideFocus="1" title="" xd:xctname="PlainText"
xd:CtrlId="CTRL2" tabIndex="0" xd:binding="my:field1" style="WIDTH: 130px">
                <xsl:value-of select="my:field1"/>
            </span>
        </div>
        <div> </div>
    </div>
    <div xd:disableEditing="yes" xd:SignatureBlock="group1"
xd:SignedSectionDisplaySignatures="true" style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 6px;
BEHAVIOR: url (#default#SignaturesInDocUI); WIDTH: 651px">
        <xsl:if test="function-available('xdXDocument:GetNamedNodeProperty')">
            <xsl:if
test="xdXDocument:GetNamedNodeProperty(/my:myFields/my:signatures1/my:signatures2,
'CanAddSignature', 'false') = 'true'">
                <button title="" style="width: 100%; height: 100%; text-align: left; border:
0px solid; padding: 2px; background-color: window; cursor: hand;">
                    <table style="color: windowtext;" class="defaultInDocUI">
                        <tbody>
                            <tr>
                                <td><img src="res://infopath.exe/AddSignatureButton.gif"
width="12" height="12"/></td>
                                <td>Click here to sign this section</td>
                            </tr>
                        </tbody>
                    </table>
                </button>
            </xsl:if>
            <xsl:for-each select="/my:myFields/my:signatures1/my:signatures2">
                <xsl:for-each select="sig:Signature">
                    <xsl:choose>
                        <xsl:when test="xdXDocument:GetNamedNodeProperty(.,
'IsValidSignature', 'false') = 'true'">
                            <button title="" style="width: 100%; height: 100%; text-align:
left; border: 0px solid; padding: 2px; background-color: window; cursor: hand;">
                                <table style="color: windowtext;" class="defaultInDocUI">
                                    <tbody>
                                        <tr>
                                            <xsl:choose>
                                                <xsl:when test="function-
available('xdImage:getImageUrl') and
sig:Object/sig:SignatureProperties/sig:SignatureProperty/xdSignatureProperties:NonRepudiation
/xdSignatureProperties:ValidSignedImage">
                                                    <td style="display: none;"><img
src="res://infopath.exe/signaturevalid.gif"/></td>
                                                    <td><img
src="{xdImage:getImageUrl(sig:Object/sig:SignatureProperties/sig:SignatureProperty/xdSignatur
eProperties:NonRepudiation/xdSignatureProperties:ValidSignedImage)}"/> </td>
                                                </xsl:when>
                                                <xsl:otherwise>
                                                    <td><img
src="res://infopath.exe/signaturevalid.gif" width="12" height="12"/></td>
                                                    <td style="color: gray;">
```

```
                                                    <div><b><xsl:value-of
select="xdXDocument:GetNamedNodeProperty(., 'SignedBy', '???')"/></b><span style="margin: 0pt
20pt">View details</span></div>
                                                    <div><xsl:value-of
select="xdXDocument:GetNamedNodeProperty(., 'SignedOn', '???')"/></div>
                                                </td>
                                            </xsl:otherwise>
                                        </xsl:choose>
                                    </tr>
                                </tbody>
                            </table>
                        </button>
                    </xsl:when>
                    <xsl:otherwise>
                        <button title="" style="width: 100%; height: 100%; text-align:
left; border: 0px solid; padding: 2px; background-color: window; cursor: hand;">
                            <table style="font: message-box; color: windowtext;">
                                <tbody>
                                    <tr>
                                        <xsl:choose>
                                            <xsl:when test="function-
available('xdImage:getImageUrl') and
sig:Object/sig:SignatureProperties/sig:SignatureProperty/xdSignatureProperties:NonRepudiation
/xdSignatureProperties:InvalidSignedImage">
                                                <td style="display: none;"><img
src="res://infopath.exe/signaturevalid.gif"/></td>
                                                <td><img
src="{xdImage:getImageUrl(sig:Object/sig:SignatureProperties/sig:SignatureProperty/xdSignatur
eProperties:NonRepudiation/xdSignatureProperties:InvalidSignedImage)}"/> </td>
                                            </xsl:when>
                                            <xsl:otherwise>
                                                <td><img
src="res://infopath.exe/signatureerror.gif" width="12" height="12"/></td>
                                                <td style="color: red;"><b>There is a
problem with this signature</b><span style="margin: 0pt 20pt">View details</span></td>
                                            </xsl:otherwise>
                                        </xsl:choose>
                                    </tr>
                                </tbody>
                            </table>
                        </button>
                    </xsl:otherwise>
                </xsl:choose>
            </xsl:for-each>
        </xsl:for-each>
    </xsl:if>
</div>
</xsl:template>
```

The next example is an example of an optional section control. This control contains a date picker control.

The following example is the 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>
```

The following example is the 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" xd:widgetIndex="0">
        <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="&quot;date&quot;,&quot;dateFormat:Short Date;&quot;"
xd:boundProp="xd:num" xd:innerCtrl="_DTText">
                    <xsl:attribute name="xd:num">
                        <xsl:value-of select="my:field2"/>
                    </xsl:attribute>
                    <xsl:choose>
                        <xsl:when test="function-available('xdFormatting:formatString')">
                            <xsl:value-of
select="xdFormatting:formatString(my:field2,&quot;date&quot;,&quot;dateFormat:Short
Date;&quot;)"/>
                        </xsl:when>
                        <xsl:otherwise>
                            <xsl:value-of select="my:field2"/>
                        </xsl:otherwise>
                    </xsl:choose>
                </span>
                <button class="xdDTButton" xd:xctname="DTPicker_DTButton"
xd:innerCtrl="_DTButton" tabIndex="-1">
                    <img src="res://infopath.exe/calendar.gif"/>
                </button>
            </div>
        </div>
        <div> </div>
    </div>
</xsl:template>
```

3.4.1.15 Table Control

The following XSL example is for a table control, as specified in section 3.4.1.15. This example is a table control that is two **rows (2)** 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:xctname="Button" xd:CtrlId="CTRL10_5" tabIndex="0" />
                    </font>
                </div>
            </td>
        </tr>
        <tr>
            <td>
                <div>
                    <font face="Verdana" size="2">1</font>
                </div>
            </td>
            <td>
                <div>
                    <font face="Verdana" size="2">1</font>
                </div>
            </td>
        </tr>
    </tbody>
</table>
```

3.4.1.16 Text Box Control

The following XSL examples are for text box controls, as specified in section 3.4.1.16.

The following example is a text box control that is bound to **my:field1**.

```
<div>
  <span class="xdTextBox" hideFocus="1" title="" xd:binding="my:field1" tabIndex="0"
xd:xctname="PlainText" xd:CtrlId="CTRL1" style="WIDTH: 130px">
    <xsl:value-of select="my:field1"/>
  </span>
</div>
```

The following example is a text box control with **multi-line** enabled that is bound to **my:field2**.

```
<div>
 <span class="xdTextBox" hideFocus="1" title="" xd:binding="my:field2" tabIndex="0"
xd:xctname="PlainText" xd:CtrlId="CTRL2"
xd:datafmt="&amp;quot;string&amp;quot;,&amp;quot;plainMultiline&amp;quot;" style="OVERFLOW-Y: auto; OVERFLOW-
X: auto; WIDTH: 130px; WHITE-SPACE: normal; WORD-WRAP: break-word">
    <xsl:choose>
      <xsl:when test="function-available('xdFormatting:formatString')">
```

```
        <xsl:value-of
select="xdFormatting:formatString(my:field2,&quot;string&quot;,&quot;plainMultiline&quot;)"
disable-output-escaping="yes"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="my:field2" disable-output-escaping="yes"/>
      </xsl:otherwise>
    </xsl:choose>
  </span>
</div>
```

The following example is a text box control that is bound to **my:field3**. The value of **xd:datafmt** is ""date","locale:1061; dateFormat:d.MM.yyyy;"" and the value of **xd:datafmt2** is "calendar:1;". This formats the value of **my:field3** to be a Gregorian date and specifies the locale as Estonian. The specific date format is "dateFormat:d.MM.yyyy". This formats the date and specifies how the day, month, and year are displayed. For this example, the date could be displayed as 14.03.2001 corresponding to March 14th, 2001.

```
<span class="xdTextBox xdBehavior_Formatting" hideFocus="1" title="" contentEditable="true"
xd:binding="my:field3" tabIndex="0" xd:xctname="PlainText" xd:CtrlId="CTRL3"
xd:boundProp="xd:num"
xd:datafmt="&quot;date&quot;,&quot;locale:1061;dateFormat:d.MM.yyyy;&quot;"
xd:datafmt2="calendar:1;" style="WIDTH: 130px">
    <xsl:attribute name="xd:num">
        <xsl:value-of select="my:field3"/>
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="function-available('xdFormatting:formatString2')">
            <xsl:value-of
select="xdFormatting:formatString2(my:field3,&quot;date&quot;,&quot;locale:1061;dateFormat:d.
MM.yyyy;&quot;,'calendar:1;')"/>
        </xsl:when>
        <xsl:when test="function-available('xdFormatting:formatString')">
            <xsl:value-of
select="xdFormatting:formatString(my:field3,&quot;date&quot;,&quot;locale:1061;dateFormat:d.M
M.yyyy;&quot;)"/>
        </xsl:when>
        <xsl:otherwise>
            <xsl:value-of select="my:field3"/>
        </xsl:otherwise>
    </xsl:choose>
</span>
```

The following example is a text box control that has conditional formatting and is bound to **my:field4**. Conditional formatting is set such that if the value of **my:field4** is "abc", the text in the control is bold.

```
<div>
  <span class="xdTextBox" hideFocus="1" title="" xd:binding="my:field4" tabIndex="0"
xd:xctname="PlainText" xd:CtrlId="CTRL4">
    <xsl:attribute name="style">WIDTH: 130px;
      <xsl:choose>
        <xsl:when test="my:field4 = &quot;abc&quot;">FONT-WEIGHT: bold; caption: Rule
1</xsl:when>
      </xsl:choose>
```

```
    </xsl:attribute>
    <xsl:value-of select="my:field4"/>
  </span>
</div>
```

The following example is a text box control that has conditional formatting and data formatting and is bound to **my:field5**. Conditional formatting is set such that if the value of **my:field5** is "def", the text in the control is underlined. The value of **xd:datafmt** is ""time","locale:1033;timeFormat:hh:mm:ss tt;"". This formats the value of **my:field5** to be a time and specifies the locale as English. The specific time format is "timeFormat:hh:mm:ss tt;". This formats the time and specifies how the hour, minutes, and seconds are displayed. For this example, the time could be displayed as 09:46:55 AM.

```
<div>
  <span class="xdTextBox xdBehavior_Formatting" hideFocus="1" title="" contentEditable="true"
xd:binding="my:field5" tabIndex="0" xd:xctname="PlainText" xd:CtrlId="CTRL5"
xd:datafmt="&amp;quot;time&amp;quot;,&amp;quot;locale:1033;timeFormat:hh:mm:ss tt;&amp;quot;"
xd:boundProp="xd:num">
    <xsl:attribute name="style">WIDTH: 130px;
      <xsl:choose>
        <xsl:when test="my:field5 = &amp;quot;def&amp;quot;">TEXT-DECORATION: underline; caption:
Rule 1</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,&amp;quot;time&amp;quot;,&amp;quot;locale:1033;timeFormat:hh:
mm:ss tt;&amp;quot;)"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="my:field5"/>
      </xsl:otherwise>
    </xsl:choose>
  </span>
</div>
```

3.4.2 Control Representation for Controls Introduced in Version 2 of the Structure Specification

3.4.2.1 Choice Group Control and Choice Section Control

The following XSL example is for choice group and choice section controls, as specified in section 2.4.1.21.1. This example is a choice group control that has two choice section controls. One contains the text "Choice Section 1" and can be conditionally formatted. The second contains a textbox control.

The following example is the choice group call.

```
<div style="WIDTH: 651px; MARGIN-BOTTOM: 6px" class="xdSection xdRepeating"
xd:xctname="choicegroup" xd:ref="/my:myFields/my:group1">
    <div><xsl:apply-templates select="my:group1/my:group2" mode="_2"/>
    </div>
    <div><xsl:apply-templates select="my:group1/my:group3" mode="_3"/>
    </div>
    <div> </div>
</div>
```

The following example is the choice goup body.

```
<xsl:template match="my:group2" mode="_2">
    <div style="WIDTH: 100%; MARGIN-BOTTOM: 6px" class="xdSection xdRepeating" title=""
align="left" xd:xctname="choiceterm" xd:CtrlId="CTRL1" tabIndex="-1" xd:widgetIndex="0">
        <xsl:attribute name="style">WIDTH: 100%; MARGIN-BOTTOM: 6px;<xsl:choose>
                <xsl:when test="../my:group3/my:field1 = &quot;1&quot;">BACKGROUND-COLOR:
#3366ff; caption: Rule 1</xsl:when>
            </xsl:choose>
        </xsl:attribute>
        <div> </div>
        <div>
            <span style="FONT-SIZE: 9pt">Choice Section 1</span>
        </div>
        <div> </div>
    </div>
</xsl:template>
<xsl:template match="my:group3" mode="_3">
    <div style="WIDTH: 100%; MARGIN-BOTTOM: 6px" class="xdSection xdRepeating" title=""
align="left" xd:xctname="choiceterm" xd:CtrlId="CTRL2" tabIndex="-1">
        <div> </div>
        <div> <span hideFocus="1" class="xdTextBox" title="" xd:xctname="PlainText"
xd:CtrlId="CTRL3" tabIndex="0" xd:binding="my:field1" style="WIDTH: 130px">
                <xsl:value-of select="my:field1"/>
            </span>
        </div>
        <div> </div>
    </div>
</xsl:template>
```

3.4.2.2 Combo Box Control

The following XSL examples are for the combo box control, as specified in section 2.3.2.2.

The following example is a combo box control with the static values " Select or type...", "1", "2", and "3".

```
<span class="xdComboBox xdBehavior_ComboBox" xd:xctname="combobox" style="WIDTH:
130px;LAYOUT-GRID:none;">
    <xsl:if test="function-available('ipApp:GetMajorVersion') and ipApp:GetMajorVersion()
&gt;= 12">
        <select tabIndex="-1" disabled="true" style="WIDTH:
130px;VISIBILITY:hidden;WIDTH:100%;"/>
        <span xd:xctname="PlainText" hideFocus="1" class="xdTextBox
xdBehavior_ComboBoxTextField" title="" xd:binding="my:field1" tabIndex="0" xd:CtrlId="CTRL1">
```

```
            <xsl:attribute name="style">WIDTH: 130px;POSITION:absolute;WIDTH:0px;WORD-
WRAP:normal</xsl:attribute>
            <xsl:choose>
                <xsl:when test="my:field1=&quot;&quot;">   Select or type...</xsl:when>
                <xsl:when test="my:field1=&quot;1&quot;">1</xsl:when>
                <xsl:when test="my:field1=&quot;2&quot;">2</xsl:when>
                <xsl:when test="my:field1=&quot;3&quot;">3</xsl:when>
                <xsl:otherwise>
                    <xsl:value-of select="my:field1"/>
                </xsl:otherwise>
            </xsl:choose>
        </span>
    </xsl:if>
    <select class="xdComboBox xdBehavior_Select" title="" size="1" xd:binding="my:field1"
xd:xctname="dropdown" xd:CtrlId="CTRL1" xd:boundProp="value">
        <xsl:choose>
            <xsl:when test="function-available('ipApp:GetMajorVersion') and
ipApp:GetMajorVersion() &gt;= 12">
                <xsl:attribute name="tabIndex">-1</xsl:attribute>
            </xsl:when>
            <xsl:otherwise>
                <xsl:attribute name="tabIndex">0</xsl:attribute>
            </xsl:otherwise>
        </xsl:choose>
        <xsl:attribute name="style">WIDTH: 130px
            <xsl:choose>
                <xsl:when test="function-available('ipApp:GetMajorVersion') and
ipApp:GetMajorVersion() &gt;= 12">;POSITION:absolute;WIDTH:0px;</xsl:when>
                <xsl:otherwise>;WIDTH: 130px;</xsl:otherwise>
            </xsl:choose>
        </xsl:attribute>
        <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 or type...
        </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>
</span>
```

The following example is a combo box control with values that are dynamically generated from an external data source (2) called "sample".

```
<span class="xdComboBox xdBehavior_ComboBox" xd:xctname="combobox" style="WIDTH:
130px;LAYOUT-GRID:none;">
        <xsl:if test="function-available('ipApp:GetMajorVersion') and ipApp:GetMajorVersion()
&gt;= 12">
            <select tabIndex="-1" disabled="true" style="WIDTH:
130px;VISIBILITY:hidden;WIDTH:100%;"/>
            <span xd:xctname="PlainText" hideFocus="1" class="xdTextBox
xdBehavior_ComboBoxTextField" title="" xd:binding="my:field2" value="" xd:CtrlId="CTRL2"
tabIndex="0">
                <xsl:attribute name="style">WIDTH: 130px;POSITION:absolute;WIDTH:0px;WORD-
WRAP:normal</xsl:attribute>
                <xsl:variable name="val" select="my:field2"/>
                <xsl:choose>
                    <xsl:when
test="xdXDocument:GetDOM(&quot;Sample&quot;)/Filters/Exclude/Filter/Guid[.=$val]/.">
                        <xsl:value-of
select="xdXDocument:GetDOM(&quot;Sample&quot;)/Filters/Exclude/Filter/Guid[.=$val]/."/>
                    </xsl:when>
                    <xsl:otherwise>
                        <xsl:value-of select="my:field2"/>
                    </xsl:otherwise>
                </xsl:choose>
            </span>
        </xsl:if>
        <select class="xdComboBox xdBehavior_Select" title="" size="1" xd:binding="my:field2"
value="" xd:xctname="dropdown" xd:CtrlId="CTRL2" xd:boundProp="value">
            <xsl:choose>
                <xsl:when test="function-available('ipApp:GetMajorVersion') and
ipApp:GetMajorVersion() &gt;= 12">
                    <xsl:attribute name="tabIndex">-1</xsl:attribute>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:attribute name="tabIndex">0</xsl:attribute>
                </xsl:otherwise>
            </xsl:choose>
            <xsl:attribute name="style">WIDTH: 130px
                <xsl:choose>
                    <xsl:when test="function-available('ipApp:GetMajorVersion') and
ipApp:GetMajorVersion() &gt;= 12">;POSITION:absolute;WIDTH:0px;</xsl:when>
                    <xsl:otherwise>;WIDTH: 130px;</xsl:otherwise>
                </xsl:choose>
            </xsl:attribute>
            <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;)/Filters/Exclude/Filter/Guid[.=$val] or
$val='')">
                        <option selected="selected">
                            <xsl:attribute name="value">
                                <xsl:value-of select="$val"/>
                            </xsl:attribute>
```

```
                        <xsl:value-of select="$val"/>
                    </option>
                </xsl:if>
                <xsl:for-each
select="xdXDocument:GetDOM(&quot;Sample&quot;)/Filters/Exclude/Filter/Guid">
                    <option>
                        <xsl:attribute name="value">
                            <xsl:value-of select="."/>
                        </xsl:attribute>
                        <xsl:if test="$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>
</span>
```

The following example is a combo box control with values that are dynamically generated from an external data source (2) called "sample", displaying only unique entries.

```
<span class="xdComboBox xdBehavior_ComboBox" xd:xctname="combobox" style="WIDTH:
130px;LAYOUT-GRID:none;">
    <xsl:if test="function-available('ipApp:GetMajorVersion') and ipApp:GetMajorVersion()
&gt;= 12">
        <select tabIndex="-1" disabled="true" style="WIDTH:
130px;VISIBILITY:hidden;WIDTH:100%;"/>
        <span xd:xctname="PlainText" hideFocus="1" class="xdTextBox
xdBehavior_ComboBoxTextField" title="" xd:binding="my:field3" value="" xd:CtrlId="CTRL3"
tabIndex="0">
            <xsl:attribute name="style">WIDTH: 130px;POSITION:absolute;WIDTH:0px;WORD-
WRAP:normal</xsl:attribute>
            <xsl:variable name="val" select="my:field3"/>
            <xsl:choose>
                <xsl:when
test="xdXDocument:GetDOM(&quot;Sample&quot;)/Filters/Exclude/Filter/Guid[.=$val]/.">
                    <xsl:value-of
select="xdXDocument:GetDOM(&quot;Sample&quot;)/Filters/Exclude/Filter/Guid[.=$val]/."/>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:value-of select="my:field3"/>
                </xsl:otherwise>
            </xsl:choose>
        </span>
    </xsl:if>
    <select class="xdComboBox xdBehavior_Select" title="" size="1" xd:binding="my:field3"
value="" xd:xctname="dropdown" xd:CtrlId="CTRL3" xd:boundProp="value">
        <xsl:choose>
```

```
            <xsl:when test="function-available('ipApp:GetMajorVersion') and
ipApp:GetMajorVersion() &gt;= 12">
                <xsl:attribute name="tabIndex">-1</xsl:attribute>
            </xsl:when>
            <xsl:otherwise>
                <xsl:attribute name="tabIndex">0</xsl:attribute>
            </xsl:otherwise>
        </xsl:choose>
        <xsl:attribute name="style">WIDTH: 130px
            <xsl:choose>
                <xsl:when test="function-available('ipApp:GetMajorVersion') and
ipApp:GetMajorVersion() &gt;= 12">;POSITION:absolute;WIDTH:0px;</xsl:when>
                <xsl:otherwise>;WIDTH: 130px;</xsl:otherwise>
            </xsl:choose>
        </xsl:attribute>
        <xsl:attribute name="value"><xsl:value-of select="my:field3"/></xsl:attribute>
        <xsl:choose>
            <xsl:when test="function-available('xdXDocument:GetDOM')">
                <option/>
                <xsl:variable name="val" select="my:field3"/>
                    <xsl:if
test="not(xdXDocument:GetDOM(&quot;Sample&quot;)/Filters/Exclude/Filter/Guid[.=$val] or
$val='')">
                        <option selected="selected">
                            <xsl:attribute name="value">
                                <xsl:value-of select="$val"/>
                            </xsl:attribute>
                                <xsl:value-of select="$val"/>
                        </option>
                    </xsl:if>
                    <xsl:variable name="items">
                        <xsl:copy-of
select="xdXDocument:GetDOM(&quot;Sample&quot;)/Filters/Exclude/Filter/Guid"/>
                    </xsl:variable>
                    <xsl:variable name="uniqueItems" select="msxsl:node-set($items)/*[not(. =
preceding::Guid)]"/>
                    <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:field3"/>
                    </option>
                </xsl:otherwise>
            </xsl:choose>
        </select>
    </span>
```

The following example is a combo box control with conditional formatting and values that are dynamically generated from an external data source (2) called "sample" displaying only unique entries.

```
<span class="xdComboBox xdBehavior_ComboBox" xd:xctname="combobox" style="WIDTH:
130px;LAYOUT-GRID:none;">
    <xsl:if test="function-available('ipApp:GetMajorVersion') and ipApp:GetMajorVersion()
&gt;= 12">
        <select tabIndex="-1" disabled="true" style="WIDTH:
130px;VISIBILITY:hidden;WIDTH:100%;"/>
        <span xd:xctname="PlainText" hideFocus="1" class="xdTextBox
xdBehavior_ComboBoxTextField" title="" xd:binding="my:field4" value="" xd:CtrlId="CTRL4"
tabIndex="0">
            <xsl:attribute name="style">WIDTH: 130px;
                <xsl:choose>
                    <xsl:when test="my:field4 = &quot;1&quot;">BACKGROUND-COLOR: #00ccff;
FONT-STYLE: italic; COLOR: #ff99cc; FONT-WEIGHT: bold; TEXT-DECORATION: underline line-
through; caption: Rule 1</xsl:when>
                </xsl:choose>;POSITION:absolute;WIDTH:0px;WORD-WRAP:normal
            </xsl:attribute>
            <xsl:variable name="val" select="my:field4"/>
            <xsl:choose>
                <xsl:when
test="xdXDocument:GetDOM(&quot;Sample&quot;)/Filters/Exclude/Filter/Guid[.=$val]/.">
                    <xsl:value-of
select="xdXDocument:GetDOM(&quot;Sample&quot;)/Filters/Exclude/Filter/Guid[.=$val]/."/>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:value-of select="my:field4"/>
                </xsl:otherwise>
            </xsl:choose>
            </span>
        </xsl:if>
        <select class="xdComboBox xdBehavior_Select" title="" size="1" xd:binding="my:field4"
value="" xd:xctname="dropdown" xd:CtrlId="CTRL4" xd:boundProp="value">
            <xsl:choose>
                <xsl:when test="function-available('ipApp:GetMajorVersion') and
ipApp:GetMajorVersion() &gt;= 12">
                    <xsl:attribute name="tabIndex">-1</xsl:attribute>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:attribute name="tabIndex">0</xsl:attribute>
                </xsl:otherwise>
            </xsl:choose>
            <xsl:attribute name="style">WIDTH: 130px;
                <xsl:choose>
                    <xsl:when test="my:field4 = &quot;1&quot;">BACKGROUND-COLOR: #00ccff;
FONT-STYLE: italic; COLOR: #ff99cc; FONT-WEIGHT: bold; TEXT-DECORATION: underline line-
through; caption: Rule 1</xsl:when>
                </xsl:choose>
                <xsl:choose>
                    <xsl:when test="function-available('ipApp:GetMajorVersion') and
ipApp:GetMajorVersion() &gt;= 12">;POSITION:absolute;WIDTH:0px;</xsl:when>
                    <xsl:otherwise>;WIDTH: 130px;</xsl:otherwise>
                </xsl:choose>
            </xsl:attribute>
            <xsl:attribute name="value">
                <xsl:value-of select="my:field4"/>
```

```
            </xsl:attribute>
            <xsl:choose>
                <xsl:when test="function-available('xdXDocument:GetDOM')">
                    <option/>
                    <xsl:variable name="val" select="my:field4"/>
                    <xsl:if
test="not(xdXDocument:GetDOM(&quot;Sample&quot;)/Filters/Exclude/Filter/Guid[.=$val] or
$val='')">
                        <option selected="selected">
                            <xsl:attribute name="value">
                                <xsl:value-of select="$val"/>
                            </xsl:attribute>
                            <xsl:value-of select="$val"/>
                        </option>
                    </xsl:if>
                        <xsl:for-each
select="xdXDocument:GetDOM(&quot;Sample&quot;)/Filters/Exclude/Filter/Guid">
                            <option>
                                <xsl:attribute name="value">
                                    <xsl:value-of select="."/>
                                </xsl:attribute>
                                <xsl:if test="$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:field4"/>
                        </option>
                    </xsl:otherwise>
        </xsl:choose>
    </select>
</span>
```

