OpenDoc

OpenDoc Class Reference

By
The OpenDoc Design Team

Version 1.0
February 21, 1994

TABLE OF CONTENTS

Table of Contents iii
Class Descriptions 22
XMPArbitrator 24
Basic Class Documentation 24
Theory of Operation 24
Invariants Maintained by Class 24
Other Persistent Properties 24
Member Functions 24
CreateOwnerIterator 24
GetFocusOwner 25
IsFocusExclusive 25
IsFocusRegistered 26
Purge 26
RegisterFocus 27
RelinquishFocus 27
RelinquishFocusSet 28
RequestFocus 28
RequestFocusSet 29
TransferFocus 29
TransferFocusSet 30
UnregisterFocus 31
XMPCanvas 32
Basic Class Documentation 32
Theory of Operation 32
Invariants Maintained by Class 32
Other Persistent Properties 32
Member Functions 33
XMPCanvas 33
~XMPCanvas 33
GetFacet 33
GetGraphicsSystem 34
GetOwner 34
GetPlatformCanvas 35
GetUpdateShape 35
InitCanvas 36
Invalidate 36
IsDynamic 37
IsOffscreen 37
Purge 38
ResetUpdateShape 38
SetFacet 39
SetOwner 39
Validate 40
XMPClipboard 41
Basic Class Documentation 41
Theory of Operation 41
Invariants Maintained by Class 41
Other Persistent Properties 41
Member Functions 41
Clear 41
GetChangeID 42

GetContentStorageUnit 42
Lock 43
SetPlatformClipboard 44
Unlock 45
XMPContainer 46
Basic Class Documentation 46
Theory of Operation 46
Invariants Maintained by Class 46
Other Persistent Properties 46
Member Functions 46
XMPContainer 46
~XMPContainer 47
Close 47
Create 48
GetDocument 48
GetID 49
GetName 49
GetStorageSystem 50
IncrementRefCount 50
InitContainer 51
Open 51
Purge 52
Release 52
ReleaseDocument 53
SetName 53
XMPDispatcher 55
Basic Class Documentation 55
Theory of Operation 55
Invariants Maintained by Class 55
Other Persistent Properties 55
Member Functions 55
AddDispatchModule 55
AddMonitor 56
Dispatch 56
GetMouseRegion 57
InvalidateFacetUnderMouse 57
Purge 58
RegisterIdle 58
RemoveDispatchModule 59
RemoveMonitor 59
SetMouseRegion 60
ShouldTerminateEventLoop 60
TerminateEventLoop 61
UnregisterIdle 61
Yield 62
XMPDispatchModule 63
Basic Class Documentation 63
Theory of Operation 63
Invariants Maintained by Class 63
Other Persistent Properties 63
Member Functions 63
XMPDispatchModule 63
~XMPDispatchModule 63
XMPDocument 65

Basic Class Documentation 65
Theory of Operation 65
Invariants Maintained by Class 65
Other Persistent Properties 65
Member Functions 65
XMPDocument 65
~XMPDocument 66
CollapseDrafts 66
CreateDraft 67
GetBaseDraft 68
GetContainer 68
GetDocumentProperties 69
GetDraft 69
GetID 70
GetName 71
IncrementRefCount 71
InitDocument 72
Purge 72
Release 73
ReleaseDraft 73
SaveToAPrevDraft 74
SetBaseDraftFromForeignDraft 75
SetName 75
XMPDraft 77
Basic Class Documentation 77
Theory of Operation 77
Invariants Maintained by Class 77
Other Persistent Properties 78
Member Functions 78
XMPDraft 78
~XMPDraft 78
AbortClone 79
BeginClone 79
ChangedFromPrev 80
CreateFrame 80
CreateLinkSource 81
CreateLinkSpec 81
CreatePart 82
CreateStorageUnit 83
EndClone 83
Externalize 83
GetDocument 84
GetDraftProperties 84
GetFrame 85
GetID 85
GetLink 86
GetLinkSource 86
GetName 87
GetPart 87
GetPermissions 88
GetStorageUnit 88
IncrementRefCount 89
InitDraft 89
Purge 90

Release 90
ReleaseFrame 91
ReleaseLink 91
ReleasePart 92
ReleaseStorageUnit 92
RemoveChanges 93
RemoveFrame 93
RemoveFromDocument 94
RemoveLink 94
RemoveLinkSource 95
RemovePart 95
RemoveStorageUnit 96
SaveToAPrevious 96
SetChangedFromPrev 97
SetName 97
XMPDragAndDrop 99
Basic Class Documentation 99
Theory of Operation 99
Invariants Maintained by Class 99
Other Persistent Properties 99
Member Functions 99
XMPDragAndDrop 99
~XMPDragAndDrop 100
Clear 100
GetDragAttributes 101
GetDragReference 101
GetStorageUnit 102
InitDragAndDrop 102
Purge 103
StartDrag 103
XMPDragItemIterator 105
Basic Class Documentation 105
Theory of Operation 105
Invariants Maintained by Class 105
Other Persistent Properties 105
Member Functions 105
First 105
IsNotComplete 105
Next 106
XMPEmbeddedFramesIterator 107
Basic Class Documentation 107
Theory of Operation 107
Invariants Maintained by Class 107
Other Persistent Properties 107
Member Functions 107
XMPExtension 108
Basic Class Documentation 108
Theory of Operation 108
Invariants Maintained by Class 108
Other Persistent Properties 108
Member Functions 108
XMPExtension 108
~XMPExtension 108
GetBase 109

InitExtension 109
Release 110
XMPFacet 111
Basic Class Documentation 111
Theory of Operation 112
Invariants Maintained by Class 112
Other Persistent Properties 112
Member Functions 112
XMPFacet 112
~XMPFacet 112
ActiveBorderContainsPoint 113
ChangeActiveShape 113
ChangeClipShape 114
ChangeExternalTransform 114
ChangeHighlight 115
ContainsPoint 115
CreateEmbeddedFacet 116
CreateFacetIterator 116
Draw 117
DrawActiveBorder 118
DrawChildren 118
DrawChildrenAlways 118
DrawnIn 119
GetActiveShape 119
GetAggregateClipShape 120
GetCanvas 120
GetClipShape 121
GetContainingFacet 121
GetContentTransform 122
GetExternalTransform 122
GetFrame 122
GetFrameTransform 123
GetHighlight 123
GetPartInfo 124
GetWindow 124
GetWindowAggregateClipShape 125
GetWindowContentTransform 125
GetWindowFrameTransform 126
HasCanvas 126
InitFacet 127
Invalidate 127
InvalidateActiveBorder 128
IsSelected 128
MoveBefore 129
MoveBehind 129
RemoveFacet 130
SetCanvas 130
SetPartInfo 131
SetSelected 131
SetWindow 132
Update 132
Validate 132
XMPFacetIterator 134
Basic Class Documentation 134

Theory of Operation 134
Invariants Maintained by Class 134
Other Persistent Properties 134
Member Functions 134
XMPFacetIterator 134
~XMPFacetIterator 135
First 135
IsNotComplete 136
Next 136
SkipChildren 136
XMPFocusModule 138
Basic Class Documentation 138
Theory of Operation 138
Invariants Maintained by Class 138
Other Persistent Properties 138
Member Functions 138
XMPFocusModule 138
~XMPFocusModule 138
InitFocusModule 139
IsFocusExclusive 139
XMPFocusOwnerIterator 141
Basic Class Documentation 141
Theory of Operation 141
Invariants Maintained by Class 141
Other Persistent Properties 141
Member Functions 141
First 141
IsNotComplete 141
Next 142
XMPFocusSet 143
Basic Class Documentation 143
Theory of Operation 143
Invariants Maintained by Class 143
Other Persistent Properties 143
Member Functions 143
XMPFocusSet 143
~XMPFocusSet 143
Add 144
Contains 144
CreateIterator 145
InitFocusSet 145
Remove 146
XMPFocusSetIterator 147
Basic Class Documentation 147
Theory of Operation 147
Invariants Maintained by Class 147
Other Persistent Properties 147
Member Functions 147
~XMPFocusSetIterator 147
First 147
IsNotComplete 148
Next 148
XMPFrame 150
Basic Class Documentation 150

Theory of Operation 151
Invariants Maintained by Class 151
Other Persistent Properties 151
Member Functions 151
ChangeFrameShape 151
ChangeInternalTransform 152
ChangeLinkStatus 152
ChangePart 153
ChangePresentation 153
ChangeUsedShape 154
ChangeViewType 154
CloneTo 155
Close 155
ContentChanged 156
CreateFacetIterator 156
DoesPropagateEvents 157
DrawActiveBorder 157
Externalize 158
FacetAdded 158
FacetRemoved 159
GetContainingFrame 159
GetFrameGroup 160
GetFrameShape 160
GetInternalTransform 160
GetLinkStatus 161
GetPart 161
GetPartInfo 162
GetPresentation 162
GetUsedShape 163
GetViewType 163
Invalidate 164
InvalidateActiveBorder 164
IsDragging 165
IsDroppable 165
IsFrozen 165
IsOverlaid 166
IsRoot 166
IsSubframe 167
Purge 167
Release 168
Remove 168
RequestFrameShape 169
SetContainingFrame 169
SetDragging 170
SetDroppable 170
SetFrameGroup 170
SetFrozen 171
SetPartInfo 171
SetPresentation 172
SetPropagateEvents 172
SetSubframe 173
SetViewType 173
Validate 174
XMPFrameFacetIterator 175

Basic Class Documentation 175
Theory of Operation 175
Invariants Maintained by Class 175
Other Persistent Properties 175
Member Functions 175
XMPFrameFacetIterator 175
~XMPFrameFacetIterator 175
First 176
IsNotComplete 176
Next 177
XMPLink 178
Basic Class Documentation 178
Theory of Operation 178
Invariants Maintained by Class 178
Other Persistent Properties 178
Member Functions 178
Externalize 178
GetChangeID 179
GetContentStorageUnit 179
GetStatus 180
Lock 180
RegisterDependent 181
ShowSourceContent 182
Unlock 182
UnregisterDependent 183
XMPLinkManager 184
Basic Class Documentation 184
Theory of Operation 184
Invariants Maintained by Class 184
Other Persistent Properties 184
Member Functions 185
AnyLinkImported 185
DraftClosing 185
DraftOpened 186
DraftSaved 186
NewSectionID 187
ReserveSectionID 187
UnsavedExportedLinks 188
XMPLinkSource 189
Basic Class Documentation 189
Theory of Operation 189
Invariants Maintained by Class 190
Other Persistent Properties 190
Member Functions 190
Clear 190
ContentChanged 191
Externalize 191
GetChangeID 192
GetContentStorageUnit 192
GetLink 193
IsAutoExport 193
Lock 194
SetAutoExport 194
ShowSourceContent 195

Unlock 195
XMPLinkSpec 197
Basic Class Documentation 197
Theory of Operation 197
Invariants Maintained by Class 197
Other Persistent Properties 197
Member Functions 198
XMPLinkSpec 198
~XMPLinkSpec 198
ReadFromProperty 198
WriteToProperty 199
XMPMenuBar 200
Basic Class Documentation 200
Theory of Operation 200
Invariants Maintained by Class 200
Other Persistent Properties 200
Member Functions 200
XMPMenuBar 200
~XMPMenuBar 201
AddMenuBefore 201
AddMenuLast 202
AddSubMenu 202
Copy 203
Display 203
GetCommand 204
GetMenu 204
GetMenuAndItem 205
InitMenuBar 205
IsCommandRegistered 206
IsCommandSynthetic 206
Purge 207
RegisterCommand 207
Release 208
RemoveMenu 208
UnregisterAll 209
UnregisterCommand 209
XMPMessageInterface 210
Basic Class Documentation 210
Theory of Operation 210
Invariants Maintained by Class 210
Other Persistent Properties 210
Member Functions 210
CreateEvent 210
CreatePartAddrDesc 211
CreatePartObjSpec 211
ProcessSemanticEvent 212
Purge 212
Send 213
XMPNameResolver 214
Basic Class Documentation 214
Theory of Operation 214
Invariants Maintained by Class 214
Other Persistent Properties 214
Member Functions 214

CallObjectAccessor 214
CreateSwapToken 215
CreateToken 215
DisposeToken 216
GetContainingFrame 216
GetContextInfo 217
Purge 217
Resolve 218
XMPNameSpace 219
Basic Class Documentation 219
Theory of Operation 219
Invariants Maintained by Class 219
Other Persistent Properties 219
Member Functions 219
Exists 219
GetName 219
GetValue 220
Purge 220
Register 221
Unregister 221
XMPNameSpaceManager 223
Basic Class Documentation 223
Theory of Operation 223
Invariants Maintained by Class 223
Other Persistent Properties 223
Member Functions 223
CreateNameSpace 223
DeleteNameSpace 224
HasNameSpace 224
Purge 224
XMPObject 226
Basic Class Documentation 226
Theory of Operation 226
Invariants Maintained by Class 226
Other Persistent Properties 226
Member Functions 226
XMPObject 226
~XMPObject 226
GetExtension 227
HasExtension 227
InitObject 228
Purge 228
ReleaseExtension 229
XMPPart 230
Basic Class Documentation 230
Theory of Operation 230
Invariants Maintained by Class 230
Other Persistent Properties 230
Member Functions 230
AttachSourceFrame 230
ChangeKind 231
CloneInto 232
ContainingPartPropertiesChanged 232
CreateEmbeddedFrame 233

CreateEmbeddedFramesIterator 234
CreateLink 234
EmbeddedFrameSpec 235
FocusAcquired 236
FocusLost 236
GetContainingPartProperties 237
GetPrintResolution 237
Open 238
RemoveEmbeddedFrame 238
RevealFrame 239
XMPPersistentObject 240
Basic Class Documentation 240
Theory of Operation 240
Invariants Maintained by Class 240
Other Persistent Properties 240
Member Functions 240
XMPPersistentObject 240
~XMPPersistentObject 241
Externalize 241
GetID 242
GetStorageUnit 242
InitPersistentObject 243
InitPersistentObjectFromStorage 243
XMPPlatformTypeSet 245
Basic Class Documentation 245
Theory of Operation 245
Invariants Maintained by Class 245
Other Persistent Properties 245
Member Functions 245
XMPPlatformTypeSet 245
~XMPPlatformTypeSet 246
Add 246
Contains 246
Count 247
CreatePlatformTypeSetIterator 247
Difference 248
Remove 248
Union 249
XMPPlatformTypeSetIterator 250
Basic Class Documentation 250
Theory of Operation 250
Invariants Maintained by Class 250
Other Persistent Properties 250
Member Functions 250
~XMPPlatformTypeSetIterator 250
First 250
IsNotComplete 251
Next 251
XMPRefCntObject 253
Basic Class Documentation 253
Theory of Operation 253
Invariants Maintained by Class 253
Other Persistent Properties 253
Member Functions 253

XMPRefCntObject 253
~XMPRefCntObject 254
IncrementRefCount 254
InitRefCntObject 254
Release 255
XMPSemanticInterface 256
Basic Class Documentation 256
Theory of Operation 256
Invariants Maintained by Class 256
Other Persistent Properties 257
Member Functions 257
XMPSemanticInterface 257
GetCoercionHandler 257
GetEventHandler 258
GetObjectAccessor 258
GetOSLSupportFlags 259
GetSpecialHandler 260
InitSemanticInterface 260
InstallAdjustMarksProc 261
InstallCoercionHandler 261
InstallCompareProc 262
InstallCountProc 262
InstallDisposeTokenProc 263
InstallEventHandler 263
InstallGetErrDescProc 264
InstallGetMarkTokenProc 264
InstallMarkProc 265
InstallObjectAccessor 265
InstallSpecialHandler 266
RemoveCoercionHandler 267
RemoveEventHandler 267
RemoveObjectAccessor 268
RemoveSpecialHandler 268
SetOSLSupportFlags 269
~XMPMacSemanticInterface 269
XMPSession 271
Basic Class Documentation 271
Theory of Operation 271
Invariants Maintained by Class 271
Other Persistent Properties 271
Member Functions 271
Close 271
GetArbitrator 271
GetClipboard 272
GetDispatcher 272
GetDragAndDrop 273
GetLinkManager 273
GetMessageInterface 274
GetNameResolver 274
GetNameSpaceManager 274
GetSemanticInterface 275
GetShellSemtInterface 275
GetStorageSystem 276
GetTranslation 276

GetType 277
GetUndo 277
GetWindowState 278
OpenXMPSession 278
Purge 279
RemoveEntry 279
Tokenize 280
UniqueChangeID 280
XMPShape 281
Basic Class Documentation 281
Theory of Operation 281
Invariants Maintained by Class 281
Other Persistent Properties 281
Member Functions 281
XMPShape 281
~XMPShape 282
ContainsPoint 282
Copy 283
CopyFrom 283
CopyPolygon 284
GetBoundingBox 284
GetGeometryMode 285
GetGXShape 285
GetPlatformShape 286
GetQDRegion 286
HasGeometry 287
Intersect 287
InverseTransform 288
IsEmpty 288
IsRectangular 289
IsSameAs 289
ReadShape 290
SetGeometryMode 290
SetGXShape 291
SetPlatformShape 291
SetPolygon 292
SetQDRegion 292
SetRectangle 293
Subtract 293
Transform 294
Union 294
WriteShape 295
XMPStorageSystem 296
Basic Class Documentation 296
Theory of Operation 296
Invariants Maintained by Class 296
Other Persistent Properties 296
Member Functions 296
GetSession 296
NeedSpace 297
XMPStorageUnit 298
Basic Class Documentation 298
Theory of Operation 298
Invariants Maintained by Class 298

Other Persistent Properties 298
Member Functions 299
XMPStorageUnit 299
~XMPStorageUnit 299
AddProperty 300
AddValue 300
ClearAllPromises 301
CloneInto 301
CloneTo 302
CopyTo 302
CountProperties 303
CountValues 303
CreateCursor 304
CreateView 304
DeleteValue 305
Exists 305
Exists 306
Externalize 307
Focus 307
Focus 308
GetDraft 308
GetGenerationNumber 309
GetID 309
GetIDFromStorageUnitRef 310
GetName 310
GetOffset 311
GetPromiseValue 311
GetProperty 312
GetSession 312
GetSize 313
GetStorageUnitRefIterator 313
GetStrongStorageUnitRef 314
GetType 314
GetValue 315
GetWeakStorageUnitRef 315
IncrementGenerationNumber 316
IncrementRefCount 316
InitStorageUnit 317
InsertValue 317
Internalize 318
IsPromiseValue 318
IsStrongStorageUnitRef 319
IsWeakStorageUnitRef 319
Lock 320
Purge 320
Release 321
Remove 321
RemoveStorageUnitRef 322
SetName 322
SetOffset 323
SetPromiseValue 323
SetType 324
SetValue 324
XMPStorageUnitCursor 326

Basic Class Documentation 326
Theory of Operation 326
Invariants Maintained by Class 326
Other Persistent Properties 326
Member Functions 326
XMPStorageUnitCursor 326
~XMPStorageUnitCursor 326
GetCursor 327
InitStorageUnitCursor 327
XMPStorageUnitRefIterator 329
Basic Class Documentation 329
Theory of Operation 329
Invariants Maintained by Class 329
Other Persistent Properties 329
Member Functions 329
XMPStorageUnitRefIterator 329
~XMPStorageUnitRefIterator 329
First 330
InitStorageUnitRefIterator 330
IsNotComplete 331
Next 331
XMPStorageUnitView 333
Basic Class Documentation 333
Theory of Operation 333
Invariants Maintained by Class 333
Other Persistent Properties 333
Member Functions 333
XMPStorageUnitView 333
~XMPStorageUnitView 334
AddProperty 334
AddValue 335
CloneInto 335
CloneTo 336
CopyTo 336
DeleteValue 337
Externalize 338
GetCursor 338
GetGenerationNumber 338
GetID 339
GetIDFromStorageUnitRef 339
GetName 340
GetOffset 340
GetPromiseValue 341
GetProperty 342
GetSize 342
GetStorageUnit 342
GetStorageUnitRefIterator 343
GetStrongStorageUnitRef 343
GetType 344
GetValue 345
GetWeakStorageUnitRef 345
IncrementGenerationNumber 346
InitStorageUnitView 346
InsertValue 347

Internalize 347
IsPromiseValue 348
IsStrongStorageUnitRef 348
IsWeakStorageUnitRef 349
Purge 350
Remove 350
RemoveStorageUnitRef 351
SetName 351
SetOffset 351
SetPromiseValue 352
SetType 353
SetValue 353
XMPTransform 355
Basic Class Documentation 355
Theory of Operation 355
Invariants Maintained by Class 355
Other Persistent Properties 355
Member Functions 355
XMPTranslation 356
Basic Class Documentation 356
Theory of Operation 356
Invariants Maintained by Class 356
Other Persistent Properties 356
Member Functions 356
XMPTranslation 356
~XMPTranslation 357
GetISOTypeFromPlatformType 357
GetPlatformTypeFromISOType 358
GetTranslateMethod 358
GetTranslationOf 359
InitTranslation 359
Purge 360
Translate 360
Translate 361
XMPTypeSet 363
Basic Class Documentation 363
Theory of Operation 363
Invariants Maintained by Class 363
Other Persistent Properties 363
Member Functions 363
XMPTypeSet 363
~XMPTypeSet 363
Add 364
Contains 364
Count 365
CreateTypeSetIterator 365
Remove 366
XMPTypeSetIterator 367
Basic Class Documentation 367
Theory of Operation 367
Invariants Maintained by Class 367
Other Persistent Properties 367
Member Functions 367
~XMPTypeSetIterator 367

First 367
IsNotComplete 368
Next 368
XMPUndo 370
Basic Class Documentation 370
Theory of Operation 370
Invariants Maintained by Class 370
Other Persistent Properties 370
Member Functions 370
AddActionToHistory 370
ClearActionHistory 371
ClearRedoHistory 372
MarkActionHistory 372
PeekRedoHistory 373
PeekUndoHistory 373
Purge 374
Redo 374
Undo 375
XMPValueIterator 376
Basic Class Documentation 376
Theory of Operation 376
Invariants Maintained by Class 376
Other Persistent Properties 376
Member Functions 376
XMPValueIterator 376
~XMPValueIterator 376
First 377
IsNotComplete 377
Next 378
XMPWindow 379
Basic Class Documentation 379
Theory of Operation 379
Invariants Maintained by Class 379
Other Persistent Properties 379
Member Functions 379
Close 379
CloseAndRemove 380
GetFacetUnderPoint 380
GetID 381
GetPlatformWindow 381
GetRootFrame 382
GetSourceFrame 382
Hide 383
IsActive 383
IsResizable 383
IsRootWindow 384
IsShown 384
Open 385
Purge 385
Resized 386
Select 386
SetShouldSave 387
SetShouldShowLinks 387
ShouldSave 388

ShouldShowLinks 388
Show 389
XMPWindowIterator 390
Basic Class Documentation 390
Theory of Operation 390
Invariants Maintained by Class 390
Other Persistent Properties 390
Member Functions 390
~XMPWindowIterator 390
First 390
IsNotComplete 391
Last 391
Next 392
Previous 392
XMPWindowState 394
Basic Class Documentation 394
Theory of Operation 394
Invariants Maintained by Class 394
Other Persistent Properties 394
Member Functions 394
ActivateFrontWindows 394
AdjustPartMenus 395
CloseWindows 395
CopyBaseMenuBar 396
CreateWindow 396
CreateWindowIterator 397
DeactivateFrontWindows 397
Externalize 398
GetActiveWindow 398
GetFrontFloatingWindow 399
GetFrontRootWindow 399
GetFrontWindow 400
GetRootWindowCount 400
GetTotalRootWindowCount 401
GetWindow 401
GetWindowCount 402
GetXMPWindow 402
Internalize 403
IsXMPWindow 403
OpenWindows 403
Purge 404
SetBaseMenuBar 404
SetDefaultWindowTitles 405
Global Functions 407
Basic Documentation 407
Functions 407
OpenXMPSession 407
ShowPartFrameInfo 407
Constants 409
XMPConstants 411
General 411
Name Spaces 411
Category Constants 411
Parts 412

UI Subsystem - Foci 412
UI Subsystem - Menu IDs 412
UI Subsystem - Standard Mac Events 413
UI Subsystem - Additional OpenDoc Events 413
UI Subsystem - Part Codes 414
UI Subsystem - Command IDs 414
UI Subsystem-Persistent Properties 416
UI Subsystem - Persistent Value Types 418
UI subsytem - Undo 418
Messaging subsystem 418
Messaging - Properties for Part Info Dialog 418
Data Interchange - Constants 419
Storage System 420
Data Interchange - Persistent Properties 421
Data Interchange - Persistent Value Types 422
Types 423
XMPTypes 425
General 425
Data Interchange 426
Storage System 427
Imaging 427
UI Subsystem 430
Messaging 431

CLASS DESCRIPTIONS

XMPArbitrator

Basic Class Documentation

The Arbitrator is used to manage competition between parts for named resources known as "foci".
The parts in an OpenDoc document communicate with a single XMPArbitrator object, obtained from the XMPSession object.
XMPArbitrator is a subclass of XMPObject.
XMPArbitrator is a platform-independent class.
Related classes are XMPFocusModule, XMPFocusSet, and XMPFocusSetIterator.
XMPArbitrator participates in the Part Activation and UI Events protocols.

Theory of Operation

The arbitrator is used to manage competition between parts for named resources known as "foci". The foci are described by segmented ISO strings, but most arbitration methods used tokenize forms of these. The Session object has a Tokenize() method.
Most foci are exclusive, and can only be "owned" by a single part at a time. However the arbitrator also supports non exclusive foci which can be owned by several parts. In this case, all the arbitrator does is provide a central place to record and obtain the list of owners of a focus.
Example foci include keystroke focus, selection focus, and menu bar focus. Keyboard events are sent to the part which has the keystroke focus. Menu events are sent to the part which has the menu bar focus (assuming there is a menu bar on a particular platform).
Many foci are related to event distribution within an OpenDoc document, but foci could also be associated with system-wide resources like ports.
A part typically requests and relinquishes a set of foci from the arbitrator. What the user thinks of as the "active" part is really the part which owns the selection focus. Don't worry. The same part will almost always also own the menu bar and keystroke foci, but it is possible, in principle, for one part to receive menu commands while another gets keystrokes. This may make sense in a dialog part which is built from control parts with which it has a more intimate relationship than the usual one between containers and embedded parts.
The arbitrator is extensible, to accomodate the addition of exotic input devices, and parts which use them. The arbitrator contains a collection of focus modules (instances of subclasses of XMPFocusModule). Additional focus modules can be installed to handle new focus types.

Invariants Maintained by Class

XMPArbitrator maintains a table associating foci with focus modules. There is a standard focus module which arbitrates the standard foci like keystroke focus and menu bar focus.

Other Persistent Properties

No persistent properties

Member Functions

CreateOwnerIterator

Function Prototype

```
XMPFocusOwnerIterator* CreateOwnerIterator(
        XMPTypeToken focus);
```

Protocol

Focus Ownership

Protection

Public. Can be called by part editors.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns an iterator which returns all the parts which own the specified non-exclusive focus.

Inputs

focus	A token for a registered focus type.

Outputs

<return>	An iterator

Exceptions Signalled

kXMPErrOutOfMemory	Out of memory

Pre conditions

A valid non-exclusive focus.

Post conditions

No change in internal state.