3.4.2.3 Date and Time Picker Control

The following XSL example is for date and time picker controls, as specified in section 2.3.2.3. This example is a date and time picker control where the **xd:datafmt** for the date picker control is set to format the value of **my:field1** to be a short date without time. The **xd:datafmt** for the textbox control is set to format the value of **my:field1** to be time without seconds and no date information.

```
<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;datetime&quot;,&quot;dateFormat:Short
Date;timeFormat:none;&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;datetime&quot;,&quot;dateFormat:Short
Date;timeFormat:none;&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="0">
        <img src="res://infopath.exe/calendar.gif"/>
    </button>
</div>
<span class="xdTextBox xdBehavior_Formatting" hideFocus="1" title="" contentEditable="true"
xd:CtrlId="CTRL2" xd:xctname="PlainText"
xd:datafmt="&quot;datetime&quot;,&quot;dateFormat:none;noSeconds:1;&quot;"
xd:boundProp="xd:num" xd:binding="my:field1" tabIndex="0" style="WIDTH: 130px">
    <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:none;noSeco
nds:1;&quot;)"/>
        </xsl:when>
        <xsl:otherwise>
            <xsl:value-of select="my:field1"/>
        </xsl:otherwise>
    </xsl:choose>
</span>
```

3.4.2.4 External Item Picker Control

The following XSL example is for external item picker controls, as specified in section 2.3.2.4. This example is an external item picker control with conditional formatting. Conditional formatting is set such that if the value of **my:field1** is "false", the control is disabled.

```
<object class="xdActiveX" hideFocus="1" style="WIDTH: 288px; HEIGHT: 23px"
classid="clsid:ad74fc20-e09f-4e47-8a87-1da49930867a" tabIndex="0" xd:boundProp="xd:inline"
xd:bindingProperty="InfoPathValue" xd:bindingType="xmlNode" xd:server="http://"
xd:CtrlId="CTRL1" xd:xctname="entitypicker" tabStop="true" contentEditable="false"
xd:binding="my:group">
    <xsl:if test="function-available('xdImage:getImageUrl')">
        <xsl:attribute name="src"><xsl:value-of
select="xdImage:getImageUrl(my:group)"/></xsl:attribute>
    </xsl:if>
    <xsl:attribute name="style">WIDTH: 288px; HEIGHT: 23px;<xsl:choose>
            <xsl:when test="my:field1 = string(false())">caption: Rule 1</xsl:when>
        </xsl:choose>
    </xsl:attribute>
    <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="Calibri,11,0,400,0,0,0"/>
    <param NAME="Caption" VALUE="Entity..."/>
    <param NAME="EntityNamespace" VALUE="EntityNamespace"/>
    <param NAME="EntityName" VALUE="EntityName"/>
    <param NAME="EntityFinderName" VALUE="Entity Item List"/>
    <param NAME="BDCInstanceName" VALUE=""/>
    <param NAME="DisplayFieldName" VALUE="Name"/>
    <param NAME="SystemInstanceName" VALUE="SystemInstance"/>
    <param NAME="EntityBDCServerURL" VALUE=""/>
    <param NAME="DefaultQuery" VALUE=""/>
    <param NAME="AssociationName" VALUE=""/>
    <param NAME="EIRLocation" VALUE=""/>
    <param NAME="DisplayNameLocation" VALUE=""/>
    <param NAME="Id1Location" VALUE=""/>
    <param NAME="Id2Location" VALUE=""/>
    <param NAME="Id3Location" VALUE=""/>
    <param NAME="Id4Location" VALUE=""/>
    <param NAME="Id5Location" VALUE=""/>
    <param NAME="PickerDialogTitle" VALUE=""/>
    <param NAME="BackgroundColor" VALUE="2147483653"/>
    <param NAME="MaxLines" VALUE="4"/>
    <param NAME="Direction" VALUE="0"/>
    <param NAME="MaxResults" VALUE="100"/>
    <param NAME="EntityButtonWidth" VALUE="0"/>
    <param NAME="QueryRequired" VALUE="0"/>
    <param NAME="RefreshOnOpen" VALUE="0"/>
    <param NAME="PickerTargetMode" VALUE="3"/>
    <param NAME="MultiItem" VALUE="0"/>
</object>
```

3.4.2.5 Embedded Picture Control

The following XSL examples are for embedded picture controls, as specified in section 2.3.2.5.

The following example is an embedded picture control displaying the picture stored in **my:field1**. When the user hovers over the control with the cursor, it displays the message "This is an embedded picture control" because this is the value set for the **title** attribute.

```
<xsl:if test="function-available('xdImage:getImageUrl')">
    <img class="xdInlinePicture" hideFocus="1" title="This is an embedded picture control"
alt="Click here to insert a picture" tabStop="true" xd:boundProp="" xd:binding="my:field1"
tabIndex="0" xd:xctname="InlineImage" xd:CtrlId="CTRL1" xd:inline="my:field1"
src="{xdImage:getImageUrl(my:field1)}"/>
</xsl:if>
```

The following example is an embedded picture control displaying the image stored in **my:field1**. The conditional formatting is set such that if **my:field1** is not empty, the control is disabled.

```
<xsl:if test="function-available('xdImage:getImageUrl')">
    <img class="xdInlinePicture" hideFocus="1" alt="Click here to insert a picture"
tabStop="true" xd:boundProp="" xd:binding="my:field1" tabIndex="0" xd:xctname="InlineImage"
xd:CtrlId="CTRL1" xd:inline="my:field1" src="{xdImage:getImageUrl(my:field1)}">
        <xsl:attribute name="style">
```

```
            <xsl:choose>
                <xsl:when test="my:field1 != &quot;&quot;">caption: Rule 1</xsl:when>
            </xsl:choose>
        </xsl:attribute>
        <xsl:choose>
            <xsl:when test="my:field1 != &quot;&quot;">
                <xsl:attribute name="disabled">true</xsl:attribute>
            </xsl:when>
        </xsl:choose>
    </img>
</xsl:if>
```

The following example is an embedded picture control displaying the image stored in **my:field1** with a black border. The conditional formatting is set such that if **my:field1** is empty, the control has a red background color.

```
<xsl:if test="function-available('xdImage:getImageUrl')">
    <img class="xdInlinePicture" hideFocus="1" style="BORDER-RIGHT: #000000 1pt solid;
BORDER-TOP: #000000 1pt solid; BORDER-LEFT: #000000 1pt solid; BORDER-BOTTOM: #000000 1pt
solid" alt="Click here to insert a picture" tabStop="true" xd:boundProp=""
xd:binding="my:field1" tabIndex="0" xd:xctname="InlineImage" xd:CtrlId="CTRL1"
xd:inline="my:field1" src="{xdImage:getImageUrl(my:field1)}">
        <xsl:attribute name="style">BORDER-RIGHT: #000000 1pt solid; BORDER-TOP: #000000 1pt
solid; BORDER-LEFT: #000000 1pt solid; BORDER-BOTTOM: #000000 1pt solid;
            <xsl:choose>
                <xsl:when test="my:field1 = &quot;&quot;">BACKGROUND-COLOR: #ff0000; caption:
Rule 1</xsl:when>
            </xsl:choose>
        </xsl:attribute>
    </img>
</xsl:if>
```

3.4.2.6 Hyperlink Input Control

The following XSL examples are for hyperlink input controls, as specified in section 2.3.2.6.

The following example only allows the user to specify a URL in a hyperlink input control.

```
<span class="xdHyperlinkBox xdBehavior_Formatting xdHyperlinkBoxClickable" tabStop="true"
xd:boundProp="href"
xd:binding="my:field1" tabIndex="-1" xd:CtrlId="CTRL1" xd:xctname="HyperlinkBox" style="FONT-
SIZE: 8pt; FONT-FAMILY: Tahoma">
    <button class="xdHyperlinkBoxButtonClickable" tabIndex="0">
        <img style="vertical-align:bottom" src="res://infopath.exe/hyperlinkbox.png"/>
    </button>
    <span style="width:5px;"/>
    <xsl:choose>
        <xsl:when test="string-length(my:field1)!=0">
            <A class="hyperlinkAnchor" tabIndex="0" style="">
                <xsl:attribute name="title">
                    <xsl:value-of select="my:field1"/>
                </xsl:attribute>
                <xsl:attribute name="href">
                    <xsl:value-of select="my:field1"/>
                </xsl:attribute>
```

```
                <xsl:value-of select="substring(normalize-space(my:field1), 0, 256)"/>
            </A>
        </xsl:when>
        <xsl:otherwise>Click here to insert a hyperlink</xsl:otherwise>
    </xsl:choose>
</span>
```

The following example allows the user to specify a URL in a hyperlink input control, as well as the text for the hyperlink. Font size and color are also specified.

```
<span class="xdHyperlinkBox xdBehavior_Formatting xdHyperlinkBoxClickable"
xd:binding_secondary="my:field1/@my:field2" xd:boundPropSecondary="displaytext"
tabStop="true" xd:boundProp="href" xd:binding="my:field1" tabIndex="-1" xd:CtrlId="CTRL1"
xd:xctname="HyperlinkBox" style="FONT-WEIGHT: bold; FONT-SIZE: small; COLOR: #ff0000; FONT-
FAMILY: Tahoma">
    <button class="xdHyperlinkBoxButtonClickable" tabIndex="0">
        <img style="vertical-align:bottom" src="res://infopath.exe/hyperlinkbox.png"/>
    </button>
    <span style="width:5px;"/>
    <xsl:choose>
        <xsl:when test="string-length(my:field1)!=0">
            <A class="hyperlinkAnchor" tabIndex="0" style="COLOR: #ff0000;">
                <xsl:attribute name="title">
                    <xsl:value-of select="my:field1"/>
                </xsl:attribute>
                <xsl:attribute name="href">
                    <xsl:value-of select="my:field1"/>
                </xsl:attribute>
                <xsl:choose>
                    <xsl:when test="string-length(normalize-space(my:field1/@my:field2))=0">
                        <xsl:value-of select="substring(normalize-space(my:field1), 0,
256)"/>
                    </xsl:when>
                    <xsl:otherwise>
                        <xsl:value-of select="substring(normalize-
space(my:field1/@my:field2), 0, 256)"/>
                    </xsl:otherwise>
                </xsl:choose>
            </A>
        </xsl:when>
        <xsl:otherwise>Click here to insert a hyperlink</xsl:otherwise>
    </xsl:choose>
</span>
```

The following example allows the user to specify a URL in a hyperlink input control, as well as text of the hyperlink. Additionally, conditional formatting is set such that if the value of **my:field1/@my:field2** is "Contoso", the control is disabled with white text and a black background.

```
<span class="xdHyperlinkBox xdBehavior_Formatting xdHyperlinkBoxClickable" tabIndex="-1"
xd:xctname="HyperlinkBox" xd:CtrlId="CTRL1" xd:binding="my:field1" xd:boundProp="href"
tabStop="true" xd:boundPropSecondary="displaytext"
xd:binding_secondary="my:field1/@my:field2">
    <xsl:attribute name="style">FONT-SIZE: small; FONT-FAMILY: Arial;<xsl:choose>
```

```
            <xsl:when test="my:field1/@my:field2 = &quot;Contoso&quot;">COLOR: #ffffff;
BACKGROUND-COLOR: #000000; caption: Rule 1</xsl:when>
        </xsl:choose>
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="my:field1/@my:field2 = &quot;Contoso&quot;">
            <xsl:attribute name="xd:disableEditing">yes</xsl:attribute>
        </xsl:when>
    </xsl:choose>
    <button class="xdHyperlinkBoxButtonClickable" tabIndex="0">
        <img style="vertical-align:bottom" src="res://infopath.exe/hyperlinkbox.png"/>
    </button>
    <span style="width:5px;"/>
    <xsl:choose>
        <xsl:when test="string-length(my:field1)!=0">
            <A class="hyperlinkAnchor" tabIndex="0">
                <xsl:attribute name="style">
                    <xsl:choose>
                        <xsl:when test="my:field1/@my:field2 = &quot;Contoso&quot;">COLOR:
#ffffff; BACKGROUND-COLOR: #000000; caption: Rule 1</xsl:when>
                    </xsl:choose>
                </xsl:attribute>
                <xsl:choose>
                    <xsl:when test="my:field1/@my:field2 = &quot;Contoso&quot;">
                        <xsl:attribute name="xd:disableEditing">yes</xsl:attribute>
                    </xsl:when>
                </xsl:choose>
                <xsl:attribute name="title">
                    <xsl:value-of select="my:field1"/>
                </xsl:attribute>
                <xsl:attribute name="href">
                    <xsl:value-of select="my:field1"/>
                </xsl:attribute>
                <xsl:choose>
                    <xsl:when test="string-length(normalize-space(my:field1/@my:field2))=0">
                        <xsl:value-of select="substring(normalize-space(my:field1), 0,
256)"/>
                    </xsl:when>
                    <xsl:otherwise>
                        <xsl:value-of select="substring(normalize-
space(my:field1/@my:field2), 0, 256)"/>
                    </xsl:otherwise>
                </xsl:choose>
            </A>
        </xsl:when>
        <xsl:otherwise>Click here to insert a hyperlink</xsl:otherwise>
    </xsl:choose>
</span>
```

3.4.2.7 List Controls (Bulleted List Control, Numbered List Control and Plain List Control)

The following XSL examples are for list controls, as specified in section 2.3.2.7.

The following example is a plain list control with no conditional formatting.

```
<span style="WIDTH: 651px; MARGIN-BOTTOM: 1px; " class="xdRepeating" title=""
xd:xctname="PlainList">
    <ol style="LIST-STYLE-TYPE: none; MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px">
        <xsl:for-each select="my:group1/my:field1">
            <li>
                <span hideFocus="1" class="xdListItem" xd:xctname="ListItem_Plain"
tabIndex="0" xd:binding="." xd:CtrlId="CTRL1" style="WIDTH: 100%">
                    <xsl:value-of select="."/>
                </span>
            </li>
        </xsl:for-each>
    </ol>
</span>
```

The following example is a bulleted list control with conditional formatting. The bulleted list control outputs a link with the text "Insert Item" that adds an additional list item to the list (1) after clicking this link. Conditional formatting is set such that if the value of **my:field1** is "a", the control has a red background color.

```
<span style="WIDTH: 651px; MARGIN-BOTTOM: 1px; " class="xdRepeating" title=""
xd:xctname="bulletedlist">
    <ol style="LIST-STYLE-TYPE: square; MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px">
        <xsl:for-each select="my:group1/my:field1">
            <li>
                <span hideFocus="1" class="xdListItem" xd:xctname="ListItem_Plain"
tabIndex="0" xd:binding="." xd:CtrlId="CTRL1">
                    <xsl:attribute name="style">WIDTH: 100%;
                        <xsl:choose>
                            <xsl:when test=". = &quot;a&quot;">BACKGROUND-COLOR: #ff0000;
caption: Rule 1</xsl:when>
                        </xsl:choose>
                    </xsl:attribute>
                    <xsl:value-of select="."/>
                </span>
            </li>
        </xsl:for-each>
    </ol>
</span>
<div class="optionalPlaceholder" xd:xmlToEdit="field1_1" tabIndex="0"
xd:action="xTextList::insert" style="MARGIN-LEFT: 40px; WIDTH: 651px">Insert item</div>
```

The following example is a numbered list control with lower case Roman numbering, a background color set to green, and conditional formatting. Conditional formatting is set such that if the value of **my:field1** is "a", the control is disabled.

```
<span style="BACKGROUND-COLOR: #00ff00; WIDTH: 651px; MARGIN-BOTTOM: 1px; "
class="xdRepeating" title="" xd:xctname="numberedlist">
    <ol style="LIST-STYLE-TYPE: lower-roman; MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px">
        <xsl:for-each select="my:group2/my:field2">
            <li>
                <span hideFocus="1" class="xdListItem" xd:xctname="ListItem_Plain"
tabIndex="0" xd:binding="." xd:CtrlId="CTRL2">
                    <xsl:attribute name="style">WIDTH: 100%;
                        <xsl:choose>
                            <xsl:when test=". = &quot;a&quot;">caption: Rule 1</xsl:when>
```

```
                    </xsl:choose>
                </xsl:attribute>
                <xsl:choose>
                    <xsl:when test=". = &quot;a&quot;">
                        <xsl:attribute name="contentEditable">false</xsl:attribute>
                    </xsl:when>
                </xsl:choose>
                <xsl:value-of select="."/>
            </span>
        </li>
    </xsl:for-each>
</ol>
</span>
```

3.4.2.8 Linked Picture Control

The following XSL examples are for linked picture controls, as specified in section 2.3.2.8.

The following example is a linked picture control that allows the user to specify a URL and is bound to **my:field1**.

```
<img hideFocus="1" class="xdLinkedPicture" alt="Click here to insert a picture"
displaytext="" tabStop="true" xd:boundProp="src" xd:binding="my:field1" tabIndex="0"
xd:xctname="LinkedImage" xd:CtrlId="CTRL1">
    <xsl:attribute name="src">
        <xsl:value-of select="my:field1"/>
    </xsl:attribute>
</img>
```

The following example is a linked picture control that allows the user to specify a URL, stored in **my:field1**, as well as a description of the picture that is stored in **my:field1/@my:field2**.

```
<img hideFocus="1" class="xdLinkedPicture" xd:boundPropSecondary="displaytext"
xd:binding_secondary="my:field1/@my:field2" tabStop="true" xd:boundProp="src"
xd:binding="my:field1" tabIndex="0" xd:xctname="LinkedImage" xd:CtrlId="CTRL2">
    <xsl:attribute name="src">
        <xsl:value-of select="my:field1"/>
    </xsl:attribute>
    <xsl:attribute name="displaytext">
        <xsl:value-of select="my:field1/@my:field2"/>
    </xsl:attribute>
    <xsl:attribute name="alt">
        <xsl:choose>
            <xsl:when test="string-length(my:field1) &gt; 0">
                <xsl:value-of select="my:field1/@my:field2"/>
            </xsl:when>
            <xsl:otherwise>Click here to insert a picture</xsl:otherwise>
        </xsl:choose>
    </xsl:attribute>
</img>
```

The following example is a linked picture control that allows the user to specify a URL, stored in **my:field1**, as well as a description of the picture that is stored in **my:field1/@my:field2**.

Additionally, conditional formatting is set such that if the value of **my:field1/@my:field2** is "2", the control is hidden.

```
<img hideFocus="1" class="xdLinkedPicture" xd:boundPropSecondary="displaytext"
xd:binding_secondary="my:field1/@my:field2" tabStop="true" xd:boundProp="src"
xd:binding="my:field1" tabIndex="0" xd:xctname="LinkedImage" xd:CtrlId="CTRL2">
    <xsl:attribute name="style">
        <xsl:choose>
            <xsl:when test="my:field1/@my:field2 = &quot;2&quot;">DISPLAY: none; caption:
Rule 1</xsl:when>
        </xsl:choose>
    </xsl:attribute>
    <xsl:attribute name="src">
        <xsl:value-of select="my:field1"/>
    </xsl:attribute>
    <xsl:attribute name="displaytext">
        <xsl:value-of select="my:field1/@my:field2"/>
    </xsl:attribute>
    <xsl:attribute name="alt">
        <xsl:choose>
            <xsl:when test="string-length(my:field1) &gt; 0">
                <xsl:value-of select="my:field1/@my:field2"/>
            </xsl:when>
            <xsl:otherwise>Click here to insert a picture</xsl:otherwise>
        </xsl:choose>
    </xsl:attribute>
</img>
```

3.4.2.9 Multiple-Selection List Box Control

The following XSL examples are for multiple-selection list box controls, as specified in section 2.3.2.9.

The following example is a multiple-selection list box control that allows the user to select multiple values from a list of options that are in the XSL. It is an example of the production **MSLB_WITH_STATIC_CHOICES**, as specified in section 2.4.1.21.9.

```
<xsl:choose>
    <xsl:when test="function-available('ipApp:GetMajorVersion') and ipApp:GetMajorVersion()
&gt;= 12">
        <span style="WIDTH: 130px; HEIGHT: 100px" class="xdMultiSelectList" title=""
xd:xctname="multiselectlistbox" xd:CtrlId="CTRL1" tabIndex="-1" xd:ref="my:group1/my:field1">
            <span class="xdMultiSelectListItem">
                <input type="checkbox" title="abc" xd:onValue="abc" xd:boundProp="xd:value"
xd:xctname="CheckBox" tabIndex="0">
                    <xsl:attribute name="xd:value">
                        <xsl:value-of select="my:group1/my:field1[.=&quot;abc&quot;][1]"/>
                    </xsl:attribute>
                    <xsl:attribute name="xd:binding">
                        <xsl:value-of select="my:group1/my:field1[.=&quot;abc&quot;][1]"/>
                    </xsl:attribute>
                    <xsl:if test="my:group1/my:field1=&quot;abc&quot;">
                        <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
                    </xsl:if>
                </input>abc
            </span>
```

```
            <span class="xdMultiSelectListItem">
                <input type="checkbox" title="def" xd:onValue="def" xd:boundProp="xd:value"
xd:xctname="CheckBox" tabIndex="0">
                    <xsl:attribute name="xd:value">
                        <xsl:value-of select="my:group1/my:field1[.=&quot;def&quot;][1]"/>
                    </xsl:attribute>
                    <xsl:attribute name="xd:binding">
                        <xsl:value-of select="my:group1/my:field1[.=&quot;def&quot;][1]"/>
                    </xsl:attribute>
                    <xsl:if test="my:group1/my:field1=&quot;def&quot;">
                        <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
                    </xsl:if>
                </input>def
            </span>
            <xsl:for-each select="my:group1/my:field1[.!=&quot;abc&quot; and
.!=&quot;def&quot;]">
                <xsl:if test="normalize-space(.)!=''">
                    <span class="xdMultiSelectListItem">
                        <input type="checkbox" CHECKED="CHECKED" xd:onValue="{.}"
xd:boundProp="xd:value" xd:binding="." xd:xctname="CheckBox" tabIndex="0">
                            <xsl:attribute name="xd:value">
                                <xsl:value-of select="."/>
                            </xsl:attribute>
                            <xsl:attribute name="title">
                                <xsl:value-of select="."/>
                            </xsl:attribute>
                        </input>
                        <xsl:value-of select="."/>
                    </span>
                </xsl:if>
            </xsl:for-each>
        </span>
    </xsl:when>
    <xsl:otherwise>
        <span class="xdRepeating" xd:xctname="BulletedList" title="" xd:CtrlId="CTRL1"
style="WIDTH: 130px; HEIGHT: 100px; HEIGHT: auto;">
            <ol style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; LIST-STYLE-TYPE: disc">
                <xsl:for-each select="my:group1/my:field1">
                    <li>
                        <span class="xdListItem" hideFocus="1" contentEditable="true"
xd:CtrlId="CTRL1" xd:xctname="ListItem_Plain" xd:binding="." style="WIDTH: 130px; HEIGHT:
100px; HEIGHT:auto; WIDTH: 100%; WHITE-SPACE: normal; WORD-WRAP: break-word;" tabIndex="0">
                            <xsl:value-of select="."/>
                        </span>
                    </li>
                </xsl:for-each>
            </ol>
        </span>
    </xsl:otherwise>
</xsl:choose>
```