GetFocusOwner

Function Prototype

```
XMPFrame* GetFocusOwner(
        XMPTypeToken focusType);
```

Protocol

Focus Ownership

Protection

Public. Called by part editors.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the frame which owns the specified focus, or kXMPNULL.

Inputs

focusType	A token for a registered focus type.

Outputs

<return>	The frame which owns the focus. May be kXMPNull.

Exceptions Signalled

None.

Pre conditions

A valid focus.

Post conditions

The owning frame is returned.

IsFocusExclusive

Function Prototype

```
XMPBoolean IsFocusExclusive(
        XMPTypeToken focus);
```

Protocol

Part Activation, UI Events

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns kXMPTrue, if the specified focus is exclusive, kXMPFalse otherwise.

Inputs

focus	A token for a registered focus

Outputs

<return>	kXMPTrue, if the focus is exclusive

Exceptions Signalled

None.

Pre conditions

A constructed and initialized object of this class.

Post conditions

Result contains kXMPTrue if the specified focus is exclusive, kXMPFalse otherwise.

IsFocusRegistered

Function Prototype

```
XMPBoolean IsFocusRegistered(
        XMPTypeToken focus);
```

Protocol

Focus Registration

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns kXMPTrue, if the specified focus is registered, kXMPFalse otherwise.

Inputs

focus	A token for a registered focus

Outputs

<return>	kXMPTrue, if the focus is registered

Exceptions Signalled

None.

Pre conditions

A constructed and initialized object of this class.

Post conditions

Result contains kXMPTrue if the specified focus is exclusive, kXMPFalse otherwise.

Purge

This method is only of interest to Container Application developers.

Function Prototype

```
XMPSize Purge(
        XMPSize size);
```

Protocol

Low memory

Protection

Public. Called by OpenDoc in low memory situations.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Frees up some memory, usually by flushing internal state to external storage.

Inputs

size	The amount of memory requested.

Outputs

<return>	The amount of memory freed.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance

Post conditions

Some memory has been freed up.

RegisterFocus

Function Prototype

```
void RegisterFocus(
        XMPTypeToken focus,
        XMPFocusModule* focusModule);
```

Protocol

Focus Registration

Protection

Public. Called by some part editors.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Installs a new focus module to be used to manage the specified focus.

Inputs

focus	A token for a focus to be managed by the specified focus module.
focusModule	A focus module to be associated with the specified focus.

Outputs

None.

Exceptions Signalled

kXMPErrExistingFocusModule	A focus module already exists for that focus.

Pre conditions

A valid focus and focus module

Post conditions

The table contains the new focus module.

RelinquishFocus

Function Prototype

```
void RelinquishFocus(
        XMPTypeToken focus,
        XMPFrame* relinquishingFrame);
```

Protocol

Change of Ownership

Protection

Public. Called by part editors.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Sets the owner frame of the specified focus to kXMPNULL.

Inputs

focus	A focus to be relinquished.
relinquishingFrame	The frame which is relinquishing ownership of the focus.

Outputs

None.

Exceptions Signalled

kXMPErrInvalidFrame

Pre conditions

A valid focus and relinquishing frame are required.

Post conditions

The focus now has no owner.

RelinquishFocusSet

Function Prototype

```
void RelinquishFocusSet(
        XMPFocusSet* focusSet,
        XMPFrame* relinquishingFrame);
```

Protocol

Change of Ownership

Protection

Public. Called by part editors.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Sets the owner frame of each focus in the specified focus set to kXMPNULL.

Inputs

focusSet	A set of foci to be relinquished.
relinquishingFrame	The frame which is relinquishing ownership of the set of foci.

Outputs

None.

Exceptions Signalled

kXMPErrInvalidFocusSet
kXMPErrInvalidFrame

Pre conditions

A valid focus set and relinquishing frame are required.

Post conditions

Each focus now in the set now has no owner.

RequestFocus

Function Prototype

```
XMPBoolean RequestFocus(
        XMPTypeToken focus,
        XMPFrame* requestingFrame);
```

Protocol

Change of Ownership

Protection

Public. Called by part editors, which generally use RequestFocusSet, but use this method when a single focus is requested.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Ownership of the specified focus is assigned to the specified frame, provided the existing owner was willing to relinquish the requested focus.

Inputs

focus	A requested focus.
requestingFrame	The frame requesting the focus.

Outputs

<return>	kXMPTrue, if the request was granted.

Exceptions Signalled

kXMPErrFocusNotRegistered One of the requested foci is not registered.

Pre conditions

A valid focus, and requesting frame.

Post conditions

If the request is granted, the new ownership relationship is stored in the relevant focus module, and kXMPTrue is returned.
If the request fails, the existing ownership relationship is intact, and kXMPFalse is returned.

RequestFocusSet

Function Prototype

```
XMPBoolean RequestFocusSet(
        XMPFocusSet* focusSet,
        XMPFrame* requestingFrame);
```

Protocol

Change of Ownership

Protection

Public. Called by part editors.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Ownership of each focus in the specified focus set is assigned to the specified frame, provided all existing owners were willing to relinquish the requested foci. The operation is atomic. If one focus is unattainable ownership of the set is not granted.

Inputs

focusSet	A set of requested foci.
requestingFrame	The frame requesting the focus.

Outputs

<return>	kXMPTrue, if the request was granted.

Exceptions Signalled

kXMPErrFocusNotRegistered One of the requested foci is not registered.

Pre conditions

A valid focus set, and requesting frame.

Post conditions

If the request is granted, the new ownership relationships are stored in the relevant focus modules, and kXMPTrue is returned.
If the request fails, the existing ownership relationships are intact, and kXMPFalse is returned.

TransferFocus

Function Prototype

```
void TransferFocus(
        XMPTypeToken focus,
        XMPFrame* transferringFrame,
        XMPFrame* newOwner);
```

Protocol

Change of Ownership

Protection

Public. Called by part editors, for example when transferring the modal focus back to its previous owner.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Transfers a focus from its current owner to another. The new owner's FocusAcquired() method is called. If the existing owner is not the transferring frame, its FocusLost() method is also called.

Inputs

focus	A focus to be relinquished.
transferringFrame	The frame which is transferring ownership of the focus. Need not be the current owner.
newOwner	The new focus owner

Outputs

None.

Exceptions Signalled

kXMPErrInvalidFrame

Pre conditions

The focus is registered.

Post conditions

The focus is owned by "newOwner".

TransferFocusSet

Function Prototype

```
void TransferFocusSet(
        XMPFocusSet* focusSet,
        XMPFrame* transferringFrame,
        XMPFrame* newOwner);
```

Protocol

Change of Ownership

Protection

Public. Can be called by part editors.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Sets the owner frame of each focus in the specified focus set to newFrame. The new owner's FocusAcquired() method is called. If the existing owner is not the transferring frame, its FocusLost() method is also called.

Inputs

focusSet	A set of foci to be relinquished.
transferringFrame	The frame which is transferring ownership of the set of foci.
newOwner	The new frame which owns the focus set

Outputs

None.

Exceptions Signalled

kXMPErrInvalidFocusSet
kXMPErrInvalidFrame

Pre conditions

The foci are registered

Post conditions

The foci are owned by "newFrame"

UnregisterFocus

Function Prototype

```
void UnregisterFocus(
        XMPTypeToken focus);
```

Protocol

Focus Registration

Protection

Public. Called by some part editors.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Removes the specified focus module from the table.

Inputs

focus	A token for a focus to be removed.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

A valid focus and focus module.

Post conditions

The specified focus module is not in the table.

XMPCanvas

Basic Class Documentation

Ancestors: XMPCanvas -> XMPObject

XMPCanvas objects are wrappers for platform-specific data structures representing drawing environments.

OpenDoc makes few assumptions about the underlying drawing environment of a canvas. A canvas can hold anything from a bitmap or a structured display list to a stream of PostScript code. The "graphicsSystem" code indicates which kind of platform data structure is encapsulated by the canvas object. Part editors will use the standard platform drawing calls for the appropriate graphics system to render their contents on the canvas.

The "platformCanvas" field holds a reference to the platform data structure. It is NOT disposed of when the canvas object is deleted; it is the canvas' owner's responsibility to do that.

A canvas can be used several different ways. The "isDynamic" flag indicates whether the canvas represents a dynamic environment like a window or bitmap, or a static environment like a printed page. Part editors will display parts differently based on that distinction. For instance, a part may have scroll bars on the screen, but not when printed. Part editors may also use different drawing commands for printing or screen display.

The "isOffscreen" flag indicates whether the canvas is the main canvas of the window (or print job), or use for off-screen rendering. Offscreen canvases can be created by parts which wish to do double-buffering, image combination such as tinting or translucency, etc. Parts embedded on offscreen canvases have to do little to no extra work to display properly.

Offscreen canvases also maintain an update shape describing the area of the canvas which is invalid and needs to be redrawn, just like most platforms' windows do. This lets embedded parts interact with their drawing environment the same way whether in the window's canvas or offscreen.

An offscreen canvas is attached to a particular facet. The canvas has a back-pointer to that facet, and also a reference to the part that created it. That part is responsible for moving the contents of the canvas to its parent canvas when updating.

Theory of Operation

Invariants Maintained by Class

The canvas is created with a particular platformCanvas in a particular graphics system. These values will never change. The isDynamic and isOffscreen flags are also set for the lifetime of the object.

Other Persistent Properties

Member Functions

XMPCanvas

Function Prototype

XMPCanvas();

Protocol

Imaging

Protection

Public. No restrictions.

Override policy

Derived class **must** override.

Derived class **must** call base class behavior, before derived class behavior.

Basic operation

Class constructor

Inputs

None.

Outputs

<return> an XMPCanvas object.

Exceptions Signalled

kXMPErrInvalidPlatformCanvas

kXMPOutOfMemory cannot allocate XMPCanvas object.

Pre conditions

None.

Post conditions

None.

~XMPCanvas

Function Prototype

~XMPCanvas();

Protocol

Imaging

Protection

Public. No restrictions.

Override policy

Derived class **must** override.

Derived class **must** call base class behavior, after derived class behavior.

Basic operation

Class destructor. Delete internal data structures.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetFacet

Function Prototype

XMPFacet* GetFacet();

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return the facet this canvas is associated with.

Inputs

None.

Outputs

<return>	This canvas' facet.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetGraphicsSystem

Function Prototype

```
XMPGraphicsSystem GetGraphicsSystem();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return the code indicating the graphics system of this canvas.

Inputs

None.

Outputs

<return>	Graphics system code. Potential values are platform-dependent.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetOwner

Function Prototype

```
XMPPart* GetOwner();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return the part which is responsible for copying this canvas' image to its parent canvas.

Inputs

None.

Outputs

<return>	The owner part.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetPlatformCanvas

Function Prototype

```
XMPPlatformCanvas GetPlatformCanvas();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, before derived class behavior.

Basic operation

Return the platform canvas of this canvas.

Inputs

None.

Outputs

<return>	Pointer to the platform-specific canvas.

Exceptions Signalled

kXMPErrInvalidGrafPort	on the Mac

Pre conditions

None.

Post conditions

None.

GetUpdateShape

Function Prototype

```
XMPShape* GetUpdateShape();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return the shape describing the area of this canvas which needs to be redrawn.

Inputs

None.

Outputs

<return>	The update shape of this canvas.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

InitCanvas

Function Prototype

```
void InitCanvas(
        XMPGraphicsSystem graphicsSystem,
        XMPPlatformCanvas platformCanvas,
        XMPBoolean isDynamic,
        XMPBoolean isOffscreen);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Initialize a newly constructed canvas so it can be used. Assign parameters to fields, and allocate internal data structures.

Inputs

graphicsSystem	The code for the canvas' graphics system. Platform-specific.
platformCanvas	The platform-specific canvas data structure.Will NOT be disposed during destruction of this object.
isDynamic	kXMPTrue if canvas is dynamic, kXMPFalse if static.
isOffscreen	kXMPTrue if canvas is offscreen, kXMPFalse otherwise.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

Canvas was just constructed.

Post conditions

Canvas can be used for imaging.

Invalidate

Function Prototype

```
void Invalidate(
        XMPShape* shape);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Add "shape" to the update shape of this canvas. Also invalidates the corresponding shape on the parent canvas, if any.

Inputs

shape	The area to add to the update shape.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

IsDynamic

Function Prototype

```
XMPBoolean IsDynamic();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, before derived class behavior.

Basic operation

Returns true if this is a dynamic canvas, false if it is static.

Inputs

None.

Outputs

<return>	kXMPTrue if this canvas is dynamic, kXMPFalse if static.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

IsOffscreen

Function Prototype

```
XMPBoolean IsOffscreen();
```

Protocol

None.

Protection

Public. No restrictions.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, before derived class behavior.

Basic operation

Returns whether this canvas is offscreen.

Inputs

None.

Outputs

<return>	kXMPTrue if this canvas is offscreen, kXMPFalse otherwise.

Exceptions Signalled

None.

Pre conditions
None.
Post conditions
None.

Purge

Function Prototype

```
XMPSize Purge(
        XMPSize size);
```

Protocol
None.
Protection
Public.
Override policy
Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.
Basic operation
Attempt to free up to "size" bytes of memory by releasing any unneeded internal storage.
Inputs
size The number of bytes to attempt to free.
Outputs
None.
Exceptions Signalled
None.
Pre conditions
None.
Post conditions
None.

ResetUpdateShape

Function Prototype

```
void ResetUpdateShape();
```

Protocol
None.
Protection
Public.
Override policy
Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.
Basic operation
Set the update shape to be the empty shape.
Inputs
None.
Outputs
None.
Exceptions Signalled
None.
Pre conditions
None.
Post conditions
None.

SetFacet

Function Prototype

```
void SetFacet(
        XMPFacet* facet);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Set the facet field of this canvas. This should be done by the facet when the canvas is added to it.