The following example is a multiple-selection list box control that allows the user to select multiple values from a list of options that are in the XSL. In addition to being able to select options, the user is able to enter custom values. It is an example of the production **MSLB_WITH_STATIC_CHOICES_AND_CUSTOM_ITEMS**, as specified in section 2.4.1.21.9.

```
<xsl:choose>
    <xsl:when test="function-available('ipApp:GetMajorVersion') and ipApp:GetMajorVersion()
&gt;= 12">
        <span style="WIDTH: 130px; HEIGHT: 100px" class="xdMultiSelectList" title=""
xd:xctname="multiselectlistbox" xd:CtrlId="CTRL1" tabIndex="-1" xd:ref="my:group1/my:field1">
            <span class="xdMultiSelectListItem">
                <input type="checkbox" title="abc" xd:onValue="abc" xd:boundProp="xd:value"
xd:xctname="CheckBox" tabIndex="0">
                    <xsl:attribute name="xd:value">
                        <xsl:value-of select="my:group1/my:field1[.=&quot;abc&quot;][1]"/>
                    </xsl:attribute>
                    <xsl:attribute name="xd:binding">
                        <xsl:value-of select="my:group1/my:field1[.=&quot;abc&quot;][1]"/>
                    </xsl:attribute>
                    <xsl:if test="my:group1/my:field1=&quot;abc&quot;">
                        <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
                    </xsl:if>
                </input>abc
            </span>
            <span class="xdMultiSelectListItem">
                <input type="checkbox" title="def" xd:onValue="def" xd:boundProp="xd:value"
xd:xctname="CheckBox" tabIndex="0">
                    <xsl:attribute name="xd:value">
                        <xsl:value-of select="my:group1/my:field1[.=&quot;def&quot;][1]"/>
                    </xsl:attribute>
                    <xsl:attribute name="xd:binding">
                        <xsl:value-of select="my:group1/my:field1[.=&quot;def&quot;][1]"/>
                    </xsl:attribute>
                    <xsl:if test="my:group1/my:field1=&quot;def&quot;">
                        <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
                    </xsl:if>
                </input>def
            </span>
            <xsl:for-each select="my:group1/my:field1[.!=&quot;abc&quot; and
.!=&quot;def&quot;]">
                <span class="xdMultiSelectListItem">
                    <input type="checkbox" CHECKED="CHECKED" xd:onValue="{.}"
xd:boundProp="xd:value" xd:binding="." xd:xctname="CheckBox" tabIndex="0">
                        <xsl:attribute name="xd:value">
                            <xsl:value-of select="."/>
                        </xsl:attribute>
                        <xsl:attribute name="title">
                            <xsl:value-of select="."/>
                        </xsl:attribute>
                    </input>
                    <span hideFocus="1" contentEditable="true" xd:binding="."
xd:xctname="PlainText" style="WIDTH: 70%;" tabIndex="0" class="xdMultiSelectFillIn">
                        <xsl:attribute name="title">
                            <xsl:value-of select="."/>
                        </xsl:attribute>
                        <xsl:value-of select="."/>
                    </span>
                </span>
            </xsl:for-each>
            <xsl:if test="not(my:group1/my:field1[.!=&quot;abc&quot; and
.!=&quot;def&quot;][1])">
                <span class="xdMultiSelectListItem">
```

```
                    <input type="checkbox" title="" xd:onValue="" xd:boundProp="xd:value"
xd:binding="my:group1/my:field1[.=''][1]" xd:xctname="CheckBox" tabIndex="0">
                        <xsl:attribute name="xd:value">
                            <xsl:value-of select="my:group1/my:field1[.=''][1]"/>
                        </xsl:attribute>
                        <xsl:if test="my:group1/my:field1=''">
                            <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
                        </xsl:if>
                    </input>
                    <span title="" hideFocus="1" xd:xctname="PlainText"
xd:binding="my:group1/my:field1[.=''][1]" style="WIDTH: 70%;" class="xdMultiSelectFillIn">
                        <xsl:attribute name="tabIndex">
                            <xsl:choose>
                                <xsl:when test="@CHECKED='CHECKED'">0</xsl:when>
                                <xsl:otherwise>-1</xsl:otherwise>
                            </xsl:choose>
                        </xsl:attribute>
                    </span>
                </span>
            </xsl:if>
        </span>
    </xsl:when>
    <xsl:otherwise>
        <span class="xdRepeating" xd:xctname="BulletedList" title="" xd:CtrlId="CTRL1"
style="WIDTH: 130px; HEIGHT: 100px; HEIGHT: auto;">
            <ol style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; LIST-STYLE-TYPE: disc">
                <xsl:for-each select="my:group1/my:field1">
                    <li>
                        <span class="xdListItem" hideFocus="1" contentEditable="true"
xd:CtrlId="CTRL1" xd:xctname="ListItem_Plain" xd:binding="." style="WIDTH: 130px; HEIGHT:
100px; HEIGHT:auto; WIDTH: 100%; WHITE-SPACE: normal; WORD-WRAP: break-word;" tabIndex="0">
                            <xsl:value-of select="."/>
                        </span>
                    </li>
                </xsl:for-each>
            </ol>
        </span>
    </xsl:otherwise>
</xsl:choose>
```

The following example is a multiple-selection list box control that allows the user to select multiple values from a list of options that are populated from another location within the form's data source (2). In addition to being able to select options, the user is able to enter custom values. It is an example of the production **MSLB_WITH_DYNAMIC_CHOICES_AND_CUSTOM_ITEMS**, as specified in section 2.4.1.21.9.

```
<xsl:choose>
    <xsl:when test="function-available('ipApp:GetMajorVersion') and ipApp:GetMajorVersion()
&gt;= 12">
        <span style="WIDTH: 130px; HEIGHT: 100px" class="xdMultiSelectList" title=""
xd:xctname="multiselectlistbox" xd:CtrlId="CTRL3" xd:boundProp="value" tabIndex="-1"
xd:ref="my:group1/my:field1">
            <xsl:variable name="values" select="my:group1/my:field1"/>
            <xsl:for-each select="my:group1/my:field2">
                <span class="xdMultiSelectListItem">
```

```
                    <input type="checkbox" xd:boundProp="xd:value" xd:binding="."
xd:xctname="CheckBox" tabIndex="0">
                        <xsl:attribute name="xd:value">
                            <xsl:value-of select="."/>
                        </xsl:attribute>
                        <xsl:attribute name="xd:onValue">
                            <xsl:value-of select="."/>
                        </xsl:attribute>
                        <xsl:attribute name="title">
                            <xsl:value-of select="."/>
                        </xsl:attribute>
                        <xsl:if test=".=$values">
                            <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
                        </xsl:if>
                    </input>
                    <xsl:value-of select="."/>
                </span>
            </xsl:for-each>
            <xsl:variable name="options" select="my:group1/my:field2/."/>
            <xsl:for-each select="my:group1/my:field1[not(.=$options)]">
                <xsl:if test="normalize-space(.)!=''">
                    <span class="xdMultiSelectListItem">
                        <input type="checkbox" CHECKED="CHECKED" xd:onValue="{.}"
xd:boundProp="xd:value" xd:binding="." xd:xctname="CheckBox" tabIndex="0">
                            <xsl:attribute name="xd:value">
                                <xsl:value-of select="."/>
                            </xsl:attribute>
                            <xsl:attribute name="title">
                                <xsl:value-of select="."/>
                            </xsl:attribute>
                        </input>
                        <xsl:value-of select="."/>
                    </span>
                </xsl:if>
            </xsl:for-each>
        </span>
    </xsl:when>
    <xsl:otherwise>
        <span class="xdRepeating" xd:xctname="BulletedList" title="" xd:CtrlId="CTRL3"
xd:boundProp="value" style="WIDTH: 130px; HEIGHT: 100px; HEIGHT: auto;">
            <ol style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; LIST-STYLE-TYPE: disc">
                <xsl:for-each select="my:group1/my:field1">
                    <li>
                        <span class="xdListItem" hideFocus="1" contentEditable="true"
xd:CtrlId="CTRL3" xd:xctname="ListItem_Plain" xd:binding="." style="WIDTH: 130px; HEIGHT:
100px; HEIGHT:auto; WIDTH: 100%; WHITE-SPACE: normal; WORD-WRAP: break-word;" tabIndex="0">
                            <xsl:value-of select="."/>
                        </span>
                    </li>
                </xsl:for-each>
            </ol>
        </span>
    </xsl:otherwise>
</xsl:choose>
```

The following example is a multiple-selection list box control that allows the user to select multiple values from a list of options that are populated from another location within the form's data source

(2). The control has conditional formatting such that if any occurrence of **my:field1** is equal to "Blue", the control has a blue background color. It is an example of the production **MSLB MSLB_WITH_DYNAMIC_CHOICES**, as specified in section 2.4.1.21.9.

```
<xsl:choose>
    <xsl:when test="function-available('ipApp:GetMajorVersion') and ipApp:GetMajorVersion()
&gt;= 12">
        <span style="WIDTH: 130px; HEIGHT: 100px" class="xdMultiSelectList" title=""
xd:xctname="multiselectlistbox" xd:CtrlId="CTRL3" xd:boundProp="value" tabIndex="-1"
xd:ref="my:group1/my:field1">
            <xsl:attribute name="style">
                WIDTH: 130px; HEIGHT: 100px;<xsl:choose>
                    <xsl:when test="my:group1/my:field1[. = &quot;Blue&quot;]">BACKGROUND-
COLOR: #0000ff; caption: Rule 1</xsl:when>
                </xsl:choose>
            </xsl:attribute>
            <xsl:variable name="values" select="my:group1/my:field1"/>
            <xsl:for-each select="my:group1/my:field2">
                <span class="xdMultiSelectListItem">
                    <input type="checkbox" xd:boundProp="xd:value" xd:binding="."
xd:xctname="CheckBox" tabIndex="0">
                        <xsl:attribute name="xd:value">
                            <xsl:value-of select="."/>
                        </xsl:attribute>
                        <xsl:attribute name="xd:onValue">
                            <xsl:value-of select="."/>
                        </xsl:attribute>
                        <xsl:attribute name="title">
                            <xsl:value-of select="."/>
                        </xsl:attribute>
                        <xsl:if test=".=$values">
                            <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
                        </xsl:if>
                    </input>
                    <xsl:value-of select="."/>
                </span>
            </xsl:for-each>
            <xsl:variable name="options" select="my:group1/my:field2/."/>
            <xsl:for-each select="my:group1/my:field1[not(.=$options)]">
                <xsl:if test="normalize-space(.)!=''">
                    <span class="xdMultiSelectListItem">
                        <input type="checkbox" CHECKED="CHECKED" xd:onValue="{.}"
xd:boundProp="xd:value" xd:binding="." xd:xctname="CheckBox" tabIndex="0">
                            <xsl:attribute name="xd:value">
                                <xsl:value-of select="."/>
                            </xsl:attribute>
                            <xsl:attribute name="title">
                                <xsl:value-of select="."/>
                            </xsl:attribute>
                        </input>
                        <xsl:value-of select="."/>
                    </span>
                </xsl:if>
            </xsl:for-each>
        </span>
    </xsl:when>
    <xsl:otherwise>
```

```
        <span class="xdRepeating" xd:xctname="BulletedList" title="" xd:CtrlId="CTRL3"
xd:boundProp="value" style="WIDTH: 130px; HEIGHT: 100px; HEIGHT: auto;">
            <ol style="MARGIN-TOP: 0px; MARGIN-BOTTOM: 0px; LIST-STYLE-TYPE: disc">
                <xsl:for-each select="my:group1/my:field1">
                    <li>
                        <span class="xdListItem" hideFocus="1" contentEditable="true"
xd:CtrlId="CTRL3" xd:xctname="ListItem_Plain" xd:binding="." style="WIDTH: 130px; HEIGHT:
100px; HEIGHT:auto; WIDTH: 100%; WHITE-SPACE: normal; WORD-WRAP: break-word;" tabIndex="0">
                            <xsl:attribute name="style">
                                WIDTH: 130px; HEIGHT: 100px; HEIGHT:auto; WIDTH: 100%; WHITE-
SPACE: normal; WORD-WRAP: break-word;<xsl:choose>
                                    <xsl:when test="../my:field1[. =
&quot;Blue&quot;]">BACKGROUND-COLOR: #0000ff; caption: Rule 1</xsl:when>
                                </xsl:choose>
                            </xsl:attribute>
                            <xsl:value-of select="."/>
                        </span>
                    </li>
                </xsl:for-each>
            </ol>
        </span>
    </xsl:otherwise>
</xsl:choose>
```

3.4.2.10 Picture Button Control

The following XSL examples are for picture button controls, as specified in .section 2.3.2.10

The following is an example of a picture button control with conditional formatting. The **src** attribute is set to image **img1.jpg**. This means that the picture button displays image **img1.jpg** contained in the **form template**. Conditional formatting is set such that if the value of **my:field3** is "true", the control is hidden.

```
<button class="xdPictureButton" contentEditable="false" style="BORDER-RIGHT: medium none;
BORDER-TOP: medium none; BORDER-LEFT: medium none; WIDTH: 215px; BORDER-BOTTOM: medium none;
HEIGHT: 160px" xd:CtrlId="CTRL1_5" xd:xctname="PictureButton" xd:HideInPrintView="true"
tabIndex="0">
    <xsl:attribute name="style">BORDER-RIGHT: medium none; BORDER-TOP: medium none; BORDER-
LEFT: medium none; WIDTH: 215px; BORDER-BOTTOM: medium none; HEIGHT: 160px;
        <xsl:choose>
            <xsl:when test="my:field3 = string(true())">DISPLAY: none; caption: Rule
1</xsl:when>
        </xsl:choose>
    </xsl:attribute>
    <img style="WIDTH: 100%; HEIGHT: 160px; POSITION: static" alt="" src="img1.jpg"
xd:HoverSRC=""/>
</button>
```

The following example is a picture button control that is used to update the form (1) content in the Web browser. The **src** attribute is set to image "img1.jpg". This means that the picture button displays image img1.jpg contained in the **form template**. Conditional formatting is set such that if the value of **my:field2** is "Red", the control has a red background color.

```
<button class="xdPictureButton" contentEditable="false" style="BORDER-RIGHT: medium none;
BORDER-TOP: medium none; BEHAVIOR: url(#default#ActionButton) url(#default#PictureButton);
```

```
BORDER-LEFT: medium none; WIDTH: 215px; BORDER-BOTTOM: medium none; HEIGHT: 160px"
xd:CtrlId="CTRL1_5" xd:xctname="PictureButton" xd:HideInPrintView="true"
xd:action="updateForm" tabIndex="0">    <xsl:attribute name="style">BORDER-RIGHT: medium
none; BORDER-TOP: medium none; BEHAVIOR: url(#default#ActionButton)
url(#default#PictureButton); BORDER-LEFT: medium none; WIDTH: 215px; BORDER-BOTTOM: medium
none; HEIGHT: 160px;<xsl:choose>
            <xsl:when test="not(xdEnvironment:IsBrowser())">DISPLAY: none</xsl:when>
            <xsl:when test="my:field2 = &quot;Red&quot;">BACKGROUND-COLOR: #ff0000; caption:
Rule 1</xsl:when>
        </xsl:choose>
    </xsl:attribute>
    <img style="WIDTH: 100%; HEIGHT: 160px; POSITION: static" alt="" src="2D363A7F.jpg"
xd:HoverSRC="img1.jpg"/>
</button>
```

The following example is a picture button control that is used to submit the form data. The **src** attribute is set to image "img1.jpg". This means that the picture button displays image img1.jpg contained in the form template when the mouse cursor does not hover over the control. The **HoverSRC** attribute is set to image "img2.jpg". This means that the picture button displays image img2.jpg contained in the form template when the mouse cursor hovers over the control. This control has two conditional formatting settings. If the value of **my:field1** is "1", the control is disabled and has a yellow background color. If the first conditional formatting condition is false and if the value of **my:field2** is "abc", the control is hidden. The **HideInPrintView** attribute is set to "false". This means that the picture button is shown when the form (1) is printed.

```
<button class="xdPictureButton" contentEditable="false" style="BORDER-RIGHT: medium none;
BORDER-TOP: medium none; BEHAVIOR: url(#default#ActionButton) url(#default#PictureButton);
BORDER-LEFT: medium none; WIDTH: 215px; BORDER-BOTTOM: medium none; HEIGHT: 160px"
xd:CtrlId="CTRL1_5" xd:xctname="PictureButton" xd:HideInPrintView="false" xd:action="submit"
tabIndex="0">
    <xsl:attribute name="style">BORDER-RIGHT: medium none; BORDER-TOP: medium none; BEHAVIOR:
url(#default#ActionButton) url(#default#PictureButton); BORDER-LEFT: medium none; WIDTH:
215px; BORDER-BOTTOM: medium none; HEIGHT: 160px;
        <xsl:choose>
            <xsl:when test="my:field1 = &quot;1&quot;">BACKGROUND-COLOR: #ffff00; caption:
Rule 1</xsl:when>
            <xsl:when test="my:field2 = &quot;abc&quot;">DISPLAY: none; caption: Rule
2</xsl:when>
        </xsl:choose>
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="my:field1 = &quot;1&quot;">
            <xsl:attribute name="disabled">true</xsl:attribute>
        </xsl:when>
        <xsl:when test="my:field2 = &quot;abc&quot;"/>
    </xsl:choose>
    <img style="WIDTH: 100%; HEIGHT: 160px; POSITION: static" alt="" src="img1.jpg"
xd:HoverSRC="img2.jpg"/>
</button>
```

The following example is a picture button control that is used to refresh the content of a secondary data source. The **src** attribute is set to image "img1.png". This means that the picture button displays image img1.png contained in the form template. Conditional formatting is set such that if the value of **my:field1** is "1", the control is disabled and has a yellow background color.

```
<button class="xdPictureButton" contentEditable="false" style="BORDER-RIGHT: medium none;
BORDER-TOP: medium none; BEHAVIOR: url(#default#ActionButton) url(#default#PictureButton);
BORDER-LEFT: medium none; WIDTH: 215px; BORDER-BOTTOM: medium none; HEIGHT: 160px"
xd:CtrlId="CTRL5_5" xd:xctname="PictureButton" xd:HideInPrintView="true" xd:action="refresh"
xd:auxDom="ChristipTest" tabIndex="0">
    <xsl:attribute name="style">BORDER-RIGHT: medium none; BORDER-TOP: medium none; BEHAVIOR:
url(#default#ActionButton) url(#default#PictureButton); BORDER-LEFT: medium none; WIDTH:
215px; BORDER-BOTTOM: medium none; HEIGHT: 160px;<xsl:choose>
            <xsl:when test="my:field1 = 1">BACKGROUND-COLOR: #ffff00; caption: Rule
1</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>
    <img style="WIDTH: 100%; HEIGHT: 160px; POSITION: static" alt="" src="img1.png"
xd:HoverSRC=""/>
</button>
```

3.4.2.11 SharePoint File Attachment Control

The following XSL example is for a SharePoint file attachment control, as specified in section 2.3.2.11, that is bound to **dfs:dataFields/my:SharePointListItem_RW/my:Attachments**.

```
<span class="xdSharePointFileAttachment" title="" style="WIDTH: 100%"
xmlns:xd="http://schemas.microsoft.com/office/infopath/2003"
xd:xctname="SharePointFileAttachment" xd:CtrlId="CTRL16"
xd:binding="dfs:dataFields/my:SharePointListItem_RW/my:Attachments" xd:disableEditing="yes">
    <SPAN tabindex="0"/>
    <div>
        <xsl:for-each xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
select="dfs:dataFields/my:SharePointListItem_RW/my:Attachments/attachmentURL">
            <div class="xdAttachItem" title="" xd:xctname="SharePointAttachItem"
xd:disableEditing="yes">
                <SPAN STYLE="width:32px" xd:binding="." xd:disableEditing="yes"
tabindex="0"/>
                <a>
                    <xsl:variable name="IsLocal" select="xdXDocument:GetNamedNodeProperty(.,
'IsLocal', '')"/>
                    <xsl:if test="$IsLocal!='true'">
                        <xsl:attribute name="href">
                            <xsl:value-of select="."/>
                        </xsl:attribute>
                    </xsl:if>
                    <xsl:value-of select="xdXDocument:GetNamedNodeProperty(.,
'FileAttachURL', '')"/>
                </a>
            </div>
        </xsl:for-each>
    </div>
</span>
```

3.4.3 Control-Specific Attributes

xd:action

The following XSLT fragment is an example of a button control with a submit **action**, as specified in section 2.4.2.1.

```
<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 that allows insertion of sections.

```
<xsl:apply-templates select="my:group3/my:group4" mode="_2"/>
<div class="optionalPlaceholder" xd:xmlToEdit="group4_3" tabIndex="0"
xd:action="xCollection::insert" align="left" style="WIDTH: 651px">Insert item</div>
```

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 contact selector control with a **bindingProperty** attribute set to "Value".

```
<object class="xdActiveX" hideFocus="1" style="WIDTH: 288px; HEIGHT: 23px"
classid="clsid:61e40d31-993d-4777-8fa0-19ca59b6d0bb" tabIndex="0" tabStop="true"
xd:xctname="{{61e40d31-993d-4777-8fa0-19ca59b6d0bb}}" xd:CtrlId="CTRL1"
xd:server="http://server" xd:bindingType="xmlNode" xd:bindingProperty="Value"
xd:boundProp="xd:inline" xd:AllowMultiple="true" xd:SearchPeopleOnly="true"
xd:SharePointGroup="0" contentEditable="false" xd:binding="my:group1">
...
</object>
```

xd:bindingType

The following XSLT fragment is an example of a contact selector control with a **bindingType** attribute set to "text".

```
<object class="xdActiveX" hideFocus="1" style="WIDTH: 288px; HEIGHT: 23px"
classid="clsid:61e40d31-993d-4777-8fa0-19ca59b6d0bb" tabIndex="0" tabStop="true"
xd:xctname="{{61e40d31-993d-4777-8fa0-19ca59b6d0bb}}" xd:CtrlId="CTRL1"
xd:server="http://server" xd:bindingType="xmlNode" xd:bindingProperty="Value"
xd:boundProp="xd:inline" xd:AllowMultiple="true" xd:SearchPeopleOnly="true"
xd:SharePointGroup="0" contentEditable="false" xd:binding="my:group1">
...
</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="&quot;currency&quot;,&quot;numDigits:0;negativeOrder:0;positiveOrder:0;currencyLo
cale: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: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="&amp;quot;currency&amp;quot;,&amp;quot;numDigits:0;negativeOrder:0;positiveOrder:0;currencyLo
cale:1033;&amp;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>
        <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 a contact selector control with the **enabledProperty** attribute set to "Enabled".

```
<object class="xdActiveX" hideFocus="1" style="WIDTH: 288px; HEIGHT: 23px"
classid="clsid:61e40d31-993d-4777-8fa0-19ca59b6d0bb" tabIndex="0" tabStop="true"
xd:xctname="{{61e40d31-993d-4777-8fa0-19ca59b6d0bb}}" xd:CtrlId="CTRL1"
xd:server="http://server" xd:bindingType="xmlNode" xd:bindingProperty="Value"
xd:boundProp="xd:inline" xd:AllowMultiple="true" xd:SearchPeopleOnly="true"
xd:SharePointGroup="0" contentEditable="false" xd:binding="my:group1">
...
```

```
</object>
```

xd:enabledValue

The following XSLT fragment is an example of a contact selector with the **enabledValue** attribute set to "true".

```
<object class="xdActiveX" hideFocus="1" style="WIDTH: 288px; HEIGHT: 23px"
classid="clsid:61e40d31-993d-4777-8fa0-19ca59b6d0bb" tabIndex="0" tabStop="true"
xd:xctname="{{61e40d31-993d-4777-8fa0-19ca59b6d0bb}}" xd:CtrlId="CTRL1"
xd:server="http://server" xd:bindingType="xmlNode" xd:bindingProperty="Value"
xd:boundProp="xd:inline" xd:AllowMultiple="true" xd:SearchPeopleOnly="true"
xd:SharePointGroup="0" contentEditable="false" xd:binding="my:group1">
...
</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">
        <img src="res://infopath.exe/calendar.gif"/>
    </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;currencyLo
cale: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 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=&quot;true&quot;">
        <xsl:attribute name="CHECKED">CHECKED</xsl:attribute>
    </xsl:if>
</input> Field 4
```

xd:xctname

The following XSLT fragment is an example of a text box control.

```
<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: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>
...
```

3.4.4 XSL Function Extensions

The following examples demonstrate the use of a selection of function extensions. XSL function extensions are specified in section 2.4.3.

The following 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: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>
```

Example 1: Conditional formatting with xdMath:Avg

xdMath:Avg can be used to conditionally change the **style** attribute of an HTML tag.

```
<xsl:attributename="style">WIDTH: 130px;
<xsl:choose>
<xsl:when test="my:Average = xdMath:Avg(my:group1/my:group2)">FONT-WEIGHT: bold
</xsl:when>
</xsl:choose>
</xsl:attribute>
```

The **xdMath:Avg** function is used to calculate the average of the values in the **my:group2** nodes, which is 6+12+15 / 3 = 11. Because this value is equal to the value of the **my:Average** node, the test evaluates to true and **xsl:when** outputs "FONT-WEIGHT: bold" and makes the font bold.

Example 2: Conditional formatting with xdDate:AddDays, xdDate:Today

This example is similar to the first example in that it shows how to perform conditional formatting using XSL function extensions.

```
<xsl:attributename="style">
    <xsl:choose>
        <xsl:when test = "my:Date = xdDate:AddDays(xdDate:Today() , 3)">FONT-WEIGHT: bold;
COLOR: #ff0000</xsl:when>
    </xsl:choose>
</xsl:attribute>
```

The **xsl:when** clause is used to test if the date in the **my:Date** node is three days past today. **xdDate:Today()** is used to get today's date and the output is passed in to the **AddDays** function. **AddDays** adds three days to today's date and outputs the date three days past today. In the SampleXML1.xml snippet, **my:Date** is "2008-02-04". If today is 2008-02-01, **test** evaluates to "true" and the **style** attribute has the value "FONT-WEIGHT: bold; COLOR: #ff0000".

Example 3: Outputting time with xdDate:AddSeconds

```
<xsl:value-of select="xdDate:AddSeconds(xdDate:Now(), my:Seconds)"/>
```

The **xsl:value-of** clause outputs the value returned by the expression in the **select** attribute. **xdDate:AddSeconds** returns the time equivalent to now plus the number of seconds indicated in **my:Seconds**. **xsl:value-of** outputs the resulting time.

Example 4: Conditional formatting with xdUtil:Match

```
<span class="xdTextBox" hideFocus="1" title="" xd:CtrlId="CTRL12" xd:xctname="PlainText"
tabIndex="0" xd:binding="my:ZipCode">
    <xsl:if test="function-available('xdXDocument:GetDOM')">
        <xsl:attribute name="style">
            <xsl:choose>
                <xsl:when test = "not( xdUtil:Match( string(my:ZipCode),
&amp;quot;\d\d\d\d\d&amp;quot; ) )">COLOR: #ff0000; TEXT-DECORATION: line-through
                </xsl:when>
            </xsl:choose>
        </xsl:attribute>
        <xsl:value-of select="my:ZipCode"/>
    </xsl:if>
</span>
```

In this example, the XSL representation of a text box control is given. In the example, this textbox is used for entering zip codes. To make users aware of the case where an invalid zip code has been entered, the **xdUtil:Match** function is used to check that the entered value conforms to the zip code pattern. The first parameter of the **xdUtil:Match** function takes the value of the **my:ZipCode** node. The second parameter is the regular expression "\d\d\d\d\d", which means five consecutive digits. If the given zip code is not composed of five digits, the textbox has the **style** "COLOR: #ff0000; TEXT-DECORATION: line-through" to make it more visible to the user that the entered zip code is not valid. In SampleXML1.xml, **my:ZipCode** is 98052x, which doesn't match the given regular expression. Therefore, **test**l evaluates to "true" and the text box control's **style** attribute has 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>.
```

Example 5: Getting data from a secondary data source using xdXDocument:GetDOM

The following example uses SampleXML2.xml as the secondary data source of a form template (.xsn) file. This secondary data source is registered with a data source name "data". The data is gathered from this data source (2) for use within the XSL.

The font of a text box is bold if the average of the values in the secondary data source is less than 15.

```
<spanclass="xdTextBox"hideFocus="1"title=""tabIndex="0"xd:binding="my:field2"xd:xctname="Plai
nText"xd:CtrlId="CTRL3">
    <xsl:if test="function-available('xdXDocument:GetDOM')">
        <xsl:attribute name="style">WIDTH: 130px;
            <xsl:choose>
                <xsl:when test="xdMath:Avg( xdXDocument:getDOM(&amp;quot;data&amp;quot;)/root/value
)">FONT-WEIGHT: bold; caption: Conditional Formatting 1
                </xsl:when>
```

```
            </xsl:choose>
        </xsl:attribute>
        <xsl:value-ofselect="my:field2"/>
    </xsl:if>
</span>
```

In the preceding XSL snippet, the secondary data source content is queried using **xdXDocument:getDom** and then an XPath expression is built on it. This XPath expression returns a **node-set** containing all the **value** nodes in the **example** data source. Then **xdMath:Avg** is used to calculate the average.

The following 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>
```

Example 6: 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>
```

This is the XSL example for an expression box control. The value of the expression box is calculated from SampleXML3.xml using **xdMath:Eval**. The XPath expression passed to **xdMath:Eval** returns a **node-set** containing all the **my:group2** nodes under **my:group1**. **my:group2** nodes contain the child elements **my:Count** and **my:Value**. The expression passed in as the second parameter to the **xdMath:Eval** calculates the sum of the multiplication of the nodes from the first parameter. **xdMath:Eval** calculates this multiplication for every **group2** node and returns the result as a **node-set**.

Count	Value
3	4
12	20
4	8

The output is 3*4 + 12 * 20 + 4*8 = 284

Example 7: Displaying a dynamic hyperlink using xdServerInfo:get-SharePointSiteUrl

The following example uses **xdServerInfo:get-SharePointListUrl** to display a hyperlink that is updated based on the location of the form template.

The example assumes that a server named **serverA** has two sites (2) named **siteA** and **siteB**. Both of these sites (2) have the file hardwareList.html in their root folders. The goal is to write the XSLT of a form template that has a hyperlink to "hardwareList.html" on the site (2) where the form template is hosted.

```
<div>For hardware list please click the following link:
    <span hideFocus="1" style="TEXT-ALIGN: left; WIDTH: 130px; OVERFLOW: visible"
class="xdHyperlink" title="" tabIndex="0" xd:xctname="hyperlink">
        <a class="xdDataBindingUI" xd:CtrlId="CTRL4" xd:disableEditing="yes">
            <xsl:if test="function-available('xdXDocument:GetDOM')">
                <xsl:attribute name="href">
                    <xsl:value-of select="concat(xdServerInfo:get-SharePointSiteUrl(),
&amp;quot;hardwareList.html&amp;quot;)"/>
                </xsl:attribute>
                <xsl:value-of select="concat(xdServerInfo:get-SharePointSiteUrl(),
&amp;quot;hardwareList.html&amp;quot;)"/>
            </xsl:if>
        </a>
    </span>
</div>
```

If the form template with the preceding XSLT excerpt is hosted at **siteA**, which is located at **http://serverA/siteA/**, the view has the following text and hyperlink:

For hardware list please click the following link: http://serverA/siteA/hardwareList.html

On the other hand, if the same form template is hosted at **siteB**, which is located at **http://serverA/siteB**, the view has the following text and hyperlink:

For hardware list please click the following link: http://serverA/siteB/hardwareList.html

3.5 Print View Files (XSLT) Examples

The following example shows how a form view can be set as a print view of another form view. In this example the first form view, View 1, has two text box controls, as specified in section 2.4.1.20. The second form view, Print Version View 1, has only one of the text box controls that View 2 has, and is set as a print view of View 1.

The following XSLT excerpt shows two textbox controls for View 1 from view1.xsl.

```
…
<div>
  <span class="xdTextBox" hideFocus="1" title="" xd:binding="my:field1" tabIndex="0"
xd:xctname="PlainText" xd:CtrlId="CTRL1" style="WIDTH: 130px">
    <xsl:value-of select="my:field1"/>
  </span>
  <span class="xdTextBox" hideFocus="1" title="" xd:binding="my:field2" tabIndex="0"
xd:xctname="PlainText" xd:CtrlId="CTRL2" style="WIDTH: 130px">
    <xsl:value-of select="my:field2"/>
  </span>
</div>
…
```

The following XSL excerpt shows only one control for Print Version View 1 from PrintVersionView1.xsl.

```
…
<div>
  <span class="xdTextBox" hideFocus="1" title="" xd:CtrlId="CTRL2" xd:xctname="PlainText"
tabIndex="0" xd:binding="my:field2" style="WIDTH: 130px">
    <xsl:value-of select="my:field2"/>
  </span>
</div>
…
```

The following excerpt shows how the Print Version View 1 form view is set as a print view of View 1 in the manifest from manifest.xsf.

```
…
<xsf:views default="View 1">
  <xsf:view showMenuItem="yes" name="View 1" caption="View 1" printView="Print Version View
1">
    <xsf:mainpane transform="view1.xsl">
    </xsf:mainpane>
  </xsf:view>
  <xsf:view showMenuItem="yes" name="Print Version View 1" caption="Print Version View 1">
    <xsf:mainpane transform="PrintVersionView1.xsl">
    </xsf:mainpane>
  </xsf:view>
</xsf:views>
…
```

The **printView** attribute of the **view** element, as specified in section 2.2.1.2.104, 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 **GetRandom** method. The following fragment from the WSDL file contains the XML schema for the **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://webserviceserver/Everett"
   targetNamespace="http://webserviceserver/Everett"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
<wsdl:import namespace="http://tempuri.org/twrSchema.xsd"
  location="http://webserviceserver/anon/Service1.asmx?schema=typedDataSet" />
  <wsdl:types>
    <s:schema elementFormDefault="qualified"
       targetNamespace="http://webserviceserver/Everett">
    <s:import namespace="http://tempuri.org/twrSchema.xsd" />
    <s:import namespace="http://www.w3.org/2001/XMLSchema" />
    <s:import namespace="http://microsoft.com/wsdl/types/" />
      <s:element name="GetRandom">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="seed" type="s:int" />
            <s:element minOccurs="1" maxOccurs="1" name="min" type="s:int" />
            <s:element minOccurs="1" maxOccurs="1" name="max" type="s:int" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="GetRandomResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="GetRandomResult"
               type="s:int" />
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:schema>
  </wsdl:types>
</wsdl:definitions>
```

The following example shows how a Submit.xml would look when using the **GetRandom** Web service method.

```
<dfs:myFields
   xmlns:dfs="http://schemas.microsoft.com/office/infopath/2003/dataFormSolution"
   xmlns:tns="http://webserviceserver/Everett">
    <dfs:dataFields>
      <tns:GetRandom>
      <tns:seed></tns:seed>
      <tns:min></tns:min>
```

```
      <tns:max></tns:max>
    </tns:GetRandom>
  </dfs:dataFields>
</dfs:myFields>
```

As specified before, Submit.xml can broadly be categorized into two parts. The first part is static and contains <**dfs:myFields**> and <**dfs:dataFields**>. The second part is the template for the **GetRandom** Web service method based on the Web service's WSDL.

The child of <**dfs:dataFields**> is <**tns:GetRandom**>, which is the template of the **GetRandom** Web service method. The three parameters to this Web service method are <*tns:seed*>, <*tns:min*> and <*tns:max*>. This sub-tree validates against the XML schema element <**s:element** name="GetRandom"> in the WSDL example.

3.7 Template.XML Examples

The following example shows two textbox controls with default values. When the form template containing this template.xml file is opened for editing, the first textbox, which is bound to "my:field1" is populated with the initial value "Jean Philippe", and the second text box that is bound to"my:field2" is populated with the initial value "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 in this section. One is upgrade.xsl and one is focused on the **MSXSL:node-set()** function.

3.8.1 Upgrade.XSL Example

This example is a simple upgrade.xsl. The following form file example is 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. The following example is the upgrade.xsl in the form template (.xsn) file.

```
<?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>
```

The following example is the forma file after upgrade.xsl is applied.

```
<?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>
```

Note the following effects:

1. my:field2 has been removed entirely. That data has been discarded.
2. my:myRepeatingGroup has a new field (3), my:fieldText, which is empty for all XML nodes under my:myRepeatingGroup.
3. The data for all other fields (3) 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 an XML node tree in the style sheet. The **for-each** statement combined with the **node-set** function allows the user to iterate over this XML node tree as an XML node set.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:msxsl="urn:schemas-microsoft-com:xslt"
                xmlns:user="http://www.contoso.com"
                version="1.0">
  <xsl:variable name="books">
     <book author="Michael Howard">Writing Secure Code</book>
     <book author="Michael Kay">XSLT Reference</book>
```

```
    </xsl:variable>

    <xsl:template match="/">
       <authors>
          <xsl:for-each select="msxsl:node-set($books)/book">
             <author><xsl:value-of select="@author"/)</author>
          </xsl:for-each>
       </authors>
    </xsl:template>
  </xsl:stylesheet>
```

The following example is the transformation output.

```
<?xml version="1.0" encoding="utf-8"?>

<authors><author>Michael Howard</author><author>Michael Kay</author></authors>
```

4 Security Considerations

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)—that affect cabinet (.cab) files also affect form template (.xsn) files.

Because template.xml is a form file, as specified in [MS-IPFFX], any and all security considerations—including, but not limited to, code signatures—that affect form files also 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**, **XSF2**, and **XSF3 namespaces**, as specified in section 2.2. It also provides the full schemas used by the **built-in ActiveX controls** contact selector, as specified in sections 2.3.1.3, and external item picker, as specified in section 2.3.2.4.

5.1 The InfoPath XSF XSD file

The following XML schema defines the types and elements used in the form definition (.xsf) file. The types and elements belong to the **XSF namespace** (http://schemas.microsoft.com/office/infopath/2003/solutionDefinition), as specified in section 2.2.1.

The XML schema is extended by the additional types and elements in the XSF2 and XSF3 extensions to the form definition (.xsf) file, as specified in sections 5.2 and 5.3.

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}/\\#&amp;&quot;&gt;&lt;])(([^\p{Zl}\p{Zp}\p{C}/\\#&amp;&quot;&gt;&lt;])*(
[^\p{Z}\p{C}/\\#&amp;&quot;&gt;&lt;]))?" />
    </xsd:restriction>
  </xsd:simpleType>
  <!-- xdRoleName type -->
  <!-- uses xdViewName as base -->
  <xsd:simpleType name="xdRoleName">
    <xsd:restriction base="xsf:xdViewName"></xsd:restriction>
  </xsd:simpleType>
  <!-- xdYesNo type -->
  <xsd:simpleType name="xdYesNo">
    <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="yes" />
      <xsd:enumeration value="no" />
```

```
    </xsd:restriction>
  </xsd:simpleType>
  <!-- xdEnabledDisabled type -->
  <xsd:simpleType name="xdEnabledDisabled">
    <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="enabled" />
      <xsd:enumeration value="disabled" />
    </xsd:restriction>
  </xsd:simpleType>
  <!-- xdManualAuto type -->
  <xsd:simpleType name="xdManualAuto">
    <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="manual" />
      <xsd:enumeration value="automatic" />
    </xsd:restriction>
  </xsd:simpleType>
  <!-- xdExpressionLiteral type -->
  <xsd:simpleType name="xdExpressionLiteral">
    <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="expression" />
      <xsd:enumeration value="literal" />
    </xsd:restriction>
  </xsd:simpleType>
  <!-- xdFileName type -->
  <xsd:simpleType name="xdFileName">
    <xsd:restriction base="xsd:string">
      <xsd:minLength value="1" />
      <xsd:maxLength value="64" />
    </xsd:restriction>
  </xsd:simpleType>
  <!-- xdScriptLanguage type -->
  <xsd:simpleType name="xdScriptLanguage">
    <xsd:restriction base="xsd:NMTOKEN">
      <xsd:pattern
value="((([Jj][Aa][Vv][Aa]|(([Jj])|([Vv][Bb])))([Ss][Cc][Rr][Ii][Pp][Tt]))([.][Ee][Nn][Cc][Oo
][Dd][Ee]))|(([Jj][Aa][Vv][Aa]|(([Jj])|([Vv][Bb])))([Ss][Cc][Rr][Ii][Pp][Tt]))|([Mm][Aa][Nn][
Aa][Gg][Ee][Dd][Cc][Oo][Dd][Ee])" />
    </xsd:restriction>
  </xsd:simpleType>
  <!-- xdSolutionVersion type -->
  <xsd:simpleType name="xdSolutionVersion">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="(([0-9]{1,4}.){3}[0-9]{1,4})" />
    </xsd:restriction>
  </xsd:simpleType>
  <!-- xdEmptyString type -->
  <xsd:simpleType name="xdEmptyString">
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="0" />
    </xsd:restriction>
  </xsd:simpleType>
  <!-- xdErrorMessage type -->
  <xsd:simpleType name="xdErrorMessage">
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="1023" />
    </xsd:restriction>
  </xsd:simpleType>
```

```
<!-- xdDesignMode type -->
<xsd:simpleType name="xdDesignMode">
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:enumeration value="normal" />
    <xsd:enumeration value="protected" />
  </xsd:restriction>
</xsd:simpleType>
<!-- xdTrustLevel type -->
<xsd:simpleType name="xdTrustLevel">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="restricted" />
    <xsd:enumeration value="domain" />
  </xsd:restriction>
</xsd:simpleType>
<!-- xdSignedDataBlockName type -->
<xsd:simpleType name="xdSignedDataBlockName">
  <xsd:restriction base="xsd:ID">
    <xsd:minLength value="1" />
    <xsd:maxLength value="255" />
  </xsd:restriction>
</xsd:simpleType>
<!-- xdSignedDataBlockMessage type -->
<xsd:simpleType name="xdSignedDataBlockMessage">
  <xsd:restriction base="xsd:string">
    <xsd:maxLength value="255" />
  </xsd:restriction>
</xsd:simpleType>
<!-- xdSignatureRelationEnum type -->
<xsd:simpleType name="xdSignatureRelationEnum">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="countersign" />
    <xsd:enumeration value="cosign" />
    <xsd:enumeration value="single" />
  </xsd:restriction>
</xsd:simpleType>
<!-- xdSignSignatureLineRuleEnum type -->
<xsd:simpleType name="xdSignSignatureLineRuleEnum">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="suggestedSignerName" />
    <xsd:enumeration value="suggestedSignerEmail" />
    <xsd:enumeration value="signatureLineId" />
  </xsd:restriction>
</xsd:simpleType>
  <!-- xdHWSname type -->
<xsd:simpleType name="xdHWSname">
  <xsd:restriction base="xsd:NCName">
    <xsd:pattern value="[^-^\.^\\^\[^\]^\|^\+^?^\*^@^\{^\}^\(^\)^&gt;^&lt;^=^;^,]*" />
  </xsd:restriction>
</xsd:simpleType>
<!-- xdHWSCaption type -->
<xsd:simpleType name="xdHWSCaption">
  <xsd:restriction base="xsd:string">
    <xsd:minLength value="1" />
    <xsd:maxLength value="255" />
  </xsd:restriction>
</xsd:simpleType>
<!-- xDocumentClass -->
```

```
    <xsd:element name="xDocumentClass">
      <xsd:complexType>
        <xsd:all>
          <xsd:element ref="xsf:package" minOccurs="1" />
          <xsd:element ref="xsf:permissions" minOccurs="0" />
          <xsd:element ref="xsf:views" minOccurs="1" />
          <xsd:element ref="xsf:hwsWorkflow" minOccurs="0" />
          <xsd:element ref="xsf:externalViews" minOccurs="0" />
          <xsd:element ref="xsf:scripts" minOccurs="0" />
          <xsd:element ref="xsf:schemaErrorMessages" minOccurs="0" />
          <xsd:element ref="xsf:documentSchemas" minOccurs="0" />
          <xsd:element ref="xsf:applicationParameters" minOccurs="0" />
          <xsd:element ref="xsf:featureRestrictions" minOccurs="0" />
          <xsd:element ref="xsf:fileNew" minOccurs="0" />
          <xsd:element ref="xsf:customValidation" minOccurs="0" />
          <xsd:element ref="xsf:domEventHandlers" minOccurs="0" />
          <xsd:element ref="xsf:importParameters" minOccurs="0" />
          <xsd:element ref="xsf:listProperties" minOccurs="0" />
          <xsd:element ref="xsf:taskpane" minOccurs="0" />
          <xsd:element ref="xsf:documentSignatures" minOccurs="0" />
          <xsd:element ref="xsf:dataObjects" minOccurs="0" />
          <xsd:element ref="xsf:dataAdapters" minOccurs="0" />
          <xsd:element ref="xsf:query" minOccurs="0" />
          <xsd:element ref="xsf:submit" minOccurs="0" />
          <xsd:element ref="xsf:save" minOccurs="0" />
          <xsd:element ref="xsf:roles" minOccurs="0" />
          <xsd:element ref="xsf:onLoad" minOccurs="0" />
          <xsd:element ref="xsf:documentVersionUpgrade" minOccurs="0" />
          <xsd:element ref="xsf:extensions" minOccurs="0" />
          <xsd:element ref="xsf:ruleSets" minOccurs="0" />
          <xsd:element ref="xsf:calculations" minOccurs="0" />
        </xsd:all>
        <xsd:attribute name="name" type="xsd:string" use="optional" />
        <xsd:attribute name="author" type="xsd:string" use="optional" />
        <xsd:attribute name="description" use="optional">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:maxLength value="255" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="solutionVersion" type="xsf:xdSolutionVersion" use="optional" />
        <xsd:attribute name="productVersion" type="xsd:string" use="optional" />
        <xsd:attribute name="solutionFormatVersion" type="xsf:xdSolutionVersion" use="required"
/>
        <xsd:attribute name="dataFormSolution" type="xsf:xdYesNo" use="optional" />
        <xsd:attribute name="requireFullTrust" type="xsf:xdYesNo" use="optional" />
        <xsd:attribute name="trustLevel" type="xsf:xdTrustLevel" use="optional" />
        <xsd:attribute name="trustSetting" type="xsf:xdManualAuto" use="optional" />
        <xsd:attribute name="publishUrl" type="xsd:string" use="optional" />
      </xsd:complexType>
      <xsd:key name="view_name_key">
        <xsd:selector xpath="./xsf:views/xsf:view" />
        <xsd:field xpath="@name" />
      </xsd:key>
      <xsd:key name="externalView_name_key">
        <xsd:selector xpath="./xsf:externalViews/xsf:externalView" />
```

```
      <xsd:field xpath="@name" />
    </xsd:key>
    <xsd:key name="view_or_externalView_name_key">
      <xsd:selector xpath="./xsf:views/xsf:view | ./xsf:externalViews/xsf:externalView" />
      <xsd:field xpath="@name" />
    </xsd:key>
    <xsd:key name="ruleset_name_key">
      <xsd:selector xpath="./xsf:ruleSets/xsf:ruleSet" />
      <xsd:field xpath="@name" />
    </xsd:key>
    <xsd:key name="dataObject_name_key">
      <xsd:selector xpath="./xsf:dataObjects/xsf:dataObject" />
      <xsd:field xpath="@name" />
    </xsd:key>
    <xsd:unique name="adapter_name_unique">
      <xsd:selector xpath="./xsf:dataObjects/xsf:dataObject/xsf:query/* | ./xsf:query/* |
./xsf:dataAdapters/* | ./xsf:submit/xsf:webServiceAdapter | ./xsf:submit/xsf:davAdapter |
./xsf:submit/xsf:emailAdapter | ./xsf:submit/xsf:submitToHostAdapter" />
      <xsd:field xpath="@name" />
    </xsd:unique>
    <xsd:key name="adapter_name_key">
      <xsd:selector xpath="./xsf:dataAdapters/*" />
      <xsd:field xpath="@name" />
    </xsd:key>
    <xsd:unique name="view_external_name_unique">
      <xsd:selector xpath="./xsf:views/xsf:view | ./xsf:externalViews/xsf:externalView" />
      <xsd:field xpath="@name" />
    </xsd:unique>
  </xsd:element>
  <!-- schemaErrorMessages -->
  <xsd:element name="schemaErrorMessages">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:override" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <!-- override -->
  <xsd:element name="override">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:errorMessage" />
      </xsd:sequence>
      <xsd:attribute name="match" type="xsd:string" use="required" />
    </xsd:complexType>
  </xsd:element>
  <!-- applicationParameters -->
  <xsd:element name="applicationParameters">
    <xsd:complexType>
      <xsd:all>
        <xsd:element ref="xsf:solutionProperties" minOccurs="0" />
      </xsd:all>
      <xsd:attribute name="application" use="required">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="InfoPath Design Mode" />
          </xsd:restriction>
```

```
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:complexType>
  </xsd:element>
  <!-- solutionProperties -->
  <xsd:element name="solutionProperties">
    <xsd:complexType>
      <xsd:attribute name="allowCustomization" type="xsf:xdYesNo" use="optional" />
      <xsd:attribute name="lastOpenView" use="optional" />
      <xsd:attribute name="scriptLanguage" type="xsf:xdScriptLanguage" use="optional" />
      <xsd:attribute name="automaticallyCreateNodes" type="xsf:xdYesNo" use="optional" />
      <xsd:attribute name="lastVersionNeedingTransform" type="xsf:xdSolutionVersion"
use="optional" />
      <xsd:attribute name="fullyEditableNamespace" type="xsd:anyURI" use="optional" />
      <xsd:attribute name="publishSaveUrl" type="xsd:string" use="optional" />
    </xsd:complexType>
  </xsd:element>
  <!-- featureRestrictions -->
  <xsd:element name="featureRestrictions">
    <xsd:complexType>
      <xsd:all>
        <xsd:element name="save" minOccurs="0">
          <xsd:complexType>
            <xsd:attribute name="ui" type="xsf:xdEnabledDisabled" use="required" />
          </xsd:complexType>
        </xsd:element>
        <xsd:element ref="xsf:exportToWeb" minOccurs="0" />
        <xsd:element ref="xsf:exportToExcel" minOccurs="0" />
        <xsd:element ref="xsf:print" minOccurs="0" />
        <xsd:element ref="xsf:sendMail" minOccurs="0" />
        <xsd:element ref="xsf:autoRecovery" minOccurs="0" />
      </xsd:all>
    </xsd:complexType>
  </xsd:element>
  <!-- exportToWeb -->
  <xsd:element name="exportToWeb">
    <xsd:complexType>
      <xsd:attribute name="ui" type="xsf:xdEnabledDisabled" use="required" />
    </xsd:complexType>
  </xsd:element>
  <!-- exportToExcel -->
  <xsd:element name="exportToExcel">
    <xsd:complexType>
      <xsd:attribute name="ui" type="xsf:xdEnabledDisabled" use="required" />
    </xsd:complexType>
  </xsd:element>
  <!-- print -->
  <xsd:element name="print">
    <xsd:complexType>
      <xsd:attribute name="ui" type="xsf:xdEnabledDisabled" use="required" />
    </xsd:complexType>
  </xsd:element>
  <!-- sendMail -->
  <xsd:element name="sendMail">
    <xsd:complexType>
      <xsd:attribute name="ui" type="xsf:xdEnabledDisabled" use="required" />
    </xsd:complexType>
```

```
    </xsd:element>
    <!-- autoRecovery -->
    <xsd:element name="autoRecovery">
      <xsd:complexType>
        <xsd:attribute name="feature" type="xsf:xdEnabledDisabled" use="required" />
      </xsd:complexType>
    </xsd:element>
    <!-- query -->
    <xsd:element name="query">
      <xsd:complexType>
        <xsd:choice>
          <xsd:element ref="xsf:queryAction" />
          <xsd:element ref="xsf:adoAdapter" />
          <xsd:element ref="xsf:webServiceAdapter" />
          <xsd:element ref="xsf:xmlFileAdapter" />
          <xsd:element ref="xsf:sharepointListAdapter" />
          <xsd:element ref="xsf:sharepointListAdapterRW" />
          <xsd:element ref="xsf:bdcAdapter" />
        </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:element ref="xsf:sharepointListAdapterRW" />
          </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="bdcAdapter">
    <xsd:complexType>
      <xsd:attribute name="lobSystemInstance" type="xsd:string" use="optional" />
      <xsd:attribute name="entityNamespace" type="xsd:string" use="required" />
      <xsd:attribute name="entityName" type="xsd:string" use="required" />
      <xsd:attribute name="specificFinder" type="xsd:string" use="required" />
      <xsd:attribute name="name" type="xsf:xdTitle" use="required" />
```

```
      <xsd:attribute name="submitAdapterName" type="xsf:xdTitle" use="required" />
      <xsd:attribute name="queryAllowed" type="xsf:xdYesNo" use="optional" />
      <xsd:attribute name="submitAllowed" type="xsf:xdYesNo" use="optional" />
      <xsd:attribute name="entitySchemaVersion" type="xsd:string" use="required" />
    </xsd:complexType>
  </xsd:element>
    <xsd:element name="grooveAdapter">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="field" minOccurs="0" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:attribute name="name" type="xsd:string" use="required" />
            <xsd:attribute name="displayName" type="xsd:string" use="optional" />
            <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="spaceName" type="xsd:string" use="required" />
      <xsd:attribute name="spaceBindableUrl" type="xsd:string" use="required" />
      <xsd:attribute name="spaceCanonicalUrl" type="xsd:string" use="required" />
      <xsd:attribute name="spaceCanonicalUrlFormattedForSandboxing" type="xsd:string"
use="required" />
      <xsd:attribute name="toolName" type="xsd:string" use="required" />
      <xsd:attribute name="toolDisplayName" type="xsd:string" use="required" />
      <xsd:attribute name="toolBindableUrl" type="xsd:string" use="required" />
      <xsd:attribute name="toolCanonicalUrl" type="xsd:string" use="required" />
      <xsd:attribute name="viewName" type="xsd:string" use="required" />
      <xsd:attribute name="viewDisplayName" type="xsd:string" use="required" />
      <xsd:attribute name="infopathGroup" type="xsd:string" use="required" />
      <xsd:attribute name="queryAllowed" type="xsf:xdYesNo" use="optional" default="yes" />
      <xsd:attribute name="submitAllowed" type="xsf:xdYesNo" use="optional" default="no" />
      <xsd:attribute name="queryThisFormOnly" type="xsf:xdYesNo" use="optional" default="no"
/>
    </xsd:complexType>
    </xsd:element>
  <xsd:element name="davAdapter">
    <xsd:complexType>
      <xsd:all>
        <xsd:element name="folderURL">
          <xsd:complexType>
            <xsd:attribute name="value" type="xsd:string" use="required" />
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="fileName">
          <xsd:complexType>
            <xsd:attribute name="value" type="xsd:string" use="required" />
            <xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="optional" />
          </xsd:complexType>
        </xsd:element>
      </xsd:all>
      <xsd:attribute name="name" type="xsf:xdTitle" use="required" />
      <xsd:attribute name="overwriteAllowed" type="xsf:xdYesNo" use="optional" />
      <xsd:attribute name="queryAllowed" type="xsf:xdYesNo" use="optional" />
      <xsd:attribute name="submitAllowed" type="xsf:xdYesNo" use="optional" />
    </xsd:complexType>
```