Inputs

facet	The facet for the canvas.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

This canvas should be the canvas for "facet".

Post conditions

None.

SetOwner

Function Prototype

```
void SetOwner(
        XMPPart* owner);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Set the pwner part of this canvas. The owner is responsible for copying this canvas' image to its parent canvas.

Inputs

owner	The new owner part.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

Validate

Function Prototype

```
void Validate(
        XMPShape* shape);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Subtract "shape" from this canvas' update shape.

Inputs

shape	The area to subtract from the update shape.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

XMPClipboard

Basic Class Documentation

XMPClipboard provides a clipboard-like data transfer service between OpenDoc documents (and parts), and between a OpenDoc document and a non-OpenDoc document.

Platform vendor should implement this class. This class is not derived from any class. However, it depends on the system service which provides data transfer within a process and between processes. (For simplicity, this data transfer service will be called system clipboard in this document.)

Theory of Operation

At the process' startup time, a XMPClipboard object is instantiated and stored with the XMPSession. Whenever a part needs the clipboard service, it can acquire the XMPClipboard object through the XMPSession object.

The main client for this class is XMPPart. Whenever data transfer is required (e.g., a part responding to a menu event corresponding to Copy), the part should get the XMPClipboard object from XMPSession, acquire the clipboard lock (to be thread-safe), put data out to the XMPClipboard object through a XMPStorageUnit, and release the clipboard lock.

On the other hand, if a part desires to receive transfered data, the part should get the XMPClipboard object from XMPSession, acquire the clipboard lock (to be thread-safe), get data from the XMPClipboard object through a XMPStorageUnit, and release the clipboard lock.

Invariants Maintained by Class

The XMPStorageUnit object must be created using XMPDraft object. Both member fields should be valid from the construction to the destruction of the class. (Note that the client of this class should never cache the XMPClipboard object nor its XMPStorageUnit. Instead, it should get these objects from XMPSystemInterface and XMPClipboard APIs whenever they are needed.) The XMPStorageUnit returned is guaranteed to contain the content of the system clipboard.

Other Persistent Properties

Member Functions

Clear

Function Prototype

```
void Clear(
        XMPClipboardKey key);
```

Protocol

Data Interchange

Protection

Public. The clipboard lock must be acquired prior to invoking this method to be thread safe.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

This function removes all the user data stored in the XMPClipboard object and clears the system clipboard. The client must not be holding a storage unit reference returned by a prior call to GetContentStorageUnit.

Inputs

Key	A XMPClipboardKey value returned by Lock.

Outputs

None.

Exceptions Signalled

<platform errors>	Host platform errors.
kXMPErrInvalidClipboardKey	The argument clipboard key is not valid.

Pre conditions

This clipboard object has been initialized.

Post conditions

The next call to GetContentStorageUnit returns an XMPStorageUnit with no properties.

GetChangeID

Function Prototype

```
XMPChangeID GetChangeID();
```

Protocol

Data Interchange

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the change id identifying the clipboard content. XMPChangeID values should be tested for equality only. XMPChangeID may be called without acquiring the clipboard lock (but then there is no guarantee the clipboard won't change before the next clipboard operation). XMPChangeID values returned by this method are only valid during the current session.

Inputs

None.

Outputs

<return>	the change identification of the current clipboard content.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetContentStorageUnit

Function Prototype

```
XMPStorageUnit* GetContentStorageUnit(
        XMPClipboardKey key);
```

Protocol

Data Interchange, Storage

Protection

Public. The returned XMPStorageUnit object should not be cached by the caller. Also, the XMPClipboard object handles the creation and destruction of the

XMPStorageUnit. Therefore, the caller should neither dispose or release the returned object.
The clipboard lock should be acquired by caller prior to calling this method, and the lock should be relinquished after the data transfer.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

This function returns a XMPStorageUnit object. The initial content of the XMPStorageUnit comes from the system clipboard. The caller can read data from or write data to the XMPStorageUnit. The caller cannot use the XMPStorageUnit after unlocking the clipboard. The caller must not release the returned XMPStorageUnit.

Inputs

key	A XMPClipboardKey value returned by Lock.

Outputs

<return>	An XMPStorageUnit containing the content of the clipboard. This object should not be released.

Exceptions Signalled

<platform errors>	Host platform errors.
kXMPErrInvalidClipboardKey	The argument clipboard key is not valid.

Pre conditions

This clipboard object has been initialized.

Post conditions

The returned XMPStorageUnit contains the content of the system clipboard.

Lock

Function Prototype

```
XMPBoolean Lock(
        XMPULong wait,
        XMPClipboardKey* key);
```

Protocol

Thread safety

Protection

Public. Parts should call to obtain an XMPClipboardKey value required by other methods.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Acquire exclusive access to the content storage unit of the clipboard for the current thread. The value XMPTrue is returned if the lock is granted; the key parameter is set to a valid clipboard key. Access is granted if the current thread already holds the lock. Nested calls to Lock should be balanced by an equal number of calls to Unlock to relinquish the lock. The parameter allows the process to wait for access to be granted. A value of zero means no wait, a value of ULONG_MAX means an indefinite wait. Other values are platform-dependent intervals. A lock must be acquired before calling Clear(), GetContentStorageUnit(), or SetPlatformClipboard(), and must be relinquished (by calling Unlock()) after use of storage units on the clipboard is complete.

Inputs

wait	The interval to wait for access to be granted. A value of zero means no wait, a value of

	ULONG_MAX means an indefinite wait. Other values are platform-dependent.

Outputs

key	If the result is kXMPTrue, a valid key value required by routines that access or modify the link's content. If the result is kXMPFalse, key is an undefined invalid key.
<result>	kXMPTrue if access is granted and kXMPFalse if denied.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

If kXMPTrue is returned, the current thread has exclusive access to the clipboard, and the key parameter is valid.

SetPlatformClipboard

Function Prototype

```
void SetPlatformClipboard(
        XMPPlatformTypeSet* typeSet,
        XMPClipboardKey key
        );
```

Protocol

Data Interchange

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

If this OpenDoc clipboard has changed since the last change to the host platform clipboard, put data on the host clipboard. The data transferred is specified by the typeSet argument, which is a set of platform-dependent types. Data is placed on the platform clipboard in the same order as its corresponding ISO data on this clipboard. To keep host clipboard behavior consistent, translation to the argument type is not attempted.

If the typeSet argument is a nil object pointer, all content values are copied to the platform clipboard; no embedded parts on the clipboard are copied.

This routine should be called prior to using a platform-specific service that uses the host clipboard.

Inputs

typeSet	The set of platform types to deposit on the host clipboard, if present on this clipboard.
key	A non-zero XMPClipboardKey value returned by Lock.

Outputs

None.

Exceptions Signalled

kXMPErrOutOfMemory	Out of Memory
kXMPErrInvalidClipboardKey	The argument clipboard key is not valid.

Pre conditions

This clipboard object has been initialized.

Post conditions

Any of the agrument platform types from the last cut or copy operation are present on the host clipboard.

Unlock

Function Prototype

```
void Unlock(
        XMPClipboardKey key);
```

Protocol

Thread safety

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Relinquish a previously acquired clipboard lock.

Inputs

key	A XMPClipboardKey value returned by Lock.

Outputs

None.

Exceptions Signalled

kXMPErrInvalidClipboardKey	The argument clipboard key is not valid.

Pre conditions

The key argument is a valid clipboard key, implying exclusive access to the clipboard by this thread.

Post conditions

The key argument is no longer valid. The current thread retains exclusive access to the clipboard if the thread already had exclusive access when the argument key was returned by Lock.

XMPContainer

Basic Class Documentation

A Container is a physical collection of bytes, like a file or a section of memory. Each Container contains one or more Documents. Each Document in turn contains one or more Drafts. Each Draft in turn contains one or more XMPStorageUnit.

XMPContainer is a class used to manipulate this physical Container. Similarly, XMPDocument is a class used to manipulate a Document; XMPDraft maniuplates a Draft; and XMPStorageUnit manipulates a Storage Unit. This set of related XMPContainer, XMPDocument, XMPDraft and XMPStorageUnit classes are collectively called a Container Suite. They are usually implemented together as a set and work intimately with each other.

The class documented here (XMPContainer) is a abstract base class. Container Suite Implementors should subclass XMPContainer to provide the functionality of a OpenDoc Container for their Container Suite.

Theory of Operation

XMPContainer is derived from XMPRefCntObject. When XMPContainer is first constructed (either through XMPAbsStorageSystem::CreateContainer or XMPAbsStorageSystem::GetContainer), its reference count is 1. Every time XMPAbsStorageSystem::GetContainer is called with containerType and id referring to this Container, the refCount of the corresponding XMPContainer object is incremented by 1. When XMPContainer is no longer needed, XMPContainer::Release should be called.

This class is never directly instantiated by anyone except the Storage System. When the Shell or a Container App needs to create or get an OpenDoc container, it will call the corresponding XMPStorageSystem method. The XMPStorageSystem method will then instantiate a XMPContainer object to manipuate the physical Container.

Invariants Maintained by Class

There is at most one (i.e. 0 or 1) XMPDocument object for every XMPDocumentID requested via GetDocument.

Until a requested document has been Released, the XMPDocument object will be valid. After a requested document has been Released, the XMPDocument object which referred to that document is no longer valid.

It is the responsibility of XMPContainer to maintain these invariants, not XMPDocument, since XMPContainer maintains the collection of XMPDocuments.

Other Persistent Properties

Member Functions

XMPContainer

This method is only of interest to Container Application developers.

Function Prototype

XMPContainer();

Protocol

Constructor

Protection

Public. Private by convention. This method is only called by XMPStorageSystem.

Override policy

Derived class **must** override.

Derived class **must** call base class behavior, before derived class behavior.

Basic operation

Constructor of the class.

Inputs

None.

Outputs

<return> a valid XMPContainer object

Exceptions Signalled

none

Pre conditions

None.

Post conditions

An un-Initialized XMPContainer object is created.

~XMPContainer

This method is only of interest to Container Application developers.

Function Prototype

~XMPContainer();

Protocol

Destructor

Protection

Public. Private by convention. This method is only called by XMPStorageSystem.

Override policy

Derived class **must** override.

Derived class **must** call base class behavior, after derived class behavior.

Basic operation

Deletes its XMPDocument object (if exists), and closes the container.

Inputs

none

Outputs

none

Exceptions Signalled

none

Pre conditions

this is a valid XMPContainer object.

Post conditions

The container associcated with this XMPContainer object is closed.
this is no longer a valid XMPContainer object.

Close

This method is only of interest to Container Application developers.

Function Prototype

XMPContainer* Close();

Protocol

Container Manipulation

Protection

Public. Private by convention. This method should only be called by XMPStorageSystem::ReleaseContainer.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Closes an access session to the container associated with this XMPContainer object.

Inputs

none

Outputs

<return> this

Exceptions Signalled

none

Pre conditions

There is an open access session to the container associated with this XMPContainer object

Post conditions

There is no access session to the container associated with this XMPContainer object

Create

This method is only of interest to Container Application developers.

Function Prototype

```
XMPContainer* Create();
```

Protocol

Container Manipulation

Protection

Public. Private by convention. This method should only be called by XMPStorageSystem::CreateContainer.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Creates the physical Container using this XMPContainer object and associates the created container with this XMPContainer object. This function does NOT open the container.

Inputs

none

Outputs

<return> this

Exceptions Signalled

kXMPErrCannotCreateContainer Cannot create the physical Container.

Pre conditions

None.

Post conditions

There exists a physical Container which conforms to the specification contained in this XMPContainer object.

GetDocument

This method is only of interest to Container Application developers.

Function Prototype

```
XMPDocument* GetDocument(
        XMPDocumentID id);
```

Protocol

Getter

Protection

Public. The Shell calls this method when a OpenDoc Document is opened.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns a fully functional XMPDocument object associated with the given id. (A fully functional object is one that has been initialized and can be used right away).

Inputs

id	ID of the document reqeusted

Outputs

<return>	A fully functional XMPDocument object.

Exceptions Signalled

kXMPErrInvalidDocument	Cannot get desired Document because the id is invalid.

Pre conditions

None.

Post conditions

The refCount of XMPDocument object is incremented by 1 if XMPDocument object is in the XMPDocument collection prior to this call. Otherwise, the refCount of XMPDocument object should be 1.

GetID

Function Prototype

```
XMPContainerID GetID();
```

Protocol

Getter

Protection

Public. A Part should not need to worry about the ID of its container.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the ID of this object. This ID should be the one used in XMPAbsStorageSystem::CreateContainer or XMPAbsStorageSystem::CreateContainer.

Inputs

none

Outputs

<return>	ID of the Container.

Exceptions Signalled

none

Pre conditions

None.

Post conditions

None.

GetName

Function Prototype

```
XMPContainerName GetName();
```

Protocol

Getter

Protection

Public. A Part can call this method but it should not need to know its Container's name.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns a copy of the name of this container. If the Container does not have a name, kXMPNULL is returned.

Inputs

none

Outputs

<return>	Name of the container.

Exceptions Signalled

none

Pre conditions

None.

Post conditions

None.

GetStorageSystem

Function Prototype

```
XMPStorageSystem* GetStorageSystem();
```

Protocol

Getter

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the XMPStorageSystem object associated with this XMPContainer object.

Inputs

none

Outputs

<return>	XMPStorageSystem object associated with this XMPContainer Object.

Exceptions Signalled

none

Pre conditions

None.

Post conditions

None.

IncrementRefCount

Function Prototype

```
void  IncrementRefCount();
```

Protocol

Reference Counting

Protection

Public. Can be called by any class which needs a reference to this XMPContainer object.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Increments the reference count of the Container.

Inputs

none

Outputs

None

Exceptions Signalled

none

Pre conditions

None.

Post conditions

The reference count for this object is incremented by 1.

InitContainer

This method is only of interest to Container Application developers.