```
</xsd:element>
<xsd:element name="emailAdapter">
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="to" minOccurs="0">
        <xsd:complexType>
          <xsd:attribute name="value" type="xsd:string" use="required" />
          <xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="optional" />
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="cc" minOccurs="0">
        <xsd:complexType>
          <xsd:attribute name="value" type="xsd:string" use="required" />
          <xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="optional" />
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="bcc" minOccurs="0">
        <xsd:complexType>
          <xsd:attribute name="value" type="xsd:string" use="required" />
          <xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="optional" />
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="subject" minOccurs="0">
        <xsd:complexType>
          <xsd:attribute name="value" type="xsd:string" use="required" />
          <xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="optional" />
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="intro" minOccurs="0">
        <xsd:complexType>
          <xsd:attribute name="value" type="xsd:string" use="required" />
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="attachmentFileName" minOccurs="0">
        <xsd:complexType>
          <xsd:attribute name="value" type="xsd:string" use="required" />
          <xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="optional" />
        </xsd:complexType>
      </xsd:element>
    </xsd:all>
    <xsd:attribute name="name" type="xsf:xdTitle" use="required" />
    <xsd:attribute name="queryAllowed" type="xsf:xdYesNo" use="optional" />
    <xsd:attribute name="submitAllowed" type="xsf:xdYesNo" use="optional" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="submitToHostAdapter">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsf:xdTitle" use="required" />
    <xsd:attribute name="queryAllowed" type="xsf:xdYesNo" use="optional" />
    <xsd:attribute name="submitAllowed" type="xsf:xdYesNo" use="optional" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="sharepointListAdapterRW">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="field" minOccurs="0" maxOccurs="unbounded">
        <xsd:complexType>
```

```
            <xsd:attribute name="internalName" type="xsd:string" use="required" />
            <xsd:attribute name="hiddenFieldName" type="xsd:string" use="optional" />
            <xsd:attribute name="type" type="xsd:string" use="required" />
            <xsd:attribute name="auxDomName" type="xsd:string" use="optional" />
            <xsd:attribute name="showFieldName" type="xsd:string" use="optional" />
            <xsd:attribute name="required" type="xsf:xdYesNo" use="optional" />
            <xsd:attribute name="appendOnly" 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="sharePointListID" type="xsd:string" use="required" />
      <xsd:attribute name="contentTypeID" 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="autogen" type="xsf:xdYesNo" use="optional" />
      <xsd:attribute name="relativeListUrl" type="xsd:string" use="optional" />
      <xsd:attribute name="version" type="xsd:string" use="optional" />
      <xsd:attribute name="sharePointListChoices" type="xsd:string" use="optional" />
      <xsd:attribute name="queryOneItemOnly" type="xsf:xdYesNo" use="optional" />
      <xsd:attribute name="sortBy" type="xsd:string" use="optional" />
      <xsd:attribute name="sortAscending" type="xsf:xdYesNo" use="optional" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="dataAdapters">
    <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element ref="xsf:adoAdapter" />
        <xsd:element ref="xsf:webServiceAdapter" />
        <xsd:element ref="xsf:xmlFileAdapter" />
        <xsd:element ref="xsf:sharepointListAdapter" />
        <xsd:element ref="xsf:davAdapter" />
        <xsd:element ref="xsf:emailAdapter" />
        <xsd:element ref="xsf:submitToHostAdapter" />
        <xsd:element ref="xsf:hwsAdapter" />
      </xsd:choice>
    </xsd:complexType>
  </xsd:element>
  <!-- documentSchemas -->
  <xsd:element name="documentSchemas">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:documentSchema" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="documentSchema">
    <xsd:complexType>
      <xsd:attribute name="location" type="xsd:string" use="required" />
      <xsd:attribute name="rootSchema" type="xsf:xdYesNo" />
    </xsd:complexType>
  </xsd:element>
  <!-- customValidation -->
  <xsd:element name="customValidation">
    <xsd:complexType>
      <xsd:sequence>
```

```
      <xsd:element ref="xsf:errorCondition" minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="errorCondition">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:errorMessage" />
    </xsd:sequence>
    <xsd:attribute name="match" type="xsd:string" use="required" />
    <xsd:attribute name="expression" type="xsd:string" use="required" />
    <xsd:attribute name="expressionContext" type="xsd:string" use="optional" />
    <xsd:attribute name="showErrorOn" type="xsd:string" use="optional" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="errorMessage">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsf:xdErrorMessage">
        <xsd:attribute name="type" use="optional">
          <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
              <xsd:enumeration value="modal" />
              <xsd:enumeration value="modeless" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="shortMessage" use="required">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:maxLength value="127" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
<!-- domEventHandlers -->
<xsd:element name="domEventHandlers">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:domEventHandler" minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:unique name="domEventHandler_handlerObject_unique">
    <xsd:selector xpath="." />
    <xsd:field xpath="@handlerObject" />
  </xsd:unique>
</xsd:element>
<xsd:element name="domEventHandler">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:ruleSetAction" minOccurs="0" maxOccurs="1" />
    </xsd:sequence>
    <xsd:attribute name="dataObject" type="xsd:string" use="optional" />
    <xsd:attribute name="match" type="xsd:string" use="required" />
```

```
      <xsd:attribute name="handlerObject" type="xsd:string" use="optional" />
    </xsd:complexType>
    <xsd:keyref name="domEventHandler_ruleSetAction" refer="xsf:ruleset_name_key">
      <xsd:selector xpath="./xsf:ruleSetAction" />
      <xsd:field xpath="@ruleSet" />
    </xsd:keyref>
  </xsd:element>
  <!-- importParameters -->
  <xsd:element name="importParameters">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:importSource" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
      <xsd:attribute name="enabled" type="xsf:xdYesNo" use="required" />
      <xsd:attribute name="useScriptHandler" type="xsf:xdYesNo" use="optional" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="importSource">
    <xsd:complexType>
      <xsd:attribute name="name" type="xsd:string" use="required" />
      <xsd:attribute name="schema" type="xsf:xdFileName" use="required" />
      <xsd:attribute name="transform" type="xsf:xdFileName" use="required" />
      <xsd:attribute name="authoringOfTransform" type="xsf:xdManualAuto" use="optional" />
    </xsd:complexType>
  </xsd:element>
  <!-- listProperties -->
  <xsd:element name="listProperties">
    <xsd:complexType>
      <xsd:all>
        <xsd:element ref="xsf:fields" />
      </xsd:all>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="fields">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:field" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="field">
    <xsd:complexType>
      <xsd:attribute name="type" type="xsd:NMTOKEN" use="required" />
      <xsd:attribute name="name" type="xsf:xdTitle" use="required" />
      <xsd:attribute name="columnName" type="xsf:xdTitle" use="required" />
      <xsd:attribute name="required" type="xsf:xdYesNo" use="optional" />
      <xsd:attribute name="viewable" type="xsf:xdYesNo" use="optional" />
      <xsd:attribute name="node" type="xsd:string" use="required" />
      <xsd:attribute name="maxLength" type="xsd:byte" />
      <xsd:attribute name="aggregation" use="optional">
        <xsd:simpleType>
          <xsd:restriction base="xsd:NMTOKEN">
            <xsd:enumeration value="sum" />
            <xsd:enumeration value="count" />
            <xsd:enumeration value="average" />
            <xsd:enumeration value="min" />
            <xsd:enumeration value="max" />
```

```
            <xsd:enumeration value="first" />
            <xsd:enumeration value="last" />
            <xsd:enumeration value="merge" />
            <xsd:enumeration value="plaintext" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:complexType>
  </xsd:element>
  <!-- submit -->
  <xsd:element name="submit">
    <xsd:complexType>
      <xsd:all>
        <xsd:element name="submitAction" minOccurs="0">
          <xsd:complexType>
            <xsd:attribute name="adapter" type="xsf:xdTitle" use="required" />
          </xsd:complexType>
          <xsd:keyref name="submitAdapter_name_keyref" refer="xsf:adapter_name_key">
            <xsd:selector xpath="." />
            <xsd:field xpath="@adapter" />
          </xsd:keyref>
        </xsd:element>
        <xsd:element ref="xsf:useHttpHandler" minOccurs="0" />
        <xsd:element ref="xsf:useScriptHandler" minOccurs="0" />
        <xsd:element ref="xsf:ruleSetAction" minOccurs="0" />
        <xsd:element ref="xsf:useQueryAdapter" minOccurs="0" />
        <xsd:element ref="xsf:webServiceAdapter" minOccurs="0" />
        <xsd:element ref="xsf:davAdapter" minOccurs="0" />
        <xsd:element ref="xsf:emailAdapter" minOccurs="0" />
        <xsd:element ref="xsf:submitToHostAdapter" minOccurs="0" />
        <xsd:element name="successMessage" type="xsd:string" minOccurs="0" />
        <xsd:element name="errorMessage" type="xsd:string" minOccurs="0" />
      </xsd:all>
      <xsd:attribute name="caption" type="xsd:string" use="optional" />
      <xsd:attribute name="onAfterSubmit" use="optional">
        <xsd:simpleType>
          <xsd:restriction base="xsd:NMTOKEN">
            <xsd:enumeration value="close" />
            <xsd:enumeration value="keepOpen" />
            <xsd:enumeration value="openNew" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="showStatusDialog" type="xsf:xdYesNo" use="optional" />
      <xsd:attribute name="showSignatureReminder" type="xsf:xdYesNo" use="optional" />
      <xsd:attribute name="disableMenuItem" type="xsf:xdYesNo" use="optional" />
    </xsd:complexType>
    <xsd:keyref name="submit_ruleSetAction" refer="xsf:ruleset_name_key">
      <xsd:selector xpath="./xsf:ruleSetAction" />
      <xsd:field xpath="@ruleSet" />
    </xsd:keyref>
  </xsd:element>
  <xsd:element name="useHttpHandler">
    <xsd:complexType>
      <xsd:attribute name="method" use="required">
        <xsd:simpleType>
          <xsd:restriction base="xsd:NMTOKEN">
```

```
            <xsd:enumeration value="POST" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="href" type="xsd:anyURI" use="required" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="useScriptHandler" />
  <xsd:element name="useQueryAdapter" />
  <!-- onLoad -->
  <xsd:element name="onLoad">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:ruleSetAction" minOccurs="1" maxOccurs="1" />
      </xsd:sequence>
    </xsd:complexType>
    <xsd:keyref name="load_ruleSetAction" refer="xsf:ruleset_name_key">
      <xsd:selector xpath="./xsf:ruleSetAction" />
      <xsd:field xpath="@ruleSet" />
    </xsd:keyref>
  </xsd:element>
  <!-- save -->
  <xsd:element name="save">
    <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="1">
        <xsd:element ref="xsf:useScriptHandler" />
      </xsd:choice>
    </xsd:complexType>
  </xsd:element>
  <!-- roles -->
  <xsd:element name="roles">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:role" minOccurs="1" maxOccurs="unbounded" />
        <xsd:element ref="xsf:membership" minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
      <xsd:attribute name="default" type="xsd:string" use="required" />
      <xsd:attribute name="initiator" type="xsd:string" use="optional" />
      <xsd:attribute name="hideStatusBarDisplay" type="xsf:xdYesNo" use="optional" />
    </xsd:complexType>
    <!-- role names must be unique -->
    <xsd:unique name="roles_name_unique">
      <xsd:selector xpath="./xsf:role" />
      <xsd:field xpath="@name" />
    </xsd:unique>
    <!-- fields must reference existing role -->
    <xsd:key name="role_name_key">
      <xsd:selector xpath="./xsf:role" />
      <xsd:field xpath="@name" />
    </xsd:key>
    <xsd:keyref name="role_default" refer="xsf:role_name_key">
      <xsd:selector xpath="." />
      <xsd:field xpath="@default" />
    </xsd:keyref>
    <xsd:keyref name="role_initiator" refer="xsf:role_name_key">
      <xsd:selector xpath="." />
      <xsd:field xpath="@initiator" />
```

```
    </xsd:keyref>
    <xsd:keyref name="role_membership" refer="xsf:role_name_key">
      <xsd:selector xpath="./xsf:membership/*" />
      <xsd:field xpath="@memberOf" />
    </xsd:keyref>
  </xsd:element>
  <xsd:element name="role">
    <xsd:complexType>
      <xsd:attribute name="name" type="xsf:xdRoleName" use="required" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="membership">
    <xsd:complexType>
      <xsd:choice minOccurs="1" maxOccurs="unbounded">
        <xsd:element ref="xsf:getUserNameFromData" />
        <xsd:element ref="xsf:userName" />
        <xsd:element ref="xsf:group" />
      </xsd:choice>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="getUserNameFromData">
    <xsd:complexType>
      <xsd:attribute name="dataObject" type="xsd:string" use="optional" />
      <xsd:attribute name="select" type="xsd:string" use="required" />
      <xsd:attribute name="memberOf" type="xsd:string" use="required" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="userName">
    <xsd:complexType>
      <xsd:attribute name="name" type="xsd:string" use="required" />
      <xsd:attribute name="memberOf" type="xsd:string" use="required" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="group">
    <xsd:complexType>
      <xsd:attribute name="name" type="xsd:string" use="required" />
      <xsd:attribute name="memberOf" type="xsd:string" use="required" />
    </xsd:complexType>
  </xsd:element>
  <!-- hwsWorkflow -->
  <xsd:element name="hwsWorkflow">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:location" minOccurs="1" maxOccurs="1" />
        <xsd:element ref="xsf:allowedActions" minOccurs="1" maxOccurs="1" />
        <xsd:element ref="xsf:allowedTasks" minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
      <xsd:attribute name="taskpaneVisible" type="xsf:xdYesNo" />
    </xsd:complexType>
    <xsd:unique name="hws_actiontask_name">
      <xsd:selector xpath="./xsf:allowedActions/xsf:action|./xsf:allowedTasks/xsf:task" />
      <xsd:field xpath="@name" />
    </xsd:unique>
  </xsd:element>
  <!-- location -->
  <xsd:element name="location">
    <xsd:complexType>
```

```
      <xsd:attribute name="url" type="xsd:string" use="required" />
    </xsd:complexType>
  </xsd:element>
  <!-- allowedActions -->
  <xsd:element name="allowedActions">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:action" minOccurs="1" maxOccurs="20" />
      </xsd:sequence>
    </xsd:complexType>
    <xsd:unique name="hws_actionTypeID_unique">
      <xsd:selector xpath="./xsf:action" />
      <xsd:field xpath="@actionTypeID" />
    </xsd:unique>
  </xsd:element>
  <!-- action -->
  <xsd:element name="action">
    <xsd:complexType>
      <xsd:attribute name="name" type="xsf:xdHWSname" use="required" />
      <xsd:attribute name="actionTypeID" type="xsd:string" use="required" />
      <xsd:attribute name="canInitiateWorkflow" type="xsf:xdYesNo" use="required" />
      <xsd:attribute name="caption" type="xsf:xdHWSCaption" use="optional" />
    </xsd:complexType>
  </xsd:element>
  <!-- allowedTasks -->
  <xsd:element name="allowedTasks">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:task" minOccurs="1" maxOccurs="20" />
      </xsd:sequence>
    </xsd:complexType>
    <xsd:unique name="hws_taskID_unique">
      <xsd:selector xpath="./xsf:task" />
      <xsd:field xpath="@taskTypeID" />
    </xsd:unique>
  </xsd:element>
  <!-- task -->
  <xsd:element name="task">
    <xsd:complexType>
      <xsd:attribute name="name" type="xsf:xdHWSname" use="required" />
      <xsd:attribute name="taskTypeID" type="xsd:string" use="required" />
      <xsd:attribute name="caption" type="xsf:xdHWSCaption" use="optional" />
    </xsd:complexType>
  </xsd:element>
  <!-- fileNew -->
  <xsd:element name="fileNew">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:initialXmlDocument" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="initialXmlDocument">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:customCategory" minOccurs="0" />
      </xsd:sequence>
```

```
      <xsd:attribute name="caption" type="xsf:xdTitle" use="required" />
      <xsd:attribute name="href" type="xsf:xdFileName" use="required" />
    </xsd:complexType>
  </xsd:element>
  <!-- customCategory -->
  <xsd:element name="customCategory">
    <xsd:complexType>
      <xsd:attribute name="name" type="xsf:xdTitle" use="required" />
    </xsd:complexType>
  </xsd:element>
  <!-- package -->
  <xsd:element name="package">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:files" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="files">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:file" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="file">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:fileProperties" minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
      <xsd:attribute name="name" type="xsf:xdFileName" use="required" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="fileProperties">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:property" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="property">
    <xsd:complexType>
      <xsd:attribute name="name" type="xsd:string" use="required" />
      <xsd:attribute name="value" type="xsd:string" use="required" />
      <xsd:attribute name="type" type="xsd:QName" use="required" />
    </xsd:complexType>
  </xsd:element>
  <!-- permissions -->
  <xsd:element name="permissions">
    <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element ref="xsf:allowedControl" />
      </xsd:choice>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="allowedControl">
    <xsd:complexType>
```

```
      <xsd:attribute name="cabFile" type="xsd:string" use="optional" />
      <xsd:attribute name="clsid" type="xsd:string" use="required" />
      <xsd:attribute name="version" type="xsd:string" use="optional" />
    </xsd:complexType>
  </xsd:element>
  <!-- View and Context-Driven Editing definitions -->
  <!-- External Views -->
  <xsd:element name="externalViews">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:externalView" minOccurs="1" maxOccurs="unbounded" />
      </xsd:sequence>
      <xsd:attribute name="default" type="xsd:string" />
    </xsd:complexType>
    <xsd:unique name="externalViews_name_unique">
      <xsd:selector xpath="./xsf:externalView" />
      <xsd:field xpath="@default" />
    </xsd:unique>
    <xsd:keyref name="external_views_printView" refer="xsf:externalView_name_key">
      <xsd:selector xpath="." />
      <xsd:field xpath="@default" />
    </xsd:keyref>
  </xsd:element>
  <xsd:element name="externalView">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:mainpane" />
      </xsd:sequence>
      <xsd:attribute name="target" type="xsd:string" />
      <xsd:attribute name="name" type="xsf:xdViewName" use="required" />
      <xsd:attribute name="designMode" type="xsf:xdDesignMode" />
    </xsd:complexType>
  </xsd:element>
  <!-- attributeData -->
  <xsd:element name="attributeData">
    <xsd:complexType>
      <xsd:attribute name="attribute" type="xsd:string" use="required" />
      <xsd:attribute name="value" type="xsd:string" use="required" />
    </xsd:complexType>
  </xsd:element>
  <!-- button -->
  <xsd:element name="button">
    <xsd:complexType>
      <xsd:attribute name="caption" type="xsf:xdTitle" />
      <xsd:attribute name="icon" type="xsd:string" />
      <xsd:attribute name="tooltip" type="xsf:xdTitle" />
      <xsd:attribute name="name" type="xsd:NMTOKEN" />
      <xsd:attribute name="xmlToEdit" type="xsd:NMTOKEN" />
      <xsd:attribute name="action">
        <xsd:simpleType>
          <xsd:restriction base="xsd:NMTOKEN">
            <xsd:enumeration value="xCollection::insert" />
            <xsd:enumeration value="xCollection::insertBefore" />
            <xsd:enumeration value="xCollection::insertAfter" />
            <xsd:enumeration value="xCollection::remove" />
            <xsd:enumeration value="xCollection::refreshFilter" />
            <xsd:enumeration value="xCollection::removeAll" />
```

```
            <xsd:enumeration value="xOptional::insert" />
            <xsd:enumeration value="xOptional::remove" />
            <xsd:enumeration value="xReplace::replace" />

            <xsd:enumeration value="xFileAttachment::attach" />
            <xsd:enumeration value="xFileAttachment::open" />
            <xsd:enumeration value="xFileAttachment::saveAs" />
            <xsd:enumeration value="xFileAttachment::remove" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="showIf">
        <xsd:simpleType>
          <xsd:restriction base="xsd:NMTOKEN">
            <xsd:enumeration value="always" />
            <xsd:enumeration value="enabled" />
            <xsd:enumeration value="immediate" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:complexType>
  </xsd:element>
  <!-- chooseFragment -->
  <xsd:element name="chooseFragment">
    <xsd:complexType mixed="true">
      <xsd:sequence>
        <xsd:any minOccurs="0" maxOccurs="unbounded" processContents="skip" />
      </xsd:sequence>
      <xsd:attribute name="parent" type="xsd:string" />
      <xsd:attribute name="followingSiblings" type="xsd:string" use="optional" />
      <xsd:attribute name="innerFragment" type="xsd:string" use="optional" />
    </xsd:complexType>
  </xsd:element>
  <!-- editWith -->
  <xsd:element name="editWith">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:masterDetail" minOccurs="0" maxOccurs="1" />
        <xsd:element ref="xsf:fragmentToInsert" minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
      <xsd:attribute name="component" use="required">
        <xsd:simpleType>
          <xsd:restriction base="xsd:NMTOKEN">
            <xsd:enumeration value="xCollection" />
            <xsd:enumeration value="xOptional" />
            <xsd:enumeration value="xReplace" />
            <xsd:enumeration value="xTextList" />
            <xsd:enumeration value="xField" />
            <xsd:enumeration value="xImage" />
            <xsd:enumeration value="xFileAttachment" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="caption" type="xsf:xdTitle" use="optional" />
      <xsd:attribute name="autoComplete" type="xsf:xdYesNo" use="optional" />
      <xsd:attribute name="proofing" type="xsf:xdYesNo" use="optional" />
      <xsd:attribute name="type" use="optional">
```

```
          <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
              <xsd:enumeration value="plain" />
              <xsd:enumeration value="formatted" />
              <xsd:enumeration value="plainMultiline" />
              <xsd:enumeration value="formattedMultiline" />
              <xsd:enumeration value="rich" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="useFilter" use="optional">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="yes" />
              <xsd:enumeration value="no" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="widgetIcon" use="optional">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="standard" />
              <xsd:enumeration value="filter" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="filterDependency" type="xsd:string" use="optional" />
        <xsd:attribute name="field" type="xsd:string" use="optional" />
        <xsd:attribute name="removeAncestors" type="xsd:nonNegativeInteger" use="optional" />
        <xsd:attribute name="maxLength" use="optional">
          <xsd:simpleType>
            <xsd:restriction base="xsd:integer">
              <xsd:minInclusive value="-1" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="allowedFileTypes" type="xsd:string" use="optional" />
        <xsd:anyAttribute namespace="http://schemas.microsoft.com/office/infopath/2003"
  processContents="skip" />
      </xsd:complexType>
    </xsd:element>
    <!-- unboundControls -->
    <xsd:element name="unboundControls">
      <xsd:complexType>
        <xsd:sequence>
          <!-- button -->
          <xsd:element name="button" minOccurs="0" maxOccurs="unbounded">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element ref="xsf:ruleSetAction" minOccurs="0" maxOccurs="1" />
              </xsd:sequence>
              <xsd:attribute name="name" use="required">
                <xsd:simpleType>
                  <!-- type of name is non qualified name, but NCName also accepts '.' and '-',
                  so these characters are  disabled by pattern restriction -->
                  <xsd:restriction base="xsd:NCName">
                    <xsd:pattern value="[^\.\^-]*" />
```

```
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:attribute>
        </xsd:complexType>
        <xsd:keyref name="button_ruleSetAction" refer="xsf:ruleset_name_key">
          <xsd:selector xpath="./xsf:ruleSetAction" />
          <xsd:field xpath="@ruleSet" />
        </xsd:keyref>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<!-- editing -->
<xsd:element name="editing">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:xmlToEdit" minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<!-- Master Detail -->
<xsd:element name="masterDetail">
  <xsd:complexType>
    <xsd:attribute name="master" type="xsd:string" />
    <xsd:attribute name="masterViewContext" type="xsd:string" />
    <xsd:attribute name="masterKey" type="xsd:string" />
    <xsd:attribute name="detailKey" type="xsd:string" />
  </xsd:complexType>
</xsd:element>
<!-- fragmentToInsert -->
<xsd:element name="fragmentToInsert">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:chooseFragment" minOccurs="1" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<!-- mainpane -->
<xsd:element name="mainpane">
  <xsd:complexType>
    <xsd:attribute name="transform" type="xsf:xdFileName" use="required" />
  </xsd:complexType>
</xsd:element>
<!-- printSettings -->
<xsd:element name="printSettings">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:header" minOccurs="0" maxOccurs="1" />
      <xsd:element ref="xsf:footer" minOccurs="0" maxOccurs="1" />
    </xsd:sequence>
    <xsd:attribute name="orientation">
      <xsd:simpleType>
        <xsd:restriction base="xsd:NMTOKEN">
          <xsd:enumeration value="portrait" />
          <xsd:enumeration value="landscape" />
        </xsd:restriction>
      </xsd:simpleType>
```