Function Prototype

void InitContainer();

Protocol

Construction

Protection

Public. Private by convention. This method should only be called by XMPStorageSystem::CreateContainer or XMPStorageSystem::GetContainer.

Override policy

Derived class **must** override.

Derived class **must** call base class behavior, before derived class behavior.

Basic operation

Initializes this XMPContainer object.

Inputs

none

Outputs

none

Exceptions Signalled

none

Pre conditions

None.

Post conditions

this XMPContainer object has been properly initialized.

Open

This method is only of interest to Container Application developers.

Function Prototype

XMPContainer* Open();

Protocol

ObjectStorage

Protection

Public. Private by convention. This method should only be called by XMPStorageSystem::CreateContainer and XMPStorageSystem::GetContainer.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Opens an access session to the container associated with this XMPContainer object.

Inputs

None

Outputs

<return> this

Exceptions Signalled

kXMPErrCannotOpenContainer Cannot open the physical Container.

Pre conditions

There is no access session to the container associated with this XMPContainer object.

Post conditions

There is an open access session to the container associated with this XMPContainer object.

Purge

This method is only of interest to Container Application developers.

Function Prototype

```
XMPSize Purge(
        XMPSize size);
```

Protocol

ObjectStorage

Protection

Public. Private by convention. This method should only be called by XMPStorageSystem::Purge.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Purges memory from ephemeral store until size bytes (not necessarily contiguous) have been freed up.

Inputs

size	Number of bytes to purge

Outputs

<return>	Number of bytes actually purged

Exceptions Signalled

none

Pre conditions

None.

Post conditions

Either size bytes (not necessarily contiguous) are free in the default heap or Purge() has been called on every XMPDocument object associated with this XMPContainer object.

Release

This method is only of interest to Container Application developers.

Function Prototype

```
XMPStorageSystem*   Release();
```

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Decrements the ref count of this XMPContainer object.

Inputs

none

Outputs

<return>	fStorage

Exceptions Signalled

none

Pre conditions

None.

Post conditions

This XMPContainer object is no longer guaranteed to be a valid XMPContainer object

ReleaseDocument

This method is only of interest to Container Application developers.

Function Prototype

```
XMPContainer* ReleaseDocument(
        XMPDocument* document);
```

Protocol

Document Manipulation

Protection

Public. Private by convention. This method should only be called by XMPDocument::Release.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Releases the XMPDocument object.

Inputs

document	the document to be released

Outputs

<return>	this XMPContainer object.

Exceptions Signalled

none

Pre conditions

document is a valid XMPDocument object in this container which was obtained via GetDocument.
The reference count of document is 0.

Post conditions

document is no longer a valid XMPDocument object.

SetName

This method is only of interest to Container Application developers.

Function Prototype

```
void SetName(
        XMPContainerName name);
```

Protocol

Setter

Protection

Public. Called by the Shell or the Container App to set the name of this Container.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Sets the name of this container.

Inputs

name	the new name of this container

Outputs

none

Exceptions Signalled

none

Pre conditions

name is a valid XMPContainerName

Post conditions

The name of the container is name.

XMPDispatcher

Basic Class Documentation

XMPDispatcher defines an extensible event dispatcher for distributing events to parts.

Each OpenDoc session has a single dispatcher object, accessed by calling XMPSession::GetDispatcher().

The dispatcher is called from the application shell's event loop to dispatch events to parts.

XMPDispatcher is a derived class of XMPObject.

A related class is XMPDispatchModule.

XMPDispatcher is implemented by platform vendors. It is platform-specific, since different platforms have different models for handling events.

Theory of Operation

The dispatcher is responsible for distributing events to parts. Events not handled by a part are handled by the shell application or dropped on the floor. The actual distribution of events is performed by dispatch modules. The dispatcher maintains a dictionary associating event codes with XMPDispatchModule objects. One or more dispatch modules will handle standard events, but part handlers can add XMPDispatchModules for new event types.

A dispatch module can also be installed as a monitor. In this case, it gets to see events of a certain type, but does not “swallow them up”. No assumptions can be made regarding the order in which events are monitored and handled.

Events are passed to parts using a single bottleneck method XMPPart::HandleEvent(). Part Editors or a part editor framework must examine the event data to determine the type of event.

Invariants Maintained by Class

XMPDispatcher maintains a dictionary of XMPDispatchModule objects, indexed by event type. The dictionary contains at least one dispatch module for handling the standard events of a particular platform.

XMPDispatcher also contains a reference to the system interface, for easy access.

Other Persistent Properties

No persistent properties

Member Functions

AddDispatchModule

Function Prototype

```
void AddDispatchModule(
        XMPEventType eventType,
        XMPDispatchModule* dispatchModule);
```

Protocol

Adding and Removing Dispatch Modules

Protection

Public. Not called by most parts.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Adds the specified dispatch module to the dictionary of dispatch modules.

Inputs

eventType	An event code
dispatchModule	A dispatch module for that event

Outputs

None.

Exceptions Signalled

kXMPErrExistingDispatchModule	module is already installed.
kXMPErrInvalidDispatchModule	specified dispatch module is invalid.

Pre conditions

A valid initialize instance. A valid initialized dispatch module.

Post conditions

The dictionary contains the specified dispatch module.

AddMonitor

Function Prototype

```
void AddMonitor(
        XMPEventType eventType,
        XMPDispatchModule* dispatchModule);
```

Protocol

Adding and Removing Dispatch Modules

Protection

Public. Not called by most parts.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Adds the specified dispatch module as a monitor for the specified event type.

Inputs

eventType	An event code
dispatchModule	A dispatch module for that event

Outputs

None

Exceptions Signalled

kXMPErrExistingDispatchModule	module is already installed.
kXMPErrInvalidDispatchModule	specified dispatch module is invalid.

Pre conditions

A valid initialize instance. A valid initialized dispatch module.

Post conditions

The dictionary contains the specified dispatch module as a monitor.

Dispatch

Function Prototype

```
XMPBoolean Dispatch(
        XMPEventData event);
```

Protocol

Event Handling

Protection

Public. Called by the OpenDoc shell application, or by container applications. May be called by Parts handling events in dialogs.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Dispatches the event to the appropriate part. Returns kXMPFalse if the event does not belong to any part, or if Part::HandleEvent returns kXMPFalse indicating that the event was not handled.

Inputs

event	A platform-dependent event structure

Outputs

<return>	kXMPTrue if the event was handled, kXMPFalse otherwise.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

Result contains kXMPTrue if any part handled the event, kXMPFalse otherwise.

GetMouseRegion

This method is Macintosh specific.

Function Prototype

```
XMPRgnHandle GetMouseRegion();
```

Protocol

Cursor Tracking

Protection

Public. Call by the shell or container apps.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the mouse region, which can be set by parts using GetMouseRegion(). The mouse region is recomputed if necessary, by calling Part::MouseEnter/MouseLeave/MouseWithin.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

Triggers a recomputation.

InvalidateFacetUnderMouse

This method is Macintosh specific.

Function Prototype

```
void InvalidateFacetUnderMouse();
```

Protocol

Cursor Tracking

Protection

Public. Called by Imaging subsystem when a facet is removed.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Sets the dispatcher's cache of the facet under the mouse to kXMPNULL.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

fFacetUnderMouse is kXMPNULL.

Purge

Function Prototype

```
XMPSize Purge(
        XMPSize size);
```

Protocol

Low Memory

Protection

Public. Called by OpenDoc.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Tries to free up some memory, usually by flushing internal state to external storage.

Inputs

size	The amount of memory requested.

Outputs

<return>	The amount of memory freed.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance

Post conditions

Some memory has been freed up.

RegisterIdle

This method is Macintosh specific.

Function Prototype

```
void RegisterIdle(
        XMPPart* part,
        XMPFrame* frame,
        XMPIdleFrequency frequency);
```

Protocol

Event Handling

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Registers the specified frame for idle time. The frame can be kXMPNULL, in which case the part as a whole gets idle time.

Inputs

frame	A frame which should receive idle messages.
frequency	How often to send idle messages.
part	The part which is interested in idle time

Outputs

None.

Exceptions Signalled

kXMPErrInvalidFrame	The specified frame is invalid

Pre conditions

A valid initialized instance.

Post conditions

The dispatcher contains the part/frame in its idle list.

RemoveDispatchModule

Function Prototype

```
void RemoveDispatchModule(
        XMPEventType eventType);
```

Protocol

Adding and Removing Dispatch Modules

Protection

Public. Not called by most parts.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Removes the dispatch module for the specified event from the dictionary.

Inputs

eventType	an event type

Outputs

None.

Exceptions Signalled

None.

Pre conditions

A dispatch module is installed for the specified event type.

Post conditions

The specified event type no longer has its associated dispatch module.

RemoveMonitor

Function Prototype

```
void RemoveMonitor(
        XMPEventType eventType,
        XMPDispatchModule* dispatchModule);
```

Protocol

Adding and Removing Dispatch Modules

Protection

Public. Not called by most parts.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Removes the specified monitor for the specified event from the dictionary.

Inputs

eventType	an event code
dispatchModule	A dispatchmodule which monitors the event type

Outputs

None.

Exceptions Signalled

None.

Pre conditions

The dispatch module is installed as a monitor for the specified event type.

Post conditions

The dispatch module is no longer a monitor for the specified event type.

SetMouseRegion

This method is Macintosh specific.

Function Prototype

```
void SetMouseRegion(
        XMPRgnHandle area);
```

Protocol

Cursor Tracking

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Sets the mouse region, which can be queried by the shell application using GetMouseRegion().

Inputs

area	the new mouse region

Outputs

None.

Exceptions Signalled

None.

Pre conditions

A valid shape.

Post conditions

Stored value is updated.

ShouldTerminateEventLoop

Function Prototype

```
XMPBoolean ShouldTerminateEventLoop();
```

Protocol

Event Handling

Protection

Public. Generally called by the OpenDoc shell application to see if the dispatcher recommends the loop be terminated.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns a Boolean value indicating whether the OpenDoc shell application should terminate.

Inputs

None.

Outputs

<return>	The value of fTerminateEventLoop

Exceptions Signalled

None.

Pre conditions

A valid and initialized instance.

Post conditions

Result contains the value of fTerminateEventLoop.

TerminateEventLoop

Function Prototype

```
void TerminateEventLoop();
```

Protocol

Event Handling

Protection

Public. Can be called as a signal to the shell application to close down its process. Use with care.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Sets a Boolean flag in the dispatcher which the shell application can access using ShouldTerminateEventLoop()

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

fTerminateEventLoop contains the value kXMPTrue.

UnregisterIdle

This method is Macintosh specific.

Function Prototype

```
void UnregisterIdleFrame(
        XMPPart* part,
        XMPFrame* frame);
```

Protocol

Event Handling

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Removes the specified part/frame from the collection of parts interested in receiving idle messages.

Inputs

frame	A frame which should no longer receive idle messages.
part	The part which was receiving idle events

Outputs

None.

Exceptions Signalled

kXMPErrInvalidFrame	The specified frame is invalid

Pre conditions

A valid initialized instance.

Post conditions

The dispatcher does not contain the part/frame in its idle list.

Yield

This method is Macintosh specific.

Function Prototype

```
void Yield(
        XMPFrame* frame);
```

Protocol

Event Handling

Protection

Public. Called by parts during a long-running operation.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Gives processor time to other parts.

Inputs

frame	The frame which is yielding

Outputs

None.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

No effect on internal state.

XMPDispatchModule

Basic Class Documentation

XMPDispatchModule is an abstract class. The dispatcher can be extended by adding subclasses of XMPDispatchModule.
Related classes are XMPDispatcher and XMPStandardDispatchModule, which is private to part developers.
XMPDispatchModule has no base class.

Theory of Operation

See XMPDispatcher

Invariants Maintained by Class

The field fSession points to the global system interface.

Other Persistent Properties

Member Functions

XMPDispatchModule

Function Prototype

XMPDispatchModule();

Protocol

Creation and Deletion

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Constructs a dispatch module. InitDispatchModule must be called.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

A constructed dispatch module. Not yet usable - InitDispatchModule must be called.

~XMPDispatchModule

Function Prototype

~XMPDispatchModule();

Protocol

Creation and Deletion

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Frees the memory allocated by this object

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

It's gone.

XMPDocument

Basic Class Documentation

A Container contains one or more Document. Each Document has a unique XMPDocumentID within a Container. Each of these Docment contains one or more Draft. Each of these Drafts corresponds to a version of the Document. In other words, a Document is a collection of versioned Drafts. A XMPDocument object is used to manipulate a Document and its Drafts.
The class documented here is a abstract base class. Container Suite implementors should subclass this class to provide the functionality of a OpenDoc Document for their Container Suite. (For definition of Container Suite, please refer to documentation on XMPContainer).
Every OpenDoc Document by default has a base Draft.

Theory of Operation

XMPDocument is derived from XMPRefCntObject. A XMPDocument object is instantiated when GetDocument of its corresponding XMPContainer. When XMPDocument is first constructed, its refCount is 1. Every time XMPContainer::GetDocument is called using the same id, the refCount of the XMPDocument object is incremented by 1. When XMPDocument object is no longer needed, XMPDocument::Release should be called.
The XMPDocument object is responsible for guaranteeing that there is only one XMPDraft object
associated with each Draft within it.

XMPDocument is only instantiated by XMPContainer of the same Container Suite. If the Shell or the Container App wants to create or get a Document, it has to call XMPContainer::GetDocument.

Invariants Maintained by Class

There is at most one XMPDraft object for every XMPDraftID requested via GetDraft or CreateDraft.
Until a requested XMPDraft has been Released, the XMPDraft object will be valid. After a requested draft has been Released, the XMPDraft object which referred to that draft is not guaranteed to be valid.
It is the responsibility of XMPDocument to maintain these invariants, not XMPDraft, since XMPDocument maintains the collection of XMPDrafts.