```
</xsd:attribute>
<xsd:attribute name="header">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="255" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="footer">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="255" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="marginUnitsType">
  <xsd:simpleType>
    <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="in" />
      <xsd:enumeration value="cm" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="rightMargin">
  <xsd:simpleType>
    <xsd:restriction base="xsd:float">
      <xsd:minInclusive value="0" />
      <xsd:maxInclusive value="100" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="leftMargin">
  <xsd:simpleType>
    <xsd:restriction base="xsd:float">
      <xsd:minInclusive value="0" />
      <xsd:maxInclusive value="100" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="topMargin">
  <xsd:simpleType>
    <xsd:restriction base="xsd:float">
      <xsd:minInclusive value="0" />
      <xsd:maxInclusive value="100" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="bottomMargin">
  <xsd:simpleType>
    <xsd:restriction base="xsd:float">
      <xsd:minInclusive value="0" />
      <xsd:maxInclusive value="100" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="printerName">
  <xsd:simpleType>
```

```
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="255" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="paperSize">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="255" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="paperSource">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="255" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="copies">
      <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
          <xsd:minInclusive value="1" />
          <xsd:maxInclusive value="9999" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="collate" type="xsf:xdYesNo" />
    <xsd:attribute name="pageRangeStart">
      <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
          <xsd:minInclusive value="1" />
          <xsd:maxInclusive value="32000" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="pageRangeEnd">
      <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
          <xsd:minInclusive value="1" />
          <xsd:maxInclusive value="32000" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="printerSpecificSettings">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="255" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
</xsd:element>
<xsd:element name="header">
  <xsd:complexType mixed="true">
    <xsd:sequence>
      <xsd:any minOccurs="0" maxOccurs="unbounded" processContents="skip" />
```

```
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="footer">
  <xsd:complexType mixed="true">
    <xsd:sequence>
      <xsd:any minOccurs="0" maxOccurs="unbounded" processContents="skip" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<!-- toolbar -->
<xsd:element name="toolbar">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:group ref="xsf:UIItem" minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:attribute name="name" type="xsf:xdTitle" use="required" />
    <xsd:attribute name="caption" type="xsf:xdTitle" use="required" />
  </xsd:complexType>
</xsd:element>
<!-- menu -->
<xsd:element name="menu">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:group ref="xsf:UIItem" minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:attribute name="caption" type="xsf:xdTitle" use="required" />
  </xsd:complexType>
</xsd:element>
<!-- menuArea -->
<xsd:element name="menuArea">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:group ref="xsf:UIItem" minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:attribute name="name" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:NMTOKEN">
          <xsd:enumeration value="msoFileMenu" />
          <xsd:enumeration value="msoEditMenu" />
          <xsd:enumeration value="msoInsertMenu" />
          <xsd:enumeration value="msoViewMenu" />
          <xsd:enumeration value="msoFormatMenu" />
          <xsd:enumeration value="msoToolsMenu" />
          <xsd:enumeration value="msoTableMenu" />
          <xsd:enumeration value="msoHelpMenu" />
          <xsd:enumeration value="msoStructuralEditingContextMenu" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
</xsd:element>
<!-- UIContainer -->
<xsd:group name="UIContainer">
  <xsd:choice>
    <xsd:element ref="xsf:toolbar" />
    <xsd:element ref="xsf:menu" />
```

```
      <xsd:element ref="xsf:menuArea" />
    </xsd:choice>
  </xsd:group>
  <!-- UIItem -->
  <xsd:group name="UIItem">
    <xsd:choice>
      <xsd:element ref="xsf:button" />
      <xsd:element ref="xsf:menu" />
    </xsd:choice>
  </xsd:group>
  <!-- taskpane -->
  <xsd:element name="taskpane">
    <xsd:complexType>
      <xsd:attribute name="caption" type="xsd:string" use="required" />
      <xsd:attribute name="href" type="xsd:string" use="required" />
    </xsd:complexType>
  </xsd:element>
  <!-- views -->
  <xsd:element name="views">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:view" minOccurs="1" maxOccurs="unbounded" />
      </xsd:sequence>
      <xsd:attribute name="default" type="xsd:string" />
    </xsd:complexType>
    <xsd:unique name="views_name_unique">
      <xsd:selector xpath="./xsf:view" />
      <xsd:field xpath="@name" />
    </xsd:unique>
    <xsd:keyref name="view_printView" refer="xsf:view_or_externalView_name_key">
      <xsd:selector xpath="./xsf:view" />
      <xsd:field xpath="@printView" />
    </xsd:keyref>
    <xsd:keyref name="views_default" refer="xsf:view_name_key">
      <xsd:selector xpath="." />
      <xsd:field xpath="@default" />
    </xsd:keyref>
  </xsd:element>
  <!-- ViewContent -->
  <xsd:group name="ViewContent">
    <xsd:choice>
      <xsd:element ref="xsf:editing" minOccurs="0" />
      <xsd:element ref="xsf:mainpane" minOccurs="0" />
      <xsd:element ref="xsf:printSettings" minOccurs="0" />
      <xsd:group ref="xsf:UIContainer" minOccurs="0" maxOccurs="unbounded" />
      <xsd:element ref="xsf:unboundControls" minOccurs="0" />
    </xsd:choice>
  </xsd:group>
  <!-- view -->
  <xsd:element name="view">
    <xsd:complexType>
      <xsd:group ref="xsf:ViewContent" minOccurs="0" maxOccurs="unbounded" />
      <xsd:attribute name="caption" type="xsf:xdViewName" />
      <xsd:attribute name="name" type="xsf:xdViewName" use="required" />
      <xsd:attribute name="showMenuItem" type="xsf:xdYesNo" use="optional" />
      <xsd:attribute name="printView" type="xsd:string" />
      <xsd:attribute name="designMode" type="xsf:xdDesignMode" />
```

```
    </xsd:complexType>
    <xsd:unique name="toolbar_name_unique">
      <xsd:selector xpath="./xsf:toolbar" />
      <xsd:field xpath="@name" />
    </xsd:unique>
    <xsd:unique name="menuArea_name_unique">
      <xsd:selector xpath="./xsf:menuArea" />
      <xsd:field xpath="@name" />
    </xsd:unique>
    <xsd:unique name="xmlToEdit_name_unique">
      <xsd:selector xpath="./xsf:editing/xsf:xmlToEdit" />
      <xsd:field xpath="@name" />
    </xsd:unique>
    <xsd:key name="xmlToEdit_name_key">
      <xsd:selector xpath="./xsf:editing/xsf:xmlToEdit" />
      <xsd:field xpath="@name" />
    </xsd:key>
    <xsd:keyref name="button_xmlToEdit_reference" refer="xsf:xmlToEdit_name_key">
      <xsd:selector xpath="./xsf:menuArea/xsf:button | ./xsf:menu/xsf:button |
./xsf:toolbar/xsf:button" />
      <xsd:field xpath="@xmlToEdit" />
    </xsd:keyref>
  </xsd:element>
  <!-- xmlToEdit -->
  <xsd:element name="xmlToEdit">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:editWith" minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:NMTOKEN" use="required" />
      <xsd:attribute name="item" type="xsd:string" use="required" />
      <xsd:attribute name="container" type="xsd:string" />
      <xsd:attribute name="viewContext">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:pattern value="((\.|\#|[a-zA-Z0-9_])[a-zA-Z0-9_]*)(\s((\.|\#|[a-zA-Z0-
9_])[a-zA-Z0-9_]*))*" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:complexType>
  </xsd:element>
  <!-- Digital Signatures -->
  <xsd:element name="documentSignatures">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:signedDataBlock" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
      <xsd:attribute name="signatureLocation" type="xsd:string" use="optional" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="signedDataBlock">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="message" type="xsf:xdSignedDataBlockMessage" minOccurs="0" />
      </xsd:sequence>
      <xsd:attribute name="name" type="xsf:xdSignedDataBlockName" use="required" />
```

```
      <xsd:attribute name="data" type="xsd:string" use="required" />
      <xsd:attribute name="signatureLocation" type="xsd:string" use="required" />
      <xsd:attribute name="mode" type="xsf:xdSignatureRelationEnum" use="required" />
    </xsd:complexType>
    <xsd:unique name="signedDataBlock_name_unique">
      <xsd:selector xpath="." />
      <xsd:field xpath="@name" />
    </xsd:unique>
  </xsd:element>
  <!-- Version Upgrade -->
  <xsd:element name="documentVersionUpgrade">
    <xsd:complexType>
      <xsd:choice>
        <xsd:element ref="xsf:useScriptHandler" />
        <xsd:element ref="xsf:useTransform" />
      </xsd:choice>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="useTransform">
    <xsd:complexType>
      <xsd:attribute name="transform" use="required">
        <xsd:simpleType>
          <xsd:union memberTypes="xsf:xdFileName xsf:xdEmptyString" />
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="minVersionToUpgrade" type="xsf:xdSolutionVersion" use="required"
/>
      <xsd:attribute name="maxVersionToUpgrade" type="xsf:xdSolutionVersion" />
    </xsd:complexType>
  </xsd:element>
  <!-- XSF Extensions -->
  <xsd:element name="extensions">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:extension" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="extension">
    <xsd:complexType mixed="true">
      <xsd:sequence>
        <xsd:any minOccurs="0" maxOccurs="unbounded" processContents="lax" />
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:NMTOKEN" use="required" />
      <xsd:anyAttribute processContents="skip" />
    </xsd:complexType>
  </xsd:element>
  <!-- Rules -->
  <xsd:element name="ruleSetAction">
    <xsd:complexType>
      <xsd:attribute name="ruleSet" type="xsd:string" use="required" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="rule">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
```

```
          <xsd:element ref="xsf:dialogBoxMessageAction" />
          <xsd:element ref="xsf:dialogBoxExpressionAction" />
          <xsd:element ref="xsf:switchViewAction" />
          <xsd:element ref="xsf:assignmentAction" />
          <xsd:element ref="xsf:changeAdapterProperty" />
          <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:signSignatureLineAction" />
          <xsd:element ref="xsf:closeDocumentAction" />
          <xsd:element ref="xsf:webPartConnectionAction" />
        </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="changeAdapterProperty">
    <xsd:complexType>
      <xsd:attribute name="adapter" type="xsd:string" use="required" />
      <xsd:attribute name="adapterProperty" 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="signSignatureLineAction">
      <xsd:complexType>
        <xsd:attribute name="matchCriteria" type="xsf:xdSignSignatureLineRuleEnum"
  use="required" />
        <xsd:attribute name="matchValue" type="xsd:string" use="required" />
        <xsd:attribute name="isExpression" type="xsf:xdYesNo" use="required" />
        <xsd:attribute name="signaturePictureEnabled" type="xsf:xdYesNo" use="required" />
        <xsd:attribute name="defaultSignaturePicture" type="xsd:string" use="optional" />
        <xsd:attribute name="isSignaturePictureExpression" type="xsf:xdYesNo" use="required" />
        <xsd:attribute name="checkHostEnabled" type="xsf:xdYesNo" use="required" />
        <xsd:attribute name="checkHost" type="xsd:string" use="optional" />
      </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="webPartConnectionAction" />
    <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 file

The following XML schema defines the types and elements used in the **XSF2 extensions** to the form definition (.xsf) file. The types and elements belong to the **XSF2 namespace** (http://schemas.microsoft.com/office/infopath/2006/solutionDefinition/extensions), as specified in section 2.2.2.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsf="http://schemas.microsoft.com/office/infopath/2003/solutionDefinition"

xmlns:xsf2="http://schemas.microsoft.com/office/infopath/2006/solutionDefinition/extensions"

targetNamespace="http://schemas.microsoft.com/office/infopath/2006/solutionDefinition/extensi
ons"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">
  <xsd:import
namespace="http://schemas.microsoft.com/office/infopath/2003/solutionDefinition"
schemaLocation="xsfschema.xsd" />

  <!--
    Please note that any changes to this schema requires the documented xsd to be updated as
well.
    It is more important because some of the types are open and do not list all the expected
values
    and the documentation is expected list them.
  -->

  <!-- Allowed values: submit, print, view, save, saveAs, close, refresh -->
  <xsd:simpleType name="serverCommandActionType">
    <xsd:restriction base="xsd:NMTOKEN">
      <xsd:pattern value="[a-zA-Z0-9_]*" />
    </xsd:restriction>
  </xsd:simpleType>
  <!-- Allowed values: none, xml, xmlXsn -->
  <xsd:simpleType name="emailAttachmentType">
    <xsd:restriction base="xsd:NMTOKEN">
      <xsd:pattern value="[a-zA-Z0-9_]*" />
    </xsd:restriction>
  </xsd:simpleType>
  <!-- Allowed values: client, server -->
  <xsd:simpleType name="compatibilityModesType">
    <xsd:restriction base="xsd:NMTOKEN">
      <xsd:pattern value="[a-zA-Z0-9_]*" />
    </xsd:restriction>
  </xsd:simpleType>
```

```
<!-- Allowed values: templatePart, formTemplate -->
<xsd:simpleType name="solutionType">
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:pattern value="[a-zA-Z0-9]*" />
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="formDescriptionType">
  <xsd:restriction base="xsd:string">
    <xsd:maxLength value="1024" />
    <xsd:minLength value="1" />
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="formLocaleType">
  <xsd:restriction base="xsd:token">
    <xsd:minLength value="1" />
  </xsd:restriction>
</xsd:simpleType>
<!-- Allowed values: CSharp, VisualBasic -->
<xsd:simpleType name="managedCodeType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[a-zA-Z0-9\.]*" />
  </xsd:restriction>
</xsd:simpleType>
<xsd:attributeGroup name="queryKeyFile">
  <xsd:attribute name="queryFile" type="xsd:string" use="optional" />
  <xsd:attribute name="queryKey" type="xsd:string" use="optional" />
</xsd:attributeGroup>

<!-- Root element for the xsf extension elements -->
<xsd:element name="solutionDefinition">
  <xsd:complexType>
    <xsd:all>
      <xsd:element ref="xsf2:server" minOccurs="0" />
      <xsd:element ref="xsf2:solutionPropertiesExtension" minOccurs="0" />
      <xsd:element ref="xsf2:mergedPrintView" minOccurs="0" />
      <xsd:element ref="xsf2:offline" minOccurs="0" />
      <xsd:element ref="xsf2:listPropertiesExtension" minOccurs="0" />
      <xsd:element ref="xsf2:dataConnections" minOccurs="0" />
      <xsd:element ref="xsf2:sendByMail" minOccurs="0" />
      <xsd:element ref="xsf2:warnings" minOccurs="0" />
      <xsd:element ref="xsf2:viewsExtension" minOccurs="0" />
      <xsd:element ref="xsf2:preview" minOccurs="0" />
      <xsd:element ref="xsf2:autoUpdatePrompt" minOccurs="0" />
      <xsd:element ref="xsf2:inputScopes" minOccurs="0" />
      <xsd:element ref="xsf2:managedCode" minOccurs="0" />
      <xsd:element ref="xsf2:submit" minOccurs="0" />
      <xsd:element ref="xsf2:featureRestrictionsExtension" minOccurs="0" />
    </xsd:all>
    <xsd:attribute name="runtimeCompatibility" use="required">
      <xsd:simpleType>
        <xsd:list itemType="xsf2:compatibilityModesType"/>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="solutionType" type="xsf2:solutionType" use="optional" />
    <xsd:attribute name="description" type="xsf2:formDescriptionType" use="optional" />
    <xsd:attribute name="allowClientOnlyCode" type="xsf:xdYesNo" use="optional"
default="no" />
```

```
      <xsd:attribute name="runtimeCompatibilityURL" type="xsd:string" use="optional"/>
      <xsd:attribute name="verifyOnServer" type="xsf:xdYesNo" use="optional"/>
      <xsd:anyAttribute processContents="skip" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="server">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf2:toolbar" minOccurs="0" />
      </xsd:sequence>
      <xsd:attribute name="formLocale" type="xsf2:formLocaleType" use="required" />
      <xsd:attribute name="isPreSubmitPostBackEnabled" type="xsf:xdYesNo" use="optional" />
      <xsd:attribute name="isMobileEnabled" type="xsf:xdYesNo" use="optional" />
      <xsd:anyAttribute processContents="skip" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="toolbar">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf2:commands" minOccurs="0" />
      </xsd:sequence>
      <xsd:attribute name="enabledTop" type="xsf:xdYesNo" use="optional" default="no" />
      <xsd:attribute name="enabledBottom" type="xsf:xdYesNo" use="optional" default="no" />
      <xsd:anyAttribute  processContents="skip" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="commands">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf2:command" maxOccurs="unbounded" minOccurs="0"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="command">
    <xsd:complexType>
      <xsd:sequence />
      <xsd:attribute name="action" type="xsf2:serverCommandActionType" use="required" />
      <xsd:attribute name="caption" type="xsf:xdTitle" use="optional" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="solutionPropertiesExtension">
    <xsd:complexType>
      <xsd:all>
        <xsd:element ref="xsf2:install" minOccurs="0" />
        <xsd:element ref="xsf2:wss" minOccurs="0" />
        <xsd:element ref="xsf2:contentType" minOccurs="0" />
        <xsd:element ref="xsf2:share" minOccurs="0" />
        <xsd:element ref="xsf2:mail" minOccurs="0" />
        <xsd:element ref="xsf2:admin" minOccurs="0" />
        <xsd:element ref="xsf2:contentTypeTemplate" minOccurs="0" />
        <xsd:element ref="xsf2:list" minOccurs="0" />
        <xsd:element ref="xsf2:entity" minOccurs="0" />
        <xsd:element ref="xsf2:workflowInitAssoc" minOccurs="0" />
        <xsd:element ref="xsf2:groove" 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:enumeration value="list" />
            <xsd:enumeration value="entity" />
            <xsd:enumeration value="workflowInitAssoc" />
            <xsd:enumeration value="groove" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:anyAttribute  processContents="skip" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="install">
    <xsd:complexType>
      <xsd:attribute name="companyName" type="xsd:string" use="required" />
      <xsd:attribute name="language" type="xsd:string" use="required" />
      <xsd:attribute name="path" type="xsd:string" use="required" />
      <xsd:attribute name="updatePath" type="xsd:string" use="optional" />
      <xsd:anyAttribute  processContents="skip" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="wss">
    <xsd:complexType>
      <xsd:sequence />
      <xsd:attribute name="path" type="xsd:string" use="required" />
      <xsd:attribute name="name" type="xsd:string" use="required" />
      <xsd:attribute name="description" type="xsd:string" use="required" />
      <xsd:attribute name="browserEnable" type="xsf:xdYesNo" use="optional" />
      <xsd:anyAttribute  processContents="skip" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="contentType">
    <xsd:complexType>
      <xsd:sequence />
      <xsd:attribute name="path" type="xsd:string" use="required" />
      <xsd:attribute name="sharepointContentTypeId" type="xsd:string" use="required" />
      <xsd:anyAttribute  processContents="skip" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="contentTypeTemplate">
    <xsd:complexType>
      <xsd:sequence />
      <xsd:attribute name="site" type="xsd:string" use="required" />
      <xsd:attribute name="path" type="xsd:string" use="required" />
      <xsd:attribute name="name" type="xsd:string" use="required" />
      <xsd:attribute name="description" type="xsd:string" use="required" />
      <xsd:attribute name="browserEnable" type="xsf:xdYesNo" use="optional" />
      <xsd:anyAttribute  processContents="skip" />
    </xsd:complexType>
```

```
</xsd:element>
<xsd:element name="share">
  <xsd:complexType>
    <xsd:sequence />
    <xsd:attribute name="formName" type="xsd:string" use="required" />
    <xsd:attribute name="path" type="xsd:string" use="required" />
    <xsd:attribute name="accessPath" type="xsd:string" use="required" />
    <xsd:anyAttribute  processContents="skip" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="mail">
  <xsd:complexType>
    <xsd:sequence />
    <xsd:attribute name="formName" type="xsd:string" use="required" />
    <xsd:anyAttribute  processContents="skip" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="admin">
  <xsd:complexType>
    <xsd:sequence />
    <xsd:attribute name="path" type="xsd:string" use="required" />
    <xsd:attribute name="site" type="xsd:string" use="required" />
    <xsd:anyAttribute  processContents="skip" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="list">
  <xsd:complexType>
    <xsd:sequence />
    <xsd:attribute name="path" type="xsd:string" use="required" />
    <xsd:anyAttribute  processContents="skip" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="entity">
  <xsd:complexType>
    <xsd:sequence />
    <xsd:attribute name="path" type="xsd:string" use="required" />
    <xsd:anyAttribute  processContents="skip" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="workflowInitAssoc">
  <xsd:complexType>
    <xsd:sequence />
    <xsd:attribute name="path" type="xsd:string" use="required" />
    <xsd:anyAttribute  processContents="skip" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="groove">
  <xsd:complexType>
    <xsd:sequence />
    <xsd:attribute name="path" type="xsd:string" use="required" />
    <xsd:anyAttribute  processContents="skip" />
  </xsd:complexType>
</xsd:element>

<xsd:element name="mergedPrintView">
  <xsd:complexType>
    <xsd:all>
```

```
        <xsd:element ref="xsf:printSettings" minOccurs="0" />
        <xsd:element ref="xsf2:includedViews" minOccurs="0" />
      </xsd:all>
      <xsd:attribute name="isDefault" type="xsf:xdYesNo" use="optional" default="no" />
      <xsd:attribute name="isCustomizable" type="xsf:xdYesNo" use="optional" default="no" />
      <xsd:attribute name="viewBreak" type="xsd:string" use="required" />
      <xsd:anyAttribute  processContents="skip" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="includedViews">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf2:includedView" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="includedView">
    <xsd:complexType>
      <xsd:sequence />
      <xsd:attribute name="name" type="xsf:xdViewName" use="required" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="offline">
    <xsd:complexType>
      <xsd:attribute name="openIfQueryFails" type="xsf:xdYesNo" default="no" use="optional"
/>
      <xsd:attribute name="cacheQueries" type="xsf:xdYesNo" default="no" use="optional" />
      <xsd:attribute name="expirationTime" type="xsd:nonNegativeInteger" use="optional" />
      <xsd:anyAttribute  processContents="skip" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="listPropertiesExtension">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf2:fieldsExtension" minOccurs="0"></xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="fieldsExtension">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf2:fieldExtension" maxOccurs="unbounded" minOccurs="1" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="fieldExtension">
    <xsd:complexType>
      <xsd:sequence>
      </xsd:sequence>
      <xsd:attribute name="columnName" type="xsd:string" use="required" />
      <xsd:attribute name="readWrite" type="xsf:xdYesNo" use="optional" default="no" />
      <xsd:attribute name="columnId" type="xsd:string" use="optional" />
      <xsd:anyAttribute  processContents="skip" />
    </xsd:complexType>
  </xsd:element>
```

```
  <xsd:element name="dataConnections">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf2:useHttpHandlerExtension" minOccurs="0" />
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
          <xsd:element ref="xsf2:davAdapterExtension" minOccurs="0" maxOccurs="unbounded" />
          <xsd:element ref="xsf2:adoAdapterExtension" minOccurs="0" maxOccurs="unbounded"/>
          <xsd:element ref="xsf2:webServiceAdapterExtension" minOccurs="0"
maxOccurs="unbounded" />
          <xsd:element ref="xsf2:emailAdapterExtension" minOccurs="0" maxOccurs="unbounded"
/>
          <xsd:element ref="xsf2:xmlFileAdapterExtension" minOccurs="0" maxOccurs="unbounded"
/>
          <xsd:element ref="xsf2:sharepointListAdapterExtension" minOccurs="0"
maxOccurs="unbounded" />
          <xsd:element ref="xsf2:sharepointListAdapterRWExtension" 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:attribute name="isRest" type="xsf:xdYesNo" use="optional"/>
      <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="sharepointListAdapterRWExtension">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf2:connectoid" minOccurs="0" />
      </xsd:sequence>
      <xsd:attribute name="ref" type="xsf:xdTitle" use="required" />
      <xsd:attribute name="queryThisFormOnly" type="xsf:xdYesNo" use="optional" default="no"
/>
      <xsd:attribute name="sharepointWebGuid" type="xsd:string" use="optional" />
      <xsd:attributeGroup ref="xsf2:queryKeyFile" />
    </xsd:complexType>
  </xsd:element>
```

```
    <xsd:element name="sendByMail">
      <xsd:complexType>
        <xsd:sequence />
        <xsd:attribute name="emailAttachmentType" type="xsf2:emailAttachmentType"
  use="optional" />
        <xsd:attribute name="disableEmailForms" type="xsf:xdYesNo" use="optional" />
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="warnings">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="xsf2:warning" maxOccurs="unbounded" minOccurs="0" />
        </xsd:sequence>
        <xsd:anyAttribute  processContents="skip" />
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="warning">
      <xsd:complexType>
        <xsd:sequence />
        <xsd:attribute name="source" type="xsd:string" use="required" />
        <xsd:attribute name="hidden" type="xsf:xdYesNo" use="optional" default="no" />
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="viewsExtension">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="xsf2:viewExtension" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="viewExtension">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="xsf2:xmlToEditExtension" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute name="ref" type="xsf:xdTitle" use="required" />
        <xsd:attribute name="designMode" type="xsd:string" use="optional" />
        <xsd:attribute name="readOnly" type="xsf:xdYesNo" use="optional" />
        <xsd:attribute name="clientOnly" type="xsf:xdYesNo" use="optional" default="no"/>
        <xsd:anyAttribute  processContents="skip" />
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="xmlToEditExtension">
      <xsd:complexType>
        <xsd:sequence/>
        <xsd:attribute name="ref" type="xsf:xdTitle" use="required" />
        <xsd:attribute name="excludeEmbeddedImages" type="xsf:xdYesNo" use="optional"
  default="no" />
        <xsd:attribute name="allowLinkedImages" type="xsf:xdYesNo" use="optional" default="no"
  />
        <xsd:anyAttribute  processContents="skip" />
      </xsd:complexType>
    </xsd:element>

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

```
    <xsd:complexType>
      <xsd:sequence />
      <xsd:attribute name="sampleData" type="xsd:string" use="optional" />
      <xsd:attribute name="domain" type="xsd:string" use="optional" />
      <xsd:attribute name="userRole" type="xsd:string" use="optional" />
      <xsd:anyAttribute  processContents="skip" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="autoUpdatePrompt">
    <xsd:complexType>
      <xsd:sequence />
      <xsd:attribute name="showPrompt" type="xsf:xdYesNo" use="optional" />
      <xsd:anyAttribute  processContents="skip" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="inputScopes">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf2:inputScope" maxOccurs="unbounded" minOccurs="0" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="inputScope">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf2:words" maxOccurs="unbounded" minOccurs="0" />
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:string" use="required" />
      <xsd:attribute name="caption" type="xsf:xdTitle" use="optional" />
      <xsd:attribute name="expression" type="xsd:string" use="optional" />
      <xsd:anyAttribute  processContents="skip" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="words">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="word" maxOccurs="unbounded" minOccurs="1">
          <xsd:complexType>
            <xsd:sequence />
            <xsd:attribute name="value" type="xsd:string" use="optional" default="" />
            <xsd:anyAttribute  processContents="skip" />
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
      <xsd:anyAttribute  processContents="skip" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="managedCode">
    <xsd:complexType>
      <xsd:sequence/>
      <xsd:attribute name="projectPath" type="xsd:string" use="optional" />
      <xsd:attribute name="language" type="xsf2:managedCodeType" use="required" />
      <xsd:attribute name="version" type="xsd:string" use="required" />
      <xsd:attribute name="enabled" type="xsf:xdYesNo" use="optional" />
```

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

  <xsd:element name="submit">
    <xsd:complexType>
      <xsd:all>
        <xsd:element name="submitAction" minOccurs="0">
          <xsd:complexType>
            <xsd:sequence />
            <xsd:attribute name="adapter" type="xsf:xdTitle" use="required" />
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="successMessage" type="xsd:string" minOccurs="0" />
        <xsd:element name="errorMessage" type="xsd:string" minOccurs="0" />
      </xsd:all>
      <xsd:attribute name="caption" type="xsd:string" use="optional" />
      <xsd:attribute name="onAfterSubmit" use="optional">
        <xsd:simpleType>
          <xsd:restriction base="xsd:NMTOKEN">
            <xsd:enumeration value="close" />
            <xsd:enumeration value="keepOpen" />
            <xsd:enumeration value="openNew" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="showStatusDialog" type="xsf:xdYesNo" use="optional" />
      <xsd:attribute name="showSignatureReminder" type="xsf:xdYesNo" use="optional" />
      <xsd:attribute name="disableMenuItem" type="xsf:xdYesNo" use="optional" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="featureRestrictionsExtension">
    <xsd:complexType>
      <xsd:all>
        <xsd:element ref="xsf2:exportToPDForXPS" minOccurs="0" />
      </xsd:all>
    </xsd:complexType>
  </xsd:element>

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

</xsd:schema>
```

5.3 The InfoPath XSF3 XSD file

The following XML schema defines the types and elements used in the **XSF3 extensions** to the form definition (.xsf) file. The types and elements belong to the **XSF3 namespace** (http://schemas.microsoft.com/office/infopath/2009/solutionDefinition/extensions), as specified in section 2.2.3.