Other Persistent Properties

A XMPDocument must have a XMPContainer object associated with it.
Every Document (hence XMPDocument object) within a Container must have a unique XMPDocumentID.
Every Document can have at most one name.
Drafts are linearly derived in a Document. The top Draft is the only Draft that can be modified. All the Drafts below the top Draft are read-only.

Member Functions

XMPDocument

This method is only of interest to Container Application developers.

Function Prototype

```
XMPDocument();
```

Protocol

Constructor

Protection

Public. Private within Storage Subsystem. This method should only be called by XMPContainer::GetDocument.

Override policy

Derived class **can** override.

Derived class **must** call base class behavior, before derived class behavior.

Basic operation

Creates an uninitialized XMPDocument object.

Inputs

None.

Outputs

<return> A valid XMPDocument object.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

~XMPDocument

This method is only of interest to Container Application developers.

Function Prototype

```
~XMPDocument();
```

Protocol

Destructor

Protection

Public. Private within Storage Subsystem. This method should only be called by XMPContainer::ReleaseDocument.

Override policy

Derived class **can** override.

Derived class **must** call base class behavior, before derived class behavior.

Basic operation

This is the destructor of the XMPDocument class object. It releases all the memory associated with this XMPDocument class object.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

this is a valid XMPDocument object.

Post conditions

this is no longer a valid XMPDocument object.

CollapseDrafts

This method is only of interest to Container Application developers.

Function Prototype

```
XMPDocument* CollapseDrafts(
        XMPDraft* from,
        XMPDraft* to);
```

Protocol

Draft Manipulation

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Removes all the drafts between from (inclusive) and to (exclusive). All the Drafts between from (inclusive) and to (exclusive) must be empty.
This call is used for getting rid of unnecessary Drafts.

Inputs

from	Draft to collapse from
to	Draft to collapse to

Outputs

<return>	this XMPDocument object

Exceptions Signalled

kXMPErrNonEmptyDraft	Non-empty draft between from and to.
kXMPErrCannotCollapseDrafts	from is not above to.
kXMPErrOutstandingDraft	There is one or more outstanding Draft (i.e., refCount >= 1) between from and to.

Pre conditions

The drafts between from (inclusive) and to (exclusive) are 'empty'.
There is no outstanding Draft (i.e., with refCount >= 1) between from (exclusive) and to (exclusive).
from must have a refCount of 1.
from must not be the base Draft.
from must be above to in the Draft topology.

Post conditions

The drafts between from (inclusive) and to (exclusive) have been removed and the appropriate draft topology is maintained.
from is no longer a valid XMPDraft object.

CreateDraft

This method is only of interest to Container Application developers.

Function Prototype

```
XMPDraft* CreateDraft(
        XMPDraft* below,
        XMPBoolean releaseBelow);
```

Protocol

Draft Manipulation

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Creates a new Draft on top of below and returns a XMPDraft object referring to the created Draft.
If releaseBelow is kXMPTrue, the below draft is released.
If this container does not support creation of drafts (e.g. a read-only medium like a CD-ROM), an exception is raised.

Inputs

below	The draft on which the new draft is to be created.
releaseBelow	Boolean to show whether or not the below draft should be released.

Outputs

<return>	a fully functional XMPDraft object with Exclusive-Write permissions.

Exceptions Signalled

kXMPErrInvalidContainer	Container does not support creation of new drafts.
kXMPErrInvalidBelowDraft	below is kXMPNULL or below is not the Top Draft of the Document.
kXMPErrInvalidPermissions	Exclusive-Write-Only below is not released.

Pre conditions

below is the Top Draft of the Document.
If below is Exclusive-Write-only, releaseBelow must be kXMPTrue.

Post conditions

None.

GetBaseDraft

This method is only of interest to Container Application developers.

Function Prototype

```
XMPDraft* GetBaseDraft(
        XMPDraftPermissions perms);
```

Protocol

Draft Manipulation

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns an XMPDraft object which refers to the base draft of the document with the given perms.

Inputs

perms	permissions

Outputs

<return>	XMPDraft object which refers to the base draft of this document

Exceptions Signalled

kXMPErrInvalidPermissions	Invalid permissions.

Pre conditions

perms can be Read-Only if the Draft has not been gotten for Exclusive -Write-only.
perms can be Exclusive-Write-only if the Draft has not been gotten (i.e., there is no valid XMPDraft object referring to the desired Draft) and the Base Draft is the top Draft in the Document.
perms can be Shared-Write if the Draft has not been gotten for Read-Only or Exclusive-Write-Only and the Base Draft is the top Draft of the Document.

Post conditions

None.

GetContainer

Function Prototype

```
XMPContainer* GetContainer();
```

Protocol

Getter

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the XMPContainer object with which this XMPDocument object is associated.

Inputs

none

Outputs

<return>	fContainer

Exceptions Signalled

none

Pre conditions

None.

Post conditions

None.

GetDocumentProperties

This method is only of interest to Container Application developers.

Function Prototype

```
XMPStorageUnit*    GetDocumentProperties();
```

Protocol

Getter

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns a XMPStorageUnit object where meta-information on this Document can be stored. Note that the client has to call XMPStorageUnit::Release when the XMPStorageUnit is not needed anymore.

Inputs

None.

Outputs

<return>	XMPStorageUnit object where meta-information on this Document is stored.

Exceptions Signalled

kXMPErrNoDocumentProperties	Cannot create XMPStorageUnit for Document Properties.

Pre conditions

None.

Post conditions

None.

GetDraft

This method is only of interest to Container Application developers.

Function Prototype

```
XMPDraft* GetDraft(
        XMPDraftPermissions perms,
        XMPDraftID id,
        XMPDraft* draft,
        XMPPositionCode posCode,
        XMPBoolean release);
```

Protocol

Draft Manipulation

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

If id is not 0, it is used to identify the Draft and the draft and posCode parameters are ignored.
If id is 0, draft and posCode are used to identify the desired draft.
Once the draft is identified, a XMPDraft object referring to the desired draft is created and returned with the given permissions.
The release parameter is only used if draft and posCode are used to identify the Draft. In this case, if release is kXMPTrue, draft is released if the desired draft can be gotten.
It is valid to get the same Draft using draft and kXMPPosSame. In this case, the Draft is gotten again with the appropriate permissions. This provides an easy way to change the permissions for a XMPDraft.

Inputs

perms	permissions
id	draft id
draft	XMPDraft object used together with posCode to identify the desired Draft.
posCode	relative position (used together with draft) to identify the desired Draft.
release	XMPBoolean to show whether draft should be released when getting the desired one. (Only used when relative positioning is done).

Outputs

<return>	a fully functional XMPDraft object

Exceptions Signalled

kXMPInvalidPermissions	Invalid permissions.
kXMPErrInvalidRefCount	draft does not have a valid refCount.
kXMPErrCannotChangePermissions	Cannot change permission on draft if it is gotten again with a different permissions.
kXMPErrInvalidPosCode	Invalid relative position.

Pre conditions

Either id refers to a valid Draft in this Document or draft and posCode together refer to a valid Draft in this Document.
perms can be Read-Only if the Draft has not been gotten for Exclusive -Write-only.
perms can be Exclusive-Write-only if the Draft has not been gotten (i.e., there is no valid XMPDraft object referring to the desired Draft) and the Draft is the top Draft of the Document.
perms can be Shared-Write if the Draft has not been gotten for Read-Only or Exclusive-Write-Only and the Draft is the top Draft of the Document.

Post conditions

The specified draft, opened with the given perms is returned.
If release is kXMPTrue and the draft is a valid XMPDraft object referring to a draft in the same document, it should be released.

GetID

Function Prototype

```
XMPDocumentID GetID();
```

Protocol

Getter

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the ID of the document referred to by this XMPDocument object. The returned ID should be the one used for getting this object in XMPContainer::GetDocument.

Inputs

none

Outputs

<return>	ID of the Document.

Exceptions Signalled

none

Pre conditions

None.

Post conditions

None.

GetName

Function Prototype

```
XMPDocumentName* GetName();
```

Protocol

Getter

Protection

Public. A Part can call this method, but it should not need to deal with the name of the Document directly.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns a copy of the name of this Document. If this Document does not have a name, kXMPNULL is returned.

Inputs

none

Outputs

<return>	Name of the Document. kXMPNULL if the Document does not have any name.

Exceptions Signalled

none

Pre conditions

None.

Post conditions

None.

IncrementRefCount

This method is only of interest to Container Application developers.

Function Prototype

```
void IncrementRefCount();
```

Protocol

Reference Counting

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Increments the reference count of this XMPDocument object.

Inputs

none

Outputs

none

Exceptions Signalled

none

Pre conditions

None.

Post conditions

Reference Count of this object is incremented by 1.

InitDocument

This method is only of interest to Container Application developers.

Function Prototype

```
void InitDocument(
        XMPContainer* container,
        XMPDocumentID id);
```

Protocol

Initialization

Protection

Public. Private within Storage Subsystem. This method should only be called by XMPContainer::GetDocument.

Override policy

Derived class **can** override.

Derived class **must** call base class behavior, before derived class behavior.

Basic operation

Initializes this XMPDocument object.

Inputs

container	XMPContainer object with which this XMPDocument is associated.
id	ID of the Document

Outputs

none

Exceptions Signalled

none

Pre conditions

this is a valid XMPDocument object.

Post conditions

this is an initialized XMPDocument object.

Purge

This method is only of interest to Container Application developers.

Function Prototype

```
XMPSize Purge(
        XMPSize size);
```

Protocol

Memory Management

Protection

Public. Private within Storage Subsystem. This method should only be called by XMPContainer::Purge.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Purge memory from ephemeral store until size bytes (not necessarily contiguous) have been freed up.

Inputs

size	the number of bytes to purge

Outputs

<return>	the number of bytes actually purged

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

Either size bytes (not necessarily contiguous) are free in the default heap or Purge() has been called on every XMPDraft object associated with this XMPDocument object.

Release

This method is only of interest to Container Application developers.

Function Prototype

```
void Release();
```

Protocol

Reference counting

Protection

Public.

Override policy

Derived class **must** override.

Derived class **must** call base class behavior, after derived class behavior.

Basic operation

Releases this XMPDocument object. If the reference count of this XMPDocument goes down to zero, this method should notify its XMPContainer object by calling XMPContainer::ReleaseDocument.

Inputs

none

Outputs

None.

Exceptions Signalled

none

Pre conditions

None.

Post conditions

this document is no longer guaranteed to be a valid XMPDocument object

ReleaseDraft

This method is only of interest to Container Application developers.

Function Prototype

```
XMPDocument* ReleaseDraft(
        XMPDraft* draft);
```

Protocol

Draft Manipulation

Protection

Public. Private within Storage Subsystem. This method should only be called by XMPDraft::Release.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Releases the given XMPDraft object.

Inputs

draft	the draft to be released

Outputs

<return>	this XMPDocument object.

Exceptions Signalled

None.

Pre conditions

draft has a refCount of 0.

Post conditions

None.

SaveToAPrevDraft

This method is only of interest to Container Application developers.

Function Prototype

```
void SaveToAPrevDraft(
        XMPDraft* from,
        XMPDraft* to);
```

Protocol

Draft Manipulation

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Moves changes from the from draft to the to draft. If to is kXMPNULL, the changes are saved to the draft immediately below from. All the Drafts from from (inclusive) to to Draft (exclusive) are empty after this operation.
This method can be used together with CollapseDrafts to get rid of unwanted Drafts.

Inputs

from	draft to move changes from
to	draft to move changes to

Outputs

none

Exceptions Signalled

kXMPErrInvalidDraft	from is not above to.
kXMPErrOutstandingDraft	There is one or more outstanding Draft between from (exclusive) and to (exclusive).

Pre conditions

from Draft is above to Draft.
There is no outstanding Draft (i.e., refCount >= 1) between from (exclusive) and to (exclusive).

Post conditions

The Drafts from from (inclusive) to to (exclusive) have had all their 'changes' removed and all such changes have been incorporated into the to Draft. In other words, the Drafts from from (inclusive) to to (exclusive) are 'empty'.

SetBaseDraftFromForeignDraft

This method is only of interest to Container Application developers.

Function Prototype

```
void SetBaseDraftFromForeignDraft(
        XMPDraft* draft);
```

Protocol

Document Manipulation; Draft Manipulation

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Takes a version from one Document and copies its content to set up a new Document.

Inputs

draft	draft to copy to base

Outputs

None.

Exceptions Signalled

None.

Pre conditions

draft is not from the same Document.

This Document is a newly created Document with only an empty Base Draft.

Post conditions

This Document's Base Draft contains the same data as draft.

SetName

This method is only of interest to Container Application developers.

Function Prototype

```
void SetName(
        XMPDocumentName name);
```

Protocol

Setter

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Sets the name of this document.

Inputs

name	New name of this Document

Outputs

none

Exceptions Signalled

none

Pre conditions

name is a valid XMPDocumentName.

Post conditions

The name of the document is name.

XMPDraft

Basic Class Documentation

An OpenDoc Draft contains information pertaining to a particular version of an OpenDoc Document.
XMPDraft objects are instantiated by calling GetDraft or CreateDraft of their XMPDocument object.
The class documented here is a abstract base class. Container Suite implementors should subclass this class to provide the functionality of a OpenDoc Document for their Container Suite. (For definition of Container Suite, please refer to documentation on XMPContainer).

Draft Properties:
- o handler preferences (for workgroups)
  (type or category, handler)*
- o mod date
  dateType
- o create date
  dateType
- o user last modified by
  usernametype/TEXTstringtype
- o comments
  Part?
- o mailerInfo
  whater AOCE says
- o memory high water mark
  long
- o has been 'Save'd by user
  boolean

Theory of Operation

This class is derived from XMPRefCntObject. A XMPDraft object is instantiated when CreateDraft or GetDraft is called on its corresponding XMPDocument. When XMPDraft is first constructed, its refCount is 1. Every time XMPDocument::GetDraft is called on the same Draft, the refCount of the XMPDraft object is incremented by 1. When XMPDraft object is no longer needed, XMPDraft::Release should be called.
The XMPDraft object is responsible for guaranteeing that there is only one XMPStorageUnit object
associated with each Storage Unit or persistent object within it.
Access is guaranteed to be exclusive if the Draft has just been created (through CreateDraft) or if kDPExclusiveWrite permissions have been requested and granted on GetDraft.
Access is not guaranteed to be exclusive if the Draft has been 'gotten' with any other permissions.

XMPDraft is only instantiated by XMPDocument of the same Container Suite. If the Shell wants to create or get a Draft, it should call XMPDocument::CreateDraft or XMPDoucment::GetDraft. If the Base Draft is needed, XMPDocument::GetBaseDraft should be called.

Invariants Maintained by Class

There is at most one XMPPersistentObject (or its derived object) for every persistent object requested via Get/Create Part/Frame/Link.

Until a requested persistent object has been Released, the persistent object will remain valid. After a requested XMPPersistentObject has been Released, the XMPPersistentObject is no longer valid.
It is the responsibility of XMPDraft to maintain these invariants, not the XMPPersistentObjects or (their derived objects), since XMPDraft maintains the collection of first class persistent objects.
There can only be one Clone (initiated by BeginClone and terminated by AbortClone or EndClone) at a time.
A Draft can have at most one name.

Other Persistent Properties

Member Functions

XMPDraft

This method is only of interest to Container Application developers.

Function Prototype

XMPDraft();

Protocol

ObjectStorage

Protection

Public. Private within Storage System. This method should only be called by XMPDocument::CreateDraft or XMPDocument::GetDraft.

Override policy

Derived class **must** override.

Derived class **must** call base class behavior, before derived class behavior.

Basic operation

This function is the constructor of the class.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

~XMPDraft

This method is only of interest to Container Application developers.

Function Prototype

~XMPDraft();

Protocol

ObjectStorage

Protection

Public. Private within Container Suite. This method should only be called by XMPDocument::ReleaseDraft.

Override policy

Derived class **must** override.

Derived class **must** call base class behavior, after derived class behavior.

Basic operation

This function is the destructor of the class.

Inputs

none

Outputs

none

Exceptions Signalled

none

Pre conditions

None.

Post conditions

None.

AbortClone

Function Prototype

```
void AbortClone(
        XMPDraftKey key);
```

Protocol

Object Management

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Aborts the transacton started by BeginClone. When this method returns, none of the XMPStorageUnits in the source XMPDraft should be copied to the destination Draft.

Inputs

key	XMPDraftKey which identifies the transaction.

Outputs

None.

Exceptions Signalled

kXMPErrInvalidDraftKey	Invalid Clone.

Pre conditions

None.

Post conditions

None.

BeginClone

Function Prototype

```
XMPDraftKey BeginClone(
        XMPCloneKind kind);
```

Protocol

Object Management

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Begins a transaction on a data transfer between one XMPDraft to another. Any XMPStorageUnit::CloneTo called between BeginClone and EndClone is considered part of the transaction.
The returned XMPDraftKey is used for CloneTo, EndClone and AbortClone to ensure thread safety.

Inputs

kind	The kind of clone operation being performed.

Outputs

<return>	Key to ensure thread safety.

Exceptions Signalled

kXMPErrCloningInProgress	Another Clone has started already.

Pre conditions

None.

Post conditions

None.

ChangedFromPrev

This method is only of interest to Container Application developers.

Function Prototype

```
XMPBoolean  ChangedFromPrev();
```

Protocol

Draft Manipulation

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Indicate whether or not this Draft is 'empty', i.e. has any changes been made between this Draft and the Draft immediately below this Draft.

Inputs

none

Outputs

<return>	whether or not this Draft contains any changes from previous Draft

Exceptions Signalled

kXMPInvalidDraft	Invalid Draft.

Pre conditions

this XMPDraft object does not refere to the base draft of the document.

Post conditions

None.

CreateFrame

Function Prototype

```
XMPFrame* CreateFrame(
        XMPFrame* containingFrame,
        XMPShape*  frameShape,
        XMPPart*  part,
        XMPULong  frameGroup,
        XMPBoolean  isOverlaid);
```

Protocol

Creation

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Creates a new Frame in this Draft, constructs a XMPFrame object associated with it, and returns the XMPFrame object.

Inputs

containingFrame	see XMPFrame documentaiont
frameShape	

part
frameGroup
isOverlaid

Outputs

<return>	new XMPFrame object

Exceptions Signalled

kXMPErrCannotCreateFrame	Cannot create XMPFrame object.

Pre conditions

Exclusive-write or Shared-write permissions on this Draft.

Post conditions

This Draft is marked dirty.

CreateLinkSource

Function Prototype

```
XMPLinkSource* CreateLinkSource(
        XMPPart* part);
```

Protocol

Creation

Protection

Public. Called by parts when creating the source of a link, typically in their CreateLink method.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Create a new XMPLinkSource object in this draft. Since all XMPLinkSource objects have a unique companion XMPLink object, this method fails if the companion object cannot be created.

The reference count of the object is incremented, so the client should call the object's Release method when finished using the object.

Inputs

part	The part containing the source content of the link.

Outputs

<return>	new XMPLinkSource object

Exceptions Signalled

kXMPErrCannotCreateLink	Cannot create the XMPLinkSource or companion XMPLink object.

Pre conditions

Exclusive-write or Shared-write permissions on this draft.

Post conditions

The returned instance is a fully functional XMPLinkSource object.
This draft is marked dirty.

CreateLinkSpec

Function Prototype

```
XMPLinkSpec* CreateLinkSpec (
        XMPPart* part,
        XMPPtr data,
        XMPULong size);
```

Protocol

Creation

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Create a link spec for content in the argument part. This draft guarantees that a subsequent call to XMPDraft::GetLink with the returned object as the link spec argument will resolve to the argument part. The XMPLinkSpec object returned contains a copy of the argument data. LinkSpecs become invalid when the document creating the link spec is closed.

The data parameter can point to any data which the part needs to create the specified link. An object specifier is recommended but not required.

Inputs

part	The part object creating this link spec.
data	Arbitrary data for private use of the part in XMPPart::CreateLink().
length	Size of the data.

Outputs

<result>	A new LinkSpec object.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

This Draft is marked dirty.

CreatePart

This method is only of interest to Container Application developers.

Function Prototype

```
XMPPart* CreatePart(
        XMPISOStr partType);
```

Protocol

Creation

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Creates a new Part in this Draft, constructs a XMPPart to refer to the new Part and returns the XMPPart object.

Inputs

partType	Type of Part to create

Outputs

<return>	new XMPPart object

Exceptions Signalled

kXMPErrCannotCreatePart	Cannot create XMPPart object.

Pre conditions

Exclusive-write or Shared-write permissions on this Draft.

Post conditions

This Draft is marked dirty.

CreateStorageUnit

Function Prototype

```
XMPStorageUnit* CreateStorageUnit();
```

Protocol

Creation

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Creates a new Storage Unit in this Draft, constructs a XMPStorageUnit to refer to it and returns the XMPStorageUnit object.

Inputs

none

Outputs

<return> new XMPStorageUnit object

Exceptions Signalled

none

Pre conditions

Exclusive-write or Shared-write permissions on this Draft.

Post conditions

This Draft is marked dirty.

EndClone

Function Prototype

```
void EndClone(
        XMPDraftKey key);
```

Protocol

Object Management

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

This function is used to end the transaction started by BeginClone. When this call returns, all the XMPStorageUnits whose CloneTo method have been called should be copied to the destination XMPDraft.

Inputs

key XMPDraftKey which identifies the transaction.

Outputs

None.

Exceptions Signalled

kXMPErrInvalidDraftKey

Pre conditions

None.

Post conditions

All the XMPStorageUnits whose CloneTo method have been called should be copied to the destination XMPDraft.

Externalize

This method is only of interest to Container Application developers.

Function Prototype

```
XMPDraft*  Externalize();
```

Protocol

Externalization

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Externalizes any internal structures of this XMPDraft objects and also calls Externalize() on all the XMPStorageUnits and XMPPersistentObjects (and derived objects) instantiated through this draft.

Inputs

none

Outputs

<return> this XMPDraft object.

Exceptions Signalled

none

Pre conditions

Exclusive-write or Shared-write permissions on this Draft.

Post conditions

All XMPStorageUnits and XMPPersistentObjects (and derived objects) have been asked to Externalize.

GetDocument

This method is only of interest to Container Application developers.

Function Prototype

```
XMPDocument* GetDocument();
```

Protocol

Getter

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the XMPDocument object from which this XMPDraft object is created.

Inputs

none

Outputs

<return> fDocument

Exceptions Signalled

none

Pre conditions

None.

Post conditions

None.

GetDraftProperties

Function Prototype

```
XMPStorageUnit* GetDraftProperties();
```

Protocol

Meta-information

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns a XMPStorageUnit object in which the Draft uses to store Draft Properties.

Inputs

none

Outputs

<return>	this Draft's Draft Properties Storage Unit

Exceptions Signalled

kXMPErrNoDraftProperties	Draft Properties Storage Unit cannot be created.

Pre conditions

None.

Post conditions

None.

GetFrame

Function Prototype

```
XMPFrame* GetFrame(
        XMPStorageUnitID id);
```

Protocol

Getter

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns a XMPFrame object which refers to a Frame with the given id.

Inputs

id	ID of the desired frame

Outputs

<return>	A fully functional XMPFrame object

Exceptions Signalled

kXMPErrCannotGetFrame	Cannot create XMPFrame object.

Pre conditions

ID refers to a valid Frame.

Post conditions

The return value is a fully functional XMPFrame object with a reference count >= 1.

GetID

This method is only of interest to Container Application developers.

Function Prototype

```
XMPDraftID  GetID();
```

Protocol

Getter

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the ID associated with this Draft.

Inputs

none

Outputs

<return>	ID of this Draft.

Exceptions Signalled

none

Pre conditions

None.

Post conditions

None.

GetLink

Function Prototype

```
XMPLink* GetLink(
        XMPStorageUnitID id,
        XMPLinkSpec* linkSpec);
```

Protocol

Getter

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns a XMPLink object corresponding to the specification. If id is not 0, linkSpec is ignored. Otherwise, linkSpec is used to retrieve the Link.

Inputs

id	ID of the desired link
theLinkSpec	a link spec from which a link is to be resolved or constructed

Outputs

<return>	a fully functional XMPLink object

Exceptions Signalled

kXMPErrCannotGetLink	Cannot create XMPLink object.

Pre conditions

id or linkSpec represents a valid Link on this Draft.

Post conditions

The return value is a fully functional XMPLink object with a reference count >= 1.

GetLinkSource

Function Prototype

```
XMPLinkSource* GetLinkSource(
        XMPStorageUnitID id);
```

Protocol

Getter

Protection

Public. Called by parts maintaining the source of a link, typically in their InitFromStorage method.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the XMPLinkSource object identified by the argument id. Since all XMPLinkSource objects have a unique companion XMPLink object, this method fails if the companion object does not exist and cannot be created.

The reference count of the object is incremented, so the client should call the object's Release method when finished using the object.

Inputs

id	ID of the desired link

Outputs

<return>	a fully functional XMPLink object

Exceptions Signalled

kXMPErrCannotGetLink	Cannot create XMPLink object.

Pre conditions

id represents a valid XMPLinkSource object in this draft.

Post conditions

The return value is a fully functional XMPLinkSource object with a reference count >= 1.

GetName

Function Prototype

```
XMPDraftName* GetName();
```

Protocol

Getter

Protection

Public. Even though a Part can call this method, it should not need to know the name of its Draft.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the name of this Draft. If this Draft does not have a name, kXMPNULL is returned.

Inputs

none

Outputs

<return>	Name of this Draft.

Exceptions Signalled

none

Pre conditions

None.

Post conditions

None.

GetPart

Function Prototype

```
XMPPart* GetPart(
        XMPStorageUnitID id);
```

Protocol

Part

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns a XMPPart object which refers to the Part with the given id.

Inputs

id	the id of the desired part

Outputs

<return>	a fully functional XMPPart object

Exceptions Signalled

kXMPErrCannotGetPart	Cannot create XMPPart object.

Pre conditions

id represents a valid Part in this Draft.

Post conditions

The return value is a fully functional XMPPart object with a reference count >= 1.

GetPermissions

Function Prototype

```
XMPDraftPermissions    GetPermissions();
```

Protocol

Getter

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the Permissions on the Draft.

Inputs

None.

Outputs

<return>	Permissions on the Draft.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetStorageUnit

Function Prototype

```
XMPStorageUnit* GetStorageUnit(
        XMPStorageUnitID id);
```

Protocol

Getter

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns a XMPStorageUnit object which refers to a Storage Unit with the given id.

Inputs

id	the id of the storage unit des9ired

Outputs

<return>	a fully functional XMPStorageUnit object

Exceptions Signalled

none

Pre conditions

id represents a valid Storage Unit in this Draft.

Post conditions

The return value is a fully functional XMPStorageUnit object with a reference count >= 1.

IncrementRefCount

This method is only of interest to Container Application developers.

Function Prototype

```
void IncrementRefCount();
```

Protocol

Object Storage

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Increments the reference count of this XMPDraft object.

Inputs

none

Outputs

none

Exceptions Signalled

none

Pre conditions

None.

Post conditions

The reference count of this object is incremented by 1.

InitDraft

This method is only of interest to Container Application developers.