```
<?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"

xmlns:xsf3="http://schemas.microsoft.com/office/infopath/2009/solutionDefinition/extensions"

targetNamespace="http://schemas.microsoft.com/office/infopath/2009/solutionDefinition/extensi
ons"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">
  <xsd:import
namespace="http://schemas.microsoft.com/office/infopath/2003/solutionDefinition"
schemaLocation="xsfschema.xsd" />
  <xsd:import
namespace="http://schemas.microsoft.com/office/infopath/2006/solutionDefinition/extensions"
schemaLocation="xsf2.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.
  -->

  <xsd:simpleType name="xdParameterType">
    <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="input" />
      <xsd:enumeration value="output" />
      <xsd:enumeration value="inputOutput" />
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="xdModeType">
    <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="entity" />
      <xsd:enumeration value="groove" />
      <xsd:enumeration value="list" />
      <xsd:enumeration value="workflowInitAssoc" />
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="xdLineStamp">
    <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="line" />
      <xsd:enumeration value="stamp" />
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:complexType name="xdSignatureLinePropertyType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="isCalculation" type="xsf:xdYesNo" use="required">
        </xsd:attribute>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
  <xsd:element name="solutionDefinition">
    <xsd:complexType>
```

```
      <xsd:all>
        <xsd:element ref="xsf3:webPartProperties" minOccurs="0" />
        <xsd:element ref="xsf3:viewsExtension" minOccurs="0" />
        <xsd:element ref="xsf3:customValidation" minOccurs="0" />
        <xsd:element ref="xsf3:baseUrl" minOccurs="0" maxOccurs="1" />
      </xsd:all>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="baseUrl">
    <xsd:complexType>
      <xsd:attribute name="relativeUrlBase" type="xsd:anyURI" use="optional" />
      <xsd:attribute name="queryServerInfo" type="xsf:xdYesNo" use="optional" />
      <xsd:anyAttribute processContents="skip" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="webPartProperties">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf3:webPartFields" minOccurs="0"></xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="webPartFields">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf3:webPartField" maxOccurs="unbounded" minOccurs="0" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="webPartField">
    <xsd:complexType>
      <xsd:sequence>
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:string" use="required" />
      <xsd:attribute name="node" type="xsd:string" use="required" />
      <xsd:attribute name="parameterType" type="xsf3:xdParameterType" use="required" />
      <xsd:anyAttribute processContents="skip" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="solutionPropertiesExtension2009">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf3:solutionMode" minOccurs="0" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="solutionMode">
    <xsd:complexType>
      <xsd:attribute name="autogenerated" type="xsf:xdYesNo" use="optional" />
      <xsd:attribute name="isListEditForm" type="xsf:xdYesNo" use="optional"/>
      <xsd:attribute name="mode" type="xsf3:xdModeType" use="required" />
      <xsd:attribute name="originalCtid" type="xsd:string" use="optional" />
      <xsd:attribute name="originalPublishUrl" type="xsd:anyURI" use="optional" />
<xsd:attribute name="originalPublishUrlFriendlyName" type="xsd:string" use="optional" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="viewsExtension">
```

```
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf3:viewExtension" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="viewExtension">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf3:xmlToEditExtension" minOccurs="0" maxOccurs="unbounded" />
        <xsd:element ref="xsf3:signatureLines" minOccurs="0" />
      </xsd:sequence>
      <xsd:attribute name="ref" type="xsf:xdTitle" use="required" />
      <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="excludeTables" type="xsf:xdYesNo" use="optional" default="no" />
      <xsd:attribute name="excludeHyperlink" type="xsf:xdYesNo" use="optional" default="no"
/>
      <xsd:anyAttribute  processContents="skip" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="signatureLines">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf3:signatureLine" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:complexType>
    <!-- signature line names must be unique -->
    <xsd:unique name="signature_line_names_unique">
      <xsd:selector xpath="./xsf3:signatureLine" />
      <xsd:field xpath="@name" />
    </xsd:unique>
  </xsd:element>
  <xsd:element name="signatureLine">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="confirmationMessage" type="xsf:xdSignedDataBlockMessage"
minOccurs="0" />
        <xsd:element name="suggestedSignerName" type="xsf3:xdSignatureLinePropertyType"
minOccurs="0" />
        <xsd:element name="suggestedSignerTitle" type="xsf3:xdSignatureLinePropertyType"
minOccurs="0" />
        <xsd:element name="suggestedSignerEmailAddress"
type="xsf3:xdSignatureLinePropertyType" minOccurs="0" />
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:NMTOKEN" use="required" />
      <xsd:attribute name="signedDataBlock" type="xsd:string" use="required" />
      <xsd:attribute name="showDate" type="xsf:xdYesNo" use="required" />
      <xsd:attribute name="signatureType" type="xsf3:xdLineStamp" use="required" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="customValidation">
    <xsd:complexType>
```

```
      <xsd:sequence>
        <xsd:element ref="xsf3:errorBlank" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="errorBlank">
    <xsd:complexType>
      <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="required" />
      <xsd:anyAttribute  processContents="skip" />
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

5.4 The Built-In ActiveX Controls XSD file

The following XML schema defines the types and elements used by the **built-in ActiveX controls** contact selector, as specified in sections 2.3.1.3, and external item picker, as specified in section 2.3.2.4.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- _lcid="1033" _version="" -->
<!-- _LocalBinding -->
<xs:schema
targetNamespace="http://schemas.microsoft.com/office/infopath/2007/PartnerControls"
elementFormDefault="qualified" attributeFormDefault="unqualified"
xmlns:pc="http://schemas.microsoft.com/office/infopath/2007/PartnerControls"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="Person">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="pc:DisplayName" minOccurs="0"/>
                <xs:element ref="pc:AccountId" minOccurs="0"/>
                <xs:element ref="pc:AccountType" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="DisplayName" type="xs:string"/>
    <xs:element name="AccountId" type="xs:string"/>
    <xs:element name="AccountType" type="xs:string"/>

    <xs:element name="BDCAssociatedEntity">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="pc:BDCEntity" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>

            <xs:attribute ref="pc:EntityNamespace"/>
            <xs:attribute ref="pc:EntityName"/>
            <xs:attribute ref="pc:SystemInstanceName"/>
            <xs:attribute ref="pc:AssociationName"/>
        </xs:complexType>
    </xs:element>
```

```
    <xs:attribute name="EntityNamespace" type="xs:string"/>
    <xs:attribute name="EntityName" type="xs:string"/>
    <xs:attribute name="SystemInstanceName" type="xs:string"/>
    <xs:attribute name="AssociationName" type="xs:string"/>

    <xs:element name="BDCEntity">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="pc:EntityDisplayName" minOccurs="0"/>
                <xs:element ref="pc:EntityInstanceReference" minOccurs="0"/>
                <xs:element ref="pc:EntityId1" minOccurs="0"/>
                <xs:element ref="pc:EntityId2" minOccurs="0"/>
                <xs:element ref="pc:EntityId3" minOccurs="0"/>
                <xs:element ref="pc:EntityId4" minOccurs="0"/>
                <xs:element ref="pc:EntityId5" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="EntityDisplayName" type="xs:string"/>
    <xs:element name="EntityInstanceReference" type="xs:string"/>
    <xs:element name="EntityId1" type="xs:string"/>
    <xs:element name="EntityId2" type="xs:string"/>
    <xs:element name="EntityId3" type="xs:string"/>
    <xs:element name="EntityId4" type="xs:string"/>
    <xs:element name="EntityId5" type="xs:string"/>

    <xs:element name="Terms">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="pc:TermInfo" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="TermInfo">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="pc:TermName" minOccurs="0"/>
                <xs:element ref="pc:TermId" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="TermName" type="xs:string"/>
    <xs:element name="TermId" type="xs:string"/>
</xs: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® InfoPath® 2010
- Microsoft® SharePoint® Server 2010
- Microsoft® InfoPath® 15 Technical Preview
- Microsoft® SharePoint® Server 15 Technical Preview

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

<1> Section 2.2.1.2.87: SharePoint Server 2010 has the following behavior: The value for this property is interpreted as "immediate".

<2> Section 2.2.2.2.2: SharePoint Server 2010 has the following behavior: The formLocale attribute is ignored.

<3> Section 2.4.1.1: SharePoint Server 2010 has the following behavior: The maximum value allowed for this property is 9999.

<4> Section 2.4.2.11: SharePoint Server 2010 has the following behavior: When the **dateFormat** item has a value of **Short Date**, **Long Date** or **Year Month**, the **locale** item is ignored and the locale is determined based on the locale of the protocol server.

<5> Section 2.4.2.37.10: SharePoint Server 2010 has the following behavior: The maximum allowed value 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 380
action element - form definition file 105
admin element - form definition file extension 173
adoAdapter element - form definition file 68
adoAdapterExtension element - form definition file extension 181
allowedActions element - form definition file 104
allowedControl element - form definition file 112
allowedTasks element - form definition file 105
AllowMultiple attribute - form view file 403
allownonmatching attribute - form view file 380
Applicability (section 1.5 30, section 1.5 30)
applicationParameters element - form definition file 58
assignmentAction element - form definition file 144
attachmentFileName element - form definition file 82
attributeData element - form definition file 114
autoAdvance attribute - form view file 381
autoRecovery element - form definition file 63
autoUpdatePrompt element - form definition file extension 189
auxDom attribute - form view file 381

B

backgroundPicture attribute - form view file 381
baseUrl element - form definition file extension 202
bcc element - form definition file 81
bdcAdapter element - form definition file 149
binding attribute - form view file 381
binding_secondary attribute - form view file 403
bindingProperty attribute - form view file 384
bindingType attribute - form view file 384
boundProp attribute - form view file 384
boundPropSecondary attribute - form view file 403
Built-in ActiveX controls XSD schema 539
business object - form template file component 33
button (with event) element - form definition file 121
Button control - form view file 274
Button control - XML schema file 214
Button control example 428
button element - form definition file 115

C

CAB format
  form definition (.xsf) file 19
  form template (.xsn) file 18
  form view files (XSLT) 19
  print view files (XSLT) 28
  resource files 30
  submit files (XML) 29
  template (XML) file 29
  unused files 30
  upgrade XSL file 29
  XML schema files (XSD) 19
calculatedField element - form definition file 148
calculations element - form definition file 147
cc element - form definition file 80
Change tracking 542
Check box control - form view file 280
Check box control - XML schema file 214
Check box control example 430
Choice group/choice section control - XML schema file 222
Choice group/choice section control example 452
Choice group/section control - form view file 332
chooseFragment element - form definition file 117
closeDocumentAction element - form definition file 146
Combo box control - form view file 334
Combo box control - XML schema file 223
Combo box control example 453
command element - form definition file extension 167
commands element - form definition file extension 167
compatibilityModesType simple type - form definition file extension 162
Complex form template example 417
confirmationMessage element - form definition file extension 209
connectoid element - form definition file extension 179
Contact selector control - form view file 282
Contact selector control - XML schema file 215
Contact selector control example 430
contentType element - form definition file extension 171
contentTypeTemplate element - form definition file extension 171
Control data formatting - form view file 272
Control representation examples – form view file 428
Control-specific attributes example 476
CtrlId attribute - form view file 386
customCategory element - form definition file 108
customValidation element - form definition file 86
customValidation element - form definition file extension 211

D

dataAdapters element - form definition file 83
dataConnections element - form definition file extension 177
dataFields element - submit file 414
datafmt attribute - form view file 387
datafmt2 attribute - form view file (section 2.4.2.37.4 404, section 2.4.2.37.11 407)

dataObject element - form definition file 66
dataObjects element - form definition file 65
Date picker control - form view file 285
Date picker control - XML schema file 215
Date picker control example 431
Date/time picker control - form view file 344
Date/time picker control - XML schema file 224
Date/time picker control example 459
davAdapter element - form definition file 76
davAdapterExtension element - form definition file extension 180
Details
action attribute - form view file 380
action element - form definition file 105
admin element - form definition file extension 173
adoAdapter element - form definition file 68
adoAdapterExtension element - form definition file extension 181
allowedActions element - form definition file 104
allowedControl element - form definition file 112
allowedTasks element - form definition file 105
AllowMultiple attribute - form view file 403
allownonmatching attribute - form view file 380
applicationParameters element - form definition file 58
assignmentAction element - form definition file 144
attachmentFileName element - form definition file 82
attributeData element - form definition file 114
autoAdvance attribute - form view file 381
autoRecovery element - form definition file 63
autoUpdatePrompt element - form definition file extension 189
auxDom attribute - form view file 381
backgroundPicture attribute - form view file 381
baseUrl element - form definition file extension 202
bcc element - form definition file 81
bdcAdapter element - form definition file 149
binding attribute - form view file 381
binding_secondary attribute - form view file 403
bindingProperty attribute - form view file 384
bindingType attribute - form view file 384
boundProp attribute - form view file 384
boundPropSecondary attribute - form view file 403
business object - form template file component 33
button (with event) element - form definition file 121
button control - form view file 274
button control - XML schema file 214
button element - form definition file 115
calculatedField element - form definition file 148
calculations element - form definition file 147
cc element - form definition file 80
check box control - form view file 280
check box control - XML schema file 214
choice group/choice section control - XML schema file 222
choice group/section control - form view file 332
chooseFragment element - form definition file 117
closeDocumentAction element - form definition file 146
combo box control - form view file 334
combo box control - XML schema file 223
command element - form definition file extension 167
commands element - form definition file extension 167
compatibilityModesType simple type - form definition file extension 162
confirmationMessage element - form definition file extension 209
connectoid element - form definition file extension 179
contact selector control - form view file 282
contact selector control - XML schema file 215
contentType element - form definition file extension 171
contentTypeTemplate element - form definition file extension 171
control data formatting - form view file 272
CtrlId attribute - form view file 386
customCategory element - form definition file 108
customValidation element - form definition file 86
customValidation element - form definition file extension 211
dataAdapters element - form definition file 83
dataConnections element - form definition file extension 177
dataFields element - submit file 414
datafmt attribute - form view file 387
datafmt2 attribute - form view file (section 2.4.2.37.4 404, section 2.4.2.37.11 407)
dataObject element - form definition file 66
dataObjects element - form definition file 65
date picker control - form view file 285
date picker control - XML schema file 215
date/time picker control - form view file 344
date/time picker control - XML schema file 224
davAdapter element - form definition file 76
davAdapterExtension element - form definition file extension 180
dialogBoxExpressionAction element - form definition file 143
dialogBoxMessageAction element - form definition file 143
disableEditing attribute - form view file 393
documentSchema element - form definition file 85
documentSchemas element - form definition file 84
documentSignatures element - form definition file 135
documentVersionUpgrade element - form definition file 137

domEventHandler element - form definition file 89
domEventHandlers element - form definition file 88
drop-down list control - form view file 291
drop-down list control - XML schema file 216
editing element - form definition file 122
editWith element - form definition file 117
emailAdapter element - form definition file 78
emailAdapterExtension element - form definition file extension 183
emailAttachmentType simple type - form definition file extension 161
embedded picture control - form view file 348
embedded picture control - XML schema file 225
enabledProperty attribute - form view file 393
enabledValue attribute - form view file 394
entity element - form definition file extension 196
entity picker control - form view file 345
entity picker control - XML schema file 224
errorBlank element - form definition file extension 211
errorCondition element - form definition file 86
errorMessage element - form definition file (section 2.2.1.2.45 87, section 2.2.1.2.56 97)
errorMessage element - form definition file extension 194
exitRuleSet element - form definition file 142
exportToExcel element - form definition file 62
exportToPDForXPS element - form definition file extension 195
exportToWeb element - form definition file 61
expression box control - form view file 298
expression box control - XML schema file 216
extension element - form definition file 139
extensions element - form definition file 138
externalView element - form definition file 114
externalViews element - form definition file 113
featureRestrictions element - form definition file 60
featureRestrictionsExtension element - form definition file extension 195
field (grooveAdapter) element - form definition file 151
field (list view) element - form definition file 92
field (ListAdapter) element - form definition file 156
field element - form definition file 75
fieldExtension element - form definition file extension 177
fields (list view) element - form definition file 91
fieldsExtension element - form definition file extension 176
file attachment control - form view file 301
file attachment control - XML schema file 216
file element - form definition file 109
fileName element - form definition file 77
fileNew element - form definition file 106
fileProperties element - form definition file 110
files element - form definition file 109
folderURL element - form definition file 77
footer element - form definition file 128
form definition file (XSF2) extension specification 158
form definition file (XSF2) specification 34
form view file - control-specific attributes 377
form view file - view representation 231
form view file - XSL function extensions 407
formDescriptionType simple type - form definition file extension 162
formLocaleType simple type - form definition file extension 163
fragmentToInsert element - form definition file 123
getUserNameFromData element - form definition file 102
ghosted attribute - form view file 394
groove element - form definition file extension 197
grooveAdapter element - form definition file 150
group element - form definition file 103
header element - form definition file 127
HideInPrintView attribute - form view file 405
HoverSrc attribute - form view file 405
hwsAdapter element - form definition file 70
hwsOperation element - form definition file 71
hwsWorkflow element - form definition file 103
hyperlink control - form view file 301
hyperlink control - XML schema file 217
hyperlink input control - form view file 350
hyperlink input control - XML schema file 225
ictID attribute - form view file 394
ictVersion attribute - form view file 394
ignored controls - form view file 376
importerrors.xml - form template file component 33
importParameters element - form definition file 89
importSource element - form definition file 90
includedView element - form definition file extension 175
includedViews element - form definition file extension 174
initialXmlDocument element - form definition file 107
inline attribute - form view file 394
innerCtrl attribute - form view file 395
input element - form definition file 72
inputscope attribute - form view file 395
inputScope element - form definition file extension 190
inputScopeId attribute - form view file 395
inputScopes element - form definition file extension 190
install element - form definition file extension 170
intro element - form definition file 82
invalid constructs - form view file 377
invalid controls - form view file 376
irm_template - form template file component 33
layoutText attribute - form view file 395
linked picture control - form view file 360
linked picture control - XML schema file 228

linkedToMaster attribute - form view file 396
list box control - form view file 303
list box control - XML schema file 217
list controls - form view file 358
list controls - XML schema file 227
list element - form definition file extension 196
listProperties element - form definition file 91
listPropertiesExtension element - form definition file extension 176
location element - form definition file 104
mail element - form definition file extension 172
mainpane element - form definition file 124
managedCode element - form definition file extension 192
managedCodeType simple type - form definition file extension 163
manifest.xsf - form template file component 32
masterDetail element - form definition file 123
masterID attribute - form view file 396
masterName attribute - form view file 396
membership element - form definition file 101
menu element - form definition file 129
merge.xsl - form template file component 33
mergedPrintView element - form definition file extension 173
message element - form definition file 137
msxsl function extension - form view file 408
multiple-selection list box control - form view file 363
multiple-selection list box control - XML schema file 230
myFields element - submit file 414
num attribute - form view file 396
offline element - form definition file extension 175
offValue attribute - form view file 397
onLoad element - form definition file 98
onValue attribute - form view file 397
openNewDocumentAction element - form definition file 145
operation element - form definition file 70
option button control - form view file 306
option button control - XML schema file 218
override element - form definition file 58
package element - form definition file 108
partFragment element - form definition file 72
permissions element - form definition file 112
picture button control - form view file 372
picture button control - XML schema file 230
postbackModel attribute - form view file 397
preview element - form definition file extension 189
primaryschema.xsd - form template file component 32
print element - form definition file 62
printSettings element - form definition file 124
property element - form definition file 110
query (secondary) element - form definition file 67
query element - form definition file 63
queryAction element - form definition file 145
ref attribute - form view file 399
relativeQuery element - form definition file extension 182
repeating section control - form view file 308
repeating section control - XML schema file 218
repeating table control - form view file 310
repeating table control - XML schema file 219
resource files - form template file component 33
rich text box control - form view file 313
rich text box control - XML schema file 220
role element - form definition file 101
roles element - form definition file 99
rule element - form definition file 140
ruleSet element - form definition file 146
ruleSetAction element - form definition file 139
ruleSets element - form definition file 147
sampledata.xml - form template file component 32
save (disabled) element - form definition file 99
save element - form definition file 61
schemaErrorMessages element - form definition file 57
script element - form definition file 65
script.js - form template file component 33
script.vbs - form template file component 33
scripts element - form definition file 64
SearchPeopleOnly attribute - form view file 405
secondaryschema.xsd - form template file component 32
secondaryschema_offline.xml - form template file component 33
section/optional section control - form view file 315
section/optional section control - XML schema file 220
sendByMail element - form definition file extension 185
sendMail element - form definition file 63
server attribute - form view file 406
server element - form definition file extension 165
serverCommandActionType simple type - form definition file extension 161
share element - form definition file extension 172
SharePoint file attachment control - form view file 375
SharePoint file attachment control - XML schema file 231
SharePointGroup attribute - form view file 406
sharepointListAdapter element - form definition file 74
sharepointListAdapterExtension element - form definition file extension 184
sharepointListAdapterRW element - form definition file 152
sharepointListAdapterRWExtension element - form definition file extension 197
SignatureBlock attribute - form view file 399
signatureLine element - form definition file extension 208

signatureLines element - form definition file extension 207
signedDataBlock element - form definition file 136
SignedSectionDisplaySignatures attribute - form view file 399
SignedSectionName attribute - form view file 400
signSignatureLineAction element - form definition file 158
solutio9nPropertiesExtension2009 element - form definition file extension 204
solutionDefinition element - form definition file extension (section 2.2.2.2.1 164, section 2.2.3.2.1 201)
solutionMode element - form definition file extension 205
solutionProperties element - form definition file 59
solutionPropertiesExtension element - form definition file extension 168
solutionType simple type - form definition file extension 162
subject element - form definition file 81
submit element - form definition file 94
submit element - form definition file extension 193
submitAction element - form definition file (section 2.2.1.2.54 96, section 2.2.1.2.115 142)
submitAction element - form definition file extension 194
submitdata.xml - form template file component 33
submitToHostAdapter element - form definition file 83
successMessage element - form definition file 97
successMessage element - form definition file extension 194
suggestedSignerEmailAddress element - form definition file extension 210
suggestedSignerName element - form definition file extension 209
suggestedSignerTitle element - form definition file extension 210
switchViewAction element - form definition file 143
table control - form view file 322
table control - XML schema file 221
task element - form definition file 106
taskpane element - form definition file 131
template.xml - form template file component 32
text box control - form view file 323
text box control - XML schema file 221
to element - form definition file 79
toolbar element - form definition file 128
toolbar element - form definition file extension 166
unboundControls element - form definition file 121
upgrade.xsl - form template file component 33
useHttpHandler element - form definition file 97
useHttpHandlerExtension element - form definition file extension 178
useQueryAdapter element - form definition file 98
userName element - form definition file 102
useScriptHandler element - form definition file 98
useTransform element - form definition file 137
value attribute - form view file 400
view element - form definition file 132
view syntax - form view file 233
view.xsl - form template file component 32
viewExtension element - form definition file extension (section 2.2.2.2.35 187, section 2.2.3.2.9 206)
views element - form definition file 131
viewsExtension element - form definition file extension (section 2.2.2.2.34 187, section 2.2.3.2.8 206)
warning element - form definition file extension 186
warnings element - form definition file extension 186
webPartConnectionAction element - form definition file 157
webPartField element - form definition file extension 204
webPartFields element - form definition file extension 203
webPartProperties element - form definition file extension 203
webServiceAdapter element - form definition file 69
webServiceAdapterExtension element - form definition file extension 181
widgetIndex attribute - form view file 406
word element - form definition file extension 192
words element - form definition file extension 191
workflowInitAssoc element - form definition file extension 197
wss element - form definition file extension 170
xctname attribute - form view file 400
xdDate function extension - form view file 408
xdDesignMode simple type - form definition file 50
xdEmptyString simple type - form definition file 50
xdEnabledDisabled simple type - form definition file 46
xdEnvironment function extension - form view file 409
xdErrorMessage simple type - form definition file 50
xdExpressionLiteral simple type - form definition file 47
xdFileName simple type - form definition file 48
xdFormatting function extension - form view file 410
xdHWSCaption simple type - form definition file 53
xdHWSname simple type - form definition file 52
xdImage function extension - form view file (section 2.4.3.5 410, section 2.4.3.11 413)

xdLineStamp simple type - form definition file extension 201
xdManualAuto simple type - form definition file 47
xdMath function extension - form view file 410
xdModeType simple type - form definition file extension 200
xDocumentClass element - form definition file 54
xdParameterType simple type - form definition file extension 199
xdRoleName simple type - form definition file 42
xdScriptLanguage simple type - form definition file 49
xdServerInfo function extension - form view file 413
xdSignatureRelationEnum simple type - form definition file 52
xdSignedDataBlockMessage simple type - form definition file 52
xdSignedDataBlockName simple type - form definition file 51
xdSignSignatureLineRuleEnum simple type - form definition file 53
xdSolutionVersion simple type - form definition file 49
xdTitle simple type - form definition file 40
xdUser function extension - form view file 411
xdUtil function extension - form view file 412
xdViewName simple type - form definition file 42
xdXDocument function extension - form view file 412
xdYesNo simple type - form definition file 43
XML schema file - control representation 212
xmlFileAdapter element - form definition file 73
xmlFileAdapterExtension element - form definition file extension 183
xmlToEdit attribute - form view file 402
xmlToEdit element - form definition file 134
xmlToEditExtension element - form definition file extension (section 2.2.2.2.36 188, section 2.2.3.2.10 207)
XSF Elements - form definition file 54
XSF Enumerations - form definition file 40
XSF2 Elements - form definition file 163
XSF2 Enumerations - form definition file 161
XSF3 Elements - form definition file 201
XSF3 Enumerations - form definition file 199
XSL root template - form view file 239
XSL root template style sheets - form view file 241
dialogBoxExpressionAction element - form definition file 143
dialogBoxMessageAction element - form definition file 143
disableEditing attribute - form view file 393
documentSchema element - form definition file 85
documentSchemas element - form definition file 84
documentSignatures element - form definition file 135
documentVersionUpgrade element - form definition file 137
domEventHandler element - form definition file 89
domEventHandlers element - form definition file 88
Drop-down list control - form view file 291
Drop-down list control - XML schema file 216
Drop-down list control example 433