Function Prototype

```
void InitDraft(
        XMPDocument* document,
        XMPDraftID id,
        XMPDraftPermissions perms);
```

Protocol

Constructor

Protection

Public. Private within the Storage System. This method should only be called by XMPDocument::CreateDraft or XMPDocument::GetDraft.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Initializes this XMPDraft object with the supplied parameters.

Inputs

document	Document from which this XMPDraft is created.
id	ID of this Draft.
perms	Access permissions for this XMPDraft.

Outputs

none

Exceptions Signalled

none

Pre conditions

None.

Post conditions

this is an initialized XMPDraft object.

Purge

This method is only of interest to Container Application developers.

Function Prototype

```
XMPSize Purge(
        XMPSize size);
```

Protocol

Memory Management

Protection

Public. Private within the Storage System. This method should only be called by XMPDocument::Purge.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Purge memory from ephemeral store until size bytes (not necessarily contiguous) have been freed up.

Inputs

size	Number of bytes to purge

Outputs

<return>	Number of bytes actually purged

Exceptions Signalled

none

Pre conditions

None.

Post conditions

Either size bytes (not necessarily contiguous) are free in the default heap or Purge() has been called on every XMPStorageUnit or XMPPersistentObjects (and their derived objects) associated with this XMPDraft object.

Release

This method is only of interest to Container Application developers.

Function Prototype

```
XMPDocument* Release();
```

Protocol

Reference counting

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Releases this XMPDraft object.

Inputs

none

Outputs

<return>	fDocument

Exceptions Signalled

none

Pre conditions

None.

Post conditions

this XMPDraft object is no longer a valid XMPDraft object.

ReleaseFrame

This method is only of interest to Container Application developers.

Function Prototype

```
void  ReleaseFrame(
        XMPFrame* frame);
```

Protocol

Reference counting

Protection

Public. Private between XMPDraft and XMPFrame. XMPFrame::Release calls this method when its refCount goes down to 0.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Releases the XMPFrame object.

Inputs

frame	XMPFrame object to be released

Outputs

none

Exceptions Signalled

kXMPErrInvalidRefCount	RefCount of object is not 0.

Pre conditions

frame is a XMPFrame object with a refCount of 0.

Post conditions

None.

ReleaseLink

This method is only of interest to Container Application developers.

Function Prototype

```
void  ReleaseLink(
        XMPLink* link);
```

Protocol

Reference counting

Protection

Public. Private between XMPDraft and XMPLink. XMPLink::Release calls this method when its refCount goes down to 0.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Releases the XMPLink object.

Inputs

link	XMPLink object to be released

Outputs

none

Exceptions Signalled

kXMPErrInvalidRefCount	Refcount of link is not 0.

Pre conditions

link is a XMPLink object with a refCount of 0.

Post conditions

None.

ReleasePart

Function Prototype

```
void  ReleasePart(
        XMPPart* part);
```

Protocol

Reference counting

Protection

Public. Private between XMPPart and XMPDraft. XMPPart::Release calls this method when its refCount goes down to 0.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Releases the given XMPPart object.

Inputs

part	XMPPart object to be released

Outputs

none

Exceptions Signalled

kXMPErrInvalidRefCount	RefCount of part is not 0.

Pre conditions

part is a XMPPart object with a refCount of 0.

Post conditions

None.

ReleaseStorageUnit

This method is only of interest to Container Application developers.

Function Prototype

```
XMPDraft*  ReleaseStorageUnit(
        XMPStorageUnit* storageUnit);
```

Protocol

Reference counting

Protection

Private. Private within the Storage System. XMPStorageUnit::Release calls this method when its refCount goes down to 0.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Decrement the reference count for the given XMPStorageUnit object.

Inputs

storageUnit	XMPStorageUnit object to be released

Outputs

<return>	this

Exceptions Signalled

kXMPErrInvalidRefCount	Refcount of storageUnit is not 0.

Pre conditions

storageUnit is a XMPStorageUnit object with a refCount of 0.

Post conditions

None.

RemoveChanges

This method is only of interest to Container Application developers.

Function Prototype

```
XMPDraft*  RemoveChanges();
```

Protocol

Draft Management

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Removes any changes been made between this Draft and the Draft immediately below it.

Inputs

none

Outputs

<return>	this

Exceptions Signalled

none

Pre conditions

this Draft is "gotten" with kDPExclusiveWrite perms.

Post conditions

this Draft is 'empty'.

RemoveFrame

This method is only of interest to Container Application developers.

Function Prototype

```
void  RemoveFrame(
        XMPFrame* frame);
```

Protocol

Destruction

Protection

Public. Private between XMPDraft and XMPFrame. This method should only be called by XMPFrame::Remove.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Removes the frame persistently from the Draft and destroys the XMPFrame associated with it.

Inputs

frame	the frame to be removed

Outputs

none

Exceptions Signalled

kXMPErrInvalidStorageUnit	Invalid XMPFrame with no XMPStorageUnit associated with it.
kXMPErrInvalidRefCount	RefCount of frame is not 1.

Pre conditions

The refcount of the given XMPFrame object is 1.

The permissions of this XMPDraft are kDPExclusiveWrite or kDPSharedWrite.

Post conditions

frame is no longer a valid XMPFrame object.

this Draft no longer contains a Frame with the corresponding id.

RemoveFromDocument

This method is only of interest to Container Application developers.

Function Prototype

```
void   RemoveFromDocument();
```

Protocol

Draft Management

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Removes this Draft persistently from its Document and destroys the XMPDraft associated with it.

Inputs

none

Outputs

none

Exceptions Signalled

none

Pre conditions

This XMPDraft is gotten with write permissions.
This Draft is 'empty'.
This Draft is not the Base Draft.

Post conditions

this object is no longer a valid XMPDraft.
The Draft which used to be directly above this one is now directly above the Draft which used to be below this one.

RemoveLink

This method is only of interest to Container Application developers.

Function Prototype

```
void  RemoveLink(
        XMPLink* link);
```

Protocol

Destruction

Protection

Public. Private between XMPDraft and XMPLink. This method should only be called by XMPLink::Remove.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Removes the Link referred to by link from this Draft.

Inputs

link	Link to be removed

Outputs

none

Exceptions Signalled

kXMPErrInvalidRefCount	Refcount of link is not 1.
kXMPErrInvalidStorageUnit	Invalid XMPLink with no XMPStorageUnit associated with it.

Pre conditions

The refcount of the given XMPLink object is 1.
The permissions of this XMPDraft are kDPExclusiveWrite or kDPSharedWrite.

Post conditions

link is no longer a valid XMPLink object
This Draft no longer contains a Link with the corresponding id.

RemoveLinkSource

This method is only of interest to Container Application developers.

Function Prototype

```
void RemoveLinkSource(
        XMPLinkSource* link);
```

Protocol

Destruction

Protection

Public. Private for use only by XMPLinkSource.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Removes the XMPLinkSource referred to by link from this draft.

Inputs

link	Link to be removed

Outputs

None.

Exceptions Signalled

kXMPErrInvalidRefCount	Refcount of link is not 1.
kXMPErrInvalidStorageUnit	Invalid XMPLinkSource with no XMPStorageUnit associated with it.

Pre conditions

The refcount of the link argument is 1.
The permissions of this draft are kDPExclusiveWrite or kDPSharedWrite.

Post conditions

link is no longer a valid XMPLinkSource object.
This draft no longer contains a XMPLinkSource object with the corresponding id.

RemovePart

Function Prototype

```
void RemovePart(
        XMPPart* part);
```

Protocol

Destruction

Protection

Public. Private between XMPDraft and XMPPart. This method should only be called by XMPPart::Remove.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Removes from this Draft the Part referred to by the given XMPPart object.

Inputs

part	Part to be removed

Outputs

none

Exceptions Signalled

kXMPErrInvalidRefCount	Refcount of part is not 1.
kXMPErrInvalidStorageUnit	Invalid XMPPart with no XMPStorageUnit associated with it.

Pre conditions

The refcount of the given XMPPart object is 1.
The permissions of this XMPDraft object are kDPExclusiveWrite or kDPSharedWrite.

Post conditions

part is no longer a valid XMPPart object.
This Draft no longer contains a Part with the corresponding id.

RemoveStorageUnit

This method is only of interest to Container Application developers.

Function Prototype

```
void   RemoveStorageUnit(
           XMPStorageUnit* storageUnit);
```

Protocol

Destruction

Protection

Public. Private with the Storage System. This method should only be called by XMPStorageUnit::Remove.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Removes from this Draft the Storage Unit referred to by the XMPStorageUnit object.

Inputs

storageUnit	Storage unit to be removed

Outputs

none

Exceptions Signalled

kXMPErrInvalidStorageUnit	Invalid storageUnit.

Pre conditions

The refcount of the given XMPStorageUnit object is 1.
The permissions of this XMPDraft are kDPExclusiveWrite or kDPSharedWrite.

Post conditions

storageUnit is no longer a valid XMPStorageUnit object.
This Draft no longer contains a storage unit with the corresponding id.

SaveToAPrevious

This method is only of interest to Container Application developers.

Function Prototype

```
XMPDraft*  SaveToAPrevious(
               XMPDraft* to);
```

Protocol

Object Management

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

same as XMPDocument::SaveToAPrevDraft(this,to).

Inputs

to	draft to move changes to

Outputs

<return>	this

Exceptions Signalled

none

Pre conditions

None.

Post conditions

None.

SetChangedFromPrev

Function Prototype

```
void SetChangedFromPrev();
```

Protocol

Externalization

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Marks this Draft dirty (i.e., ChangedFromPrev).
Parts can call this method to mark the Draft dirty. The dirty state of the Draft will trigger externalization when the Draft is closed.
Note that a Draft cannot be marked clean after it has been marked dirty.
However, RemoveChanges can be called to remove all the changes in this Draft.

Inputs

None

Outputs

None

Exceptions Signalled

None

Pre conditions

None.

Post conditions

None.

SetName

This method is only of interest to Container Application developers.

Function Prototype

```
void   SetName(
           XMPDraftName name);
```

Protocol

Setter

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Sets the name of this Draft.

Inputs

name	New name of this draft

Outputs

none

Exceptions Signalled

none

Pre conditions

name represents a valid XMPDraftName.

Post conditions

None.

XMPDragAndDrop

Basic Class Documentation

Platform implementors should implement this class to provide the functionality of a OpenDoc Drag and Drop mechanism.

This class depends on system services provided by the platform. They include a clipboard-like service where data can be transfered within a process or between processes, and a system-wide mouse tracking service which can notify a process about the location (i.e., whether the mouse is entering, leaving or in a window of the process) and the state of the mouse (i.e., whether there is a mouse-down or mouse-up).

This class provides the basic mechanism for dragging an object within a part, between parts, and between an OpenDoc document and a non-OpenDoc application.

At the process' startup time, a XMPDragAndDrop object is instantiated and stored with the XMPSession object.

Theory of Operation

The main client of this class is XMPPart. Any part can initiate a drag. Whenever a part needs to use the Drag-and-Drop mechanism, it can acquire the XMPDragAndDrop object through the XMPSession object.

When a mouse-down is detected within a frame, the part has the choice of initiating a drag. This involves acquiring the drag focus (to be thread-safe), getting the XMPDragAndDrop object from the XMPSession, copying the data for the dragged object to the XMPDragAndDrop object (through a XMPStorageUnit supplied by the XMPDragAndDrop object) and starting a drag action. Once the drag is initiated, the XMPDragAndDrop object will notify the Frame when the mouse passes over it. If the mouse is released over a Frame, it will be notified of the drop and it can retrieve the data for the dragged object from the supplied XMPStorageUnit object.

As mentioned above, copying data to and from the XMPDragAndDrop object is done through a XMPStorageUnit.

Invariants Maintained by Class

Note that the client of this class should never cache the XMPDragAndDrop object nor its XMPStorageUnit. Instead, it should always get these objects through XMPSession and XMPDragAndDrop APIs whenever they are needed.

Other Persistent Properties

Member Functions

XMPDragAndDrop

This method is only of interest to Container Application developers.

Function Prototype

```
XMPDragAndDrop();
```

Protocol

Initialization; Drag-and-drop

Protection

Public. This method is called once by XMPSession::InitSession.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

This function is the constructor of the XMPDragAndDrop object. It is called once at the process' startup time.

Inputs

None

Outputs

None

Exceptions Signalled

None

Pre conditions

None.

Post conditions

None.

~XMPDragAndDrop

This method is only of interest to Container Application developers.

Function Prototype

```
~XMPDragAndDrop();
```

Protocol

Cleanup.

Protection

Public. This method should only be called by XMPSession::~XMPSession.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

This function is the destructor for the XMPDragAndDrop object. It is called once when OpenDoc quits. It should close all the required system services that it has initialized.

Inputs

None

Outputs

None

Exceptions Signalled

None

Pre conditions

None.

Post conditions

None.

Clear

Function Prototype

```
void Clear();
```

Protocol

Embedding, Drag-and-Drop.

Protection

Public. None.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

This function removes all the user data stored in the XMPDragAndDrop object. This is called by a Part before it is going to put data into this object for transfer.

Inputs

None

Outputs

None

Exceptions Signalled

None

Pre conditions

None.

Post conditions

The XMPStorageUnit should be empty.

GetDragAttributes

This method is Macintosh specific.

Function Prototype

```
XMPULong GetDragAttributes();
```

Protocol

Embedding, Drag-and-Drop. (Macintosh)

Protection

Public. This is a Macintosh specific function used to give the parts additional information about a drag.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

This function returns a flag word containing info about the current drag. The following flags are used:

```
#define kXMPdragHasLeftSourceFrame      0x00000001
#define kXMPdragIsInSourcePart          0x00000002
#define kXMPdragIsInSourceFrame         0x00000004
```

If no drag is currently in progress, the result is 0. The contents of the flag word are intended to help the part determine if user feedback is neccessary and to help the part efficiently handle the contents of the drag.

Inputs

None

Outputs

XMPULong — Flag word contianing info on the current drag