E

editing element - form definition file 122
editWith element - form definition file 117
emailAdapter element - form definition file 78
emailAdapterExtension element - form definition file extension 183
emailAttachmentType simple type - form definition file extension 161
Embedded picture control - form view file 348
Embedded picture control - XML schema file 225
Embedded picture control example 461
enabledProperty attribute - form view file 393
enabledValue attribute - form view file 394
entity element - form definition file extension 196
Entity picker control - form view file 345
Entity picker control - XML schema file 224
Entity picker control example 460
errorBlank element - form definition file extension 211
errorCondition element - form definition file 86
errorMessage element - form definition file (section 2.2.1.2.45 87, section 2.2.1.2.56 97)
errorMessage element - form definition file extension 194
Examples
button control 428
check box control 430
choice group/choice section control 452
combo box control 453
complex form template 417
contact selector control 430
Control representation – form view file 428
control-specific attributes 476
date picker control 431
date/time picker control 459
drop-down list control 433
embedded picture control 461
entity picker control 460
expression box control 437
file attachment control 437
form definition file 417
form definition file - browser-compatible form 417
form definition file - list form 421
form view file 428
hyperlink control 438
hyperlink input control 462
linked picture control 466
list box control 438
list controls 464
MSXSL node set() 491
multiple-selection list box control 467
option button control 440
overview 416
picture button control 473

print view file 486
repeating section control 442
repeating table control 442
rich text box control 446
section/optional section control 446
SharePoint file attachment control 475
simple form template 416
submit file 488
table control 449
template.(XML) 489
text box control 450
upgrade (XSL) 489
XML schema file 427
XSL function extensions 482
exitRuleSet element - form definition file 142
exportToExcel element - form definition file 62
exportToPDForXPS element - form definition file extension 195
exportToWeb element - form definition file 61
Expression box control - form view file 298
Expression box control - XML schema file 216
Expression box control example 437
extension element - form definition file 139
extensions element - form definition file 138
externalView element - form definition file 114
externalViews element - form definition file 113

F

featureRestrictions element - form definition file 60
featureRestrictionsExtension element - form definition file extension 195
field (grooveAdapter) element - form definition file 151
field (list view) element - form definition file 92
field (ListAdapter) element - form definition file 156
field element - form definition file 75
fieldExtension element - form definition file extension 177
Fields
vendor-extensible 31
Fields - vendor-extensible 31
fields (list view) element - form definition file 91
fieldsExtension element - form definition file extension 176
File attachment control - form view file 301
File attachment control - XML schema file 216
File attachment control example 437
file element - form definition file 109
fileName element - form definition file 77
fileNew element - form definition file 106
fileProperties element - form definition file 110
files element - form definition file 109
folderURL element - form definition file 77
footer element - form definition file 128
Form definition (.xsf) file 19
Form Definition (XSF) file structure 34
Form definition file
XSF Elements 54
XSF Enumerations 40
XSF2 Elements 163
XSF2 Enumerations 161
XSF3 Elements 201
XSF3 Enumerations 199
Form definition file (XSF2) extension specification 158
Form definition file (XSF2) specification 34
Form definition file elements
action 105
adoAdapter 68
allowedActions 104
allowedControl 112
allowedTasks 105
applicationParameters 58
assignmentAction 144
attachmentFileName 82
attributeData 114
autoRecovery 63
bcc 81
bdcAdapter 149
button 115
button (with event) 121
calculatedField 148
calculations 147
cc 80
chooseFragment 117
closeDocumentAction 146
customCategory 108
customValidation 86
dataAdapters 83
dataObject 66
dataObjects 65
davAdapter 76
dialogBoxExpressionAction 143
dialogBoxMessageAction 143
documentSchema 85
documentSchemas 84
documentSignatures 135
documentVersionUpgrade 137
domEventHandler 89
domEventHandlers 88
editing 122
editWith 117
emailAdapter 78
errorCondition 86
errorMessage (section 2.2.1.2.45 87, section 2.2.1.2.56 97)
exitRuleSet 142
exportToExcel 62
exportToWeb 61
extension 139
extensions 138
externalView 114
externalViews 113
featureRestrictions 60
field 75
field (grooveAdapter) 151
field (list view) 92
field (ListAdapter) 156
fields (list view) 91
file 109
fileName 77
fileNew 106

fileProperties 110
files 109
folderURL 77
footer 128
fragmentToInsert 123
getUserNameFromData 102
grooveAdapter 150
group 103
header 127
hwsAdapter 70
hwsOperation 71
hwsWorkflow 103
importParameters 89
importSource 90
initialXmlDocument 107
input 72
intro 82
listProperties 91
location 104
mainpane 124
masterDetail 123
membership 101
menu 129
message 137
onLoad 98
openNewDocumentAction 145
operation 70
override 58
package 108
partFragment 72
permissions 112
print 62
printSettings 124
property 110
query 63
query (secondary) 67
queryAction 145
role 101
roles 99
rule 140
ruleSet 146
ruleSetAction 139
ruleSets 147
save 61
save (disabled) 99
schemaErrorMessages 57
script 65
scripts 64
sendMail 63
sharepointListAdapter 74
sharepointListAdapterRW 152
signedDataBlock 136
signSignatureLineAction 158
solutionProperties 59
subject 81
submit 94
submitAction (section 2.2.1.2.54 96, section 2.2.1.2.115 142)
submitToHostAdapter 83
successMessage 97
switchViewAction 143
task 106
taskpane 131
to 79
toolbar 128
unboundControls 121
useHttpHandler 97
useQueryAdapter 98
userName 102
useScriptHandler 98
useTransform 137
views 131
webPartConnectionAction 157
webServiceAdapter 69
xDocumentClass 54
xmlFileAdapter 73
xmlToEdit (section 2.2.1.2.104 132, section 2.2.1.2.105 134)
Form definition file example 417
Form definition file examples – browser-compatible form 417
Form definition file examples – list form 421
Form definition file extension elements
admin 173
adoAdapterExtension 181
autoUpdatePrompt 189
baseUrl 202
command 167
commands 167
confirmationMessage 209
connectoid 179
contentType 171
contentTypeTemplate 171
customValidation 211
dataConnections 177
davAdapterExtension 180
emailAdapterExtension 183
entity 196
errorBlank 211
errorMessage 194
exportToPDFforXPS 195
featureRestrictionsExtension 195
fieldExtension 177
fieldsExtension 176
groove 197
includedView 175
includedViews 174
inputScope 190
inputScopes 190
install 170
list 196
listPropertiesExtension 176
mail 172
managedCode 192
mergedPrintView 173
offline 175
preview 189
relativeQuery 182
sendByMail 185
server 165
share 172
sharepointListAdapterExtension 184

sharepointListAdapterRWExtension 197
signatureLine 208
signatureLines 207
solutio9nPropertiesExtension2009 204
solutionDefinition (section 2.2.2.2.1 164, section 2.2.3.2.1 201)
solutionMode 205
solutionPropertiesExtension 168
submit 193
submitAction 194
successMessage 194
suggestedSignerEmailAddress 210
suggestedSignerName 209
suggestedSignerTitle 210
toolbar 166
useHttpHandlerExtension 178
viewExtension (section 2.2.2.2.35 187, section 2.2.3.2.9 206)
viewsExtension (section 2.2.2.2.34 187, section 2.2.3.2.8 206)
warning 186
warnings 186
webPartField 204
webPartFields 203
webPartProperties 203
webServiceAdapterExtension 181
word 192
words 191
workflowInitAssoc 197
wss 170
xmlFileAdapterExtension 183
xmlToEditExtension (section 2.2.2.2.36 188, section 2.2.3.2.10 207)
Form definition file extension simple types
compatibilityModesType 162
emailAttachmentType 161
formDescriptionType 162
formLocaleType 163
managedCodeType 163
serverCommandActionType 161
solutionType 162
xdLineStamp 201
xdModeType 200
xdParameterType 199
Form definition file simple types
xdDesignMode 50
xdEmptyString 50
xdEnabledDisabled 46
xdErrorMessage 50
xdExpressionLiteral 47
xdFileName 48
xdHWSCaption 53
xdHWSname 52
xdManualAuto 47
xdRoleName 42
xdScriptLanguage 49
xdSignatureRelationEnum 52
xdSignedDataBlockMessage 52
xdSignedDataBlockName 51
xdSignSignatureLineRuleEnum 53
xdSolutionVersion 49
xdTitle 40
xdViewName 42
xdYesNo 43
Form template (.xsn) file 18
Form template file components
business object 33
importerrors.xml 33
irm_template 33
manifest.xsf 32
merge.xsl 33
primaryschema.xsd 32
resource files 33
sampledata.xml 32
script.js 33
script.vbs 33
secondaryschema.xsd 32
secondaryschema_offline.xml 33
submitdata.xml 33
template.xml 32
upgrade.xsl 33
view.xsl 32
Form view file
control data formatting 272
invalid constructs 377
invalid controls 376
view syntax 233
XSL root template 239
XSL root template style sheets 241
Form view file - control-specific attributes 377
Form view file - view representation 231
Form view file - XSL function extensions 407
Form view file attributes
action 380
AllowMultiple 403
allownonmatching 380
autoAdvance 381
auxDom 381
backgroundPicture 381
binding 381
binding_secondary 403
bindingProperty 384
bindingType 384
boundProp 384
boundPropSecondary 403
CtrlId 386
datafmt 387
datafmt2 (section 2.4.2.37.4 404, section 2.4.2.37.11 407)
disableEditing 393
enabledProperty 393
enabledValue 394
ghosted 394
HideInPrintView 405
HoverSrc 405
ictID 394
ictVersion 394
inline 394
innerCtrl 395
inputscope 395
inputScopeId 395
layoutText 395

linkedToMaster 396
masterID 396
masterName 396
num 396
offValue 397
onValue 397
postbackModel 397
ref 399
SearchPeopleOnly 405
server 406
SharePointGroup 406
SignatureBlock 399
SignedSectionDisplaySignatures 399
SignedSectionName 400
value 400
widgetIndex 406
xctname 400
xmlToEdit 402
Form view file controls
button 274
check box 280
choice group/section 332
combo box 334
contact selector 282
date picker 285
date/time picker 344
drop-down list 291
embedded picture 348
entity picker 345
expression box 298
file attachment 301
hyperlink 301
hyperlink input 350
Ignored 376
linked picture 360
list 358
list box 303
multiple-selection list box 363
option button 306
picture button 372
repeating section 308
repeating table 310
rich text box 313
section/optional section 315
SharePoint file attachment 375
table 322
text box 323
Form view file example 428
Form view file function extensions
msxsl 408
xdDate 408
xdEnvironment 409
xdFormatting 410
xdImage (section 2.4.3.5 410, section 2.4.3.11 413)
xdMath 410
xdServerInfo 413
xdUser 411
xdUtil 412
xdXDocument 412
Form view files (XSLT) 19
Form view files (XSLT) structure 231
formDescriptionType simple type - form definition file extension 162
formLocaleType simple type - form definition file extension 163
fragmentToInsert element - form definition file 123
Full XML schema
built-in ActiveX controls XSD file 539
InfoPath XSF XSD file 494
InfoPath XSF2 XSD file 525
InfoPath XSF3 XSD file 535
Full XML schemas 494

G

getUserNameFromData element - form definition file 102
ghosted attribute - form view file 394
Glossary (section 1.1 14, section 1.1 14)
groove element - form definition file extension 197
grooveAdapter element - form definition file 150
group element - form definition file 103

H

header element - form definition file 127
HideInPrintView attribute - form view file 405
HoverSrc attribute - form view file 405
hwsAdapter element - form definition file 70
hwsOperation element - form definition file 71
hwsWorkflow element - form definition file 103
Hyperlink control - form view file 301
Hyperlink control - XML schema file 217
Hyperlink control example 438
Hyperlink input control - form view file 350
Hyperlink input control - XML schema file 225
Hyperlink input control example 462

I

ictID attribute - form view file 394
ictVersion attribute - form view file 394
Ignored controls - form view file 376
Implementer - security considerations 493
importerrors.xml - form template file component 33
importParameters element - form definition file 89
importSource element - form definition file 90
includedView element - form definition file extension 175
includedViews element - form definition file extension 174
InfoPath form template structure format 32
InfoPath XSF XSD schema 494
InfoPath XSF2 XSD schema 525
InfoPath XSF3 XSD schema 535
Informative references (section 1.2.2 18, section 1.2.2 18)
initialXmlDocument element - form definition file 107
inline attribute - form view file 394
innerCtrl attribute - form view file 395
input element - form definition file 72

inputscope attribute - form view file 395
inputScope element - form definition file extension 190
inputScopeId attribute - form view file 395
inputScopes element - form definition file extension 190
install element - form definition file extension 170
intro element - form definition file 82
Introduction (section 1 13, section 1 13)
Invalid constructs - form view file 377
Invalid controls - form view file 376
irm_template - form template file component 33

L

layoutText attribute - form view file 395
Linked picture control - form view file 360
Linked picture control - XML schema file 228
Linked picture control example 466
linkedToMaster attribute - form view file 396
List box control - form view file 303
List box control - XML schema file 217
List box control example 438
List controls - form view file 358
List controls - XML schema file 227
List controls example 464
list element - form definition file extension 196
listProperties element - form definition file 91
listPropertiesExtension element - form definition file extension 176
Localization (section 1.6 30, section 1.6 30)
location element - form definition file 104

M

mail element - form definition file extension 172
mainpane element - form definition file 124
managedCode element - form definition file extension 192
managedCodeType simple type - form definition file extension 163
manifest.xsf - form template file component 32
masterDetail element - form definition file 123
masterID attribute - form view file 396
masterName attribute - form view file 396
membership element - form definition file 101
menu element - form definition file 129
merge.xsl - form template file component 33
mergedPrintView element - form definition file extension 173
message element - form definition file 137
msxsl function extension - form view file 408
MSXSL node set() example 491
MSXSL Node Set() function 415
Multiple-selection list box control - form view file 363
Multiple-selection list box control - XML schema file 230
Multiple-selection list box control example 467
myFields element - submit file 414

N

Normative references (section 1.2.1 16, section 1.2.1 16)
num attribute - form view file 396

O

offline element - form definition file extension 175
offValue attribute - form view file 397
onLoad element - form definition file 98
onValue attribute - form view file 397
openNewDocumentAction element - form definition file 145
operation element - form definition file 70
Option button control - form view file 306
Option button control - XML schema file 218
Option button control example 440
override element - form definition file 58
Overview 18
Overview (synopsis) 18
  form definition (.xsf) file 19
  form template (.xsn) file 18
  form view files (XSLT) 19
  print view files (XSLT) 28
  resource files 30
  submit files (XML) 29
  template.XML file 29
  unused files 30
  upgrade.XSL file 29
  XML schema files (XSD) 19

P

package element - form definition file 108
partFragment element - form definition file 72
permissions element - form definition file 112
Picture button control - form view file 372
Picture button control - XML schema file 230
Picture button control example 473
postbackModel attribute - form view file 397
preview element - form definition file extension 189
primaryschema.xsd - form template file component 32
print element - form definition file 62
Print view file examples 486
Print view files (XSLT) 28
Print view files (XSLT) structure 414
printSettings element - form definition file 124
Product behavior 541
  overview 541
property element - form definition file 110

Q

query (secondary) element - form definition file 67
query element - form definition file 63
queryAction element - form definition file 145

R

ref attribute - form view file 399

References
informative (section 1.2.2 18, section 1.2.2 18)
normative (section 1.2.1 16, section 1.2.1 16)
overview 16
Relationship to protocols and other structures (section 1.4 30, section 1.4 30)
relativeQuery element - form definition file extension 182
Repeating section control - form view file 308
Repeating section control - XML schema file 218
Repeating section control example 442
Repeating table control - form view file 310
Repeating table control - XML schema file 219
Repeating table control example 442
Resource files 30
Resource files - form template file component 33
Rich text box control - form view file 313
Rich text box control - XML schema file 220
Rich text box control example 446
role element - form definition file 101
roles element - form definition file 99
rule element - form definition file 140
ruleSet element - form definition file 146
ruleSetAction element - form definition file 139
ruleSets element - form definition file 147

S

sampledata.xml - form template file component 32
save (disabled) element - form definition file 99
save element - form definition file 61
schemaErrorMessages element - form definition file 57
script element - form definition file 65
script.js - form template file component 33
script.vbs - form template file component 33
scripts element - form definition file 64
SearchPeopleOnly attribute - form view file 405
secondaryschema.xsd - form template file component 32
secondaryschema_offline.xml - form template file component 33
Section/optional section control - form view file 315
Section/optional section control - XML schema file 220
Section/optional section control example 446
Security 493
Security - implementer considerations 493
Security considerations
overview 493
sendByMail element - form definition file extension 185
sendMail element - form definition file 63
server attribute - form view file 406
server element - form definition file extension 165
serverCommandActionType simple type - form definition file extension 161
share element - form definition file extension 172
SharePoint file attachment control - form view file 375
SharePoint file attachment control - XML schema file 231
SharePoint file attachment control example 475
SharePointGroup attribute - form view file 406
sharepointListAdapter element - form definition file 74
sharepointListAdapterExtension element - form definition file extension 184
sharepointListAdapterRW element - form definition file 152
sharepointListAdapterRWExtension element - form definition file extension 197
SignatureBlock attribute - form view file 399
signatureLine element - form definition file extension 208
signatureLines element - form definition file extension 207
signedDataBlock element - form definition file 136
SignedSectionDisplaySignatures attribute - form view file 399
SignedSectionName attribute - form view file 400
signSignatureLineAction element - form definition file 158
Simple form template example 416
solutio9nPropertiesExtension2009 element - form definition file extension 204
solutionDefinition element - form definition file extension (section 2.2.2.2.1 164, section 2.2.3.2.1 201)
solutionMode element - form definition file extension 205
solutionProperties element - form definition file 59
solutionPropertiesExtension element - form definition file extension 168
solutionType simple type - form definition file extension 162
Structures
form view file - control-specific attributes 377
form view file - view representation 231
form view file - XSL function extensions 407
overview 32
XML schema file - control representation 212
subject element - form definition file 81
submit element - form definition file 94
submit element - form definition file extension 193
Submit file elements
dataFields 414
myFields 414
Submit file examples 488
Submit files (XML) 29
Submit files (XML) structure 414
submitAction element - form definition file (section 2.2.1.2.54 96, section 2.2.1.2.115 142)
submitAction element - form definition file extension 194
submitdata.xml - form template file component 33
submitToHostAdapter element - form definition file 83
successMessage element - form definition file 97
successMessage element - form definition file extension 194
suggestedSignerEmailAddress element - form definition file extension 210

suggestedSignerName element - form definition file extension 209
suggestedSignerTitle element - form definition file extension 210
switchViewAction element - form definition file 143

T

Table control - form view file 322
Table control - XML schema file 221
Table control example 449
task element - form definition file 106
taskpane element - form definition file 131
Template (XML)
  file 29
  structure 415
Template.(XML)
  example 489
Template.(XML) example 489
template.xml - form template file component 32
Text box control - form view file 323
Text box control - XML schema file 221
Text box control example 450
to element - form definition file 79
toolbar element - form definition file 128
toolbar element - form definition file extension 166
Tracking changes 542

U

unboundControls element - form definition file 121
Unused files 30
Upgrade (XSL)
  example 489
  file) 29
  functions
    MSXSL Node Set() 415
  MSXSL node set() example 491
  structure 415
Upgrade (XSL) example 489
upgrade.xsl - form template file component 33
useHttpHandler element - form definition file 97
useHttpHandlerExtension element - form definition file extension 178
useQueryAdapter element - form definition file 98
userName element - form definition file 102
useScriptHandler element - form definition file 98
useTransform element - form definition file 137

V

value attribute - form view file 400
Vendor-extensible fields (section 1.7 31, section 1.7 31)
Versioning (section 1.6 30, section 1.6 30)
view element - form definition file 132
View syntax - form view file 233
view.xsl - form template file component 32
viewExtension element - form definition file extension (section 2.2.2.2.35 187, section 2.2.3.2.9 206)
views element - form definition file 131
viewsExtension element - form definition file extension (section 2.2.2.2.34 187, section 2.2.3.2.8 206)

W

warning element - form definition file extension 186
warnings element - form definition file extension 186
webPartConnectionAction element - form definition file 157
webPartField element - form definition file extension 204
webPartFields element - form definition file extension 203
webPartProperties element - form definition file extension 203
webServiceAdapter element - form definition file 69
webServiceAdapterExtension element - form definition file extension 181
widgetIndex attribute - form view file 406
word element - form definition file extension 192
words element - form definition file extension 191
workflowInitAssoc element - form definition file extension 197
wss element - form definition file extension 170

X

xctname attribute - form view file 400
xdDate function extension - form view file 408
xdDesignMode simple type - form definition file 50
xdEmptyString simple type - form definition file 50
xdEnabledDisabled simple type - form definition file 46
xdEnvironment function extension - form view file 409
xdErrorMessage simple type - form definition file 50
xdExpressionLiteral simple type - form definition file 47
xdFileName simple type - form definition file 48
xdFormatting function extension - form view file 410
xdHWSCaption simple type - form definition file 53
xdHWSname simple type - form definition file 52
xdImage function extension - form view file (section 2.4.3.5 410, section 2.4.3.11 413)
xdLineStamp simple type - form definition file extension 201
xdManualAuto simple type - form definition file 47
xdMath function extension - form view file 410
xdModeType simple type - form definition file extension 200
xDocumentClass element - form definition file 54
xdParameterType simple type - form definition file extension 199
xdRoleName simple type - form definition file 42
xdScriptLanguage simple type - form definition file 49
xdServerInfo function extension - form view file 413
xdSignatureRelationEnum simple type - form definition file 52

xdSignedDataBlockMessage simple type - form definition file 52
xdSignedDataBlockName simple type - form definition file 51
xdSignSignatureLineRuleEnum simple type - form definition file 53
xdSolutionVersion simple type - form definition file 49
xdTitle simple type - form definition file 40
xdUser function extension - form view file 411
xdUtil function extension - form view file 412
xdViewName simple type - form definition file 42
xdXDocument function extension - form view file 412
xdYesNo simple type - form definition file 43
XML schema file - control representation 212
XML schema file controls
button 214
check box 214
choice group/choice section 222
combo box 223
contact selector 215
date picker 215
date/time picker 224
drop-down list 216
embedded picture 225
entity picker 224
expression box 216
file attachment 216
hyperlink 217
hyperlink input 225
linked picture 228
list 227
list box 217
multiple-selection list box 230
option button 218
picture button 230
repeating section 218
repeating table 219
rich text box 220
section/optional section 220
SharePoint file attachment 231
table 221
text box 221
XML schema file example 427
XML schema files (XSD) 19
XML schema files (XSD) structure 212
XML schemas 494
xmlFileAdapter element - form definition file 73
xmlFileAdapterExtension element - form definition file extension 183
xmlToEdit attribute - form view file 402
xmlToEdit element - form definition file 134
xmlToEditExtension element - form definition file extension (section 2.2.2.2.36 188, section 2.2.3.2.10 207)
XSF Elements - form definition file 54
XSF Enumerations - form definition file 40
XSF2 Elements - form definition file 163
XSF2 Enumerations - form definition file 161
XSF3 Elements - form definition file 201
XSF3 Enumerations - form definition file 199
XSL function extensions example 482
XSL root template - form view file 239
XSL root template style sheets - form view file 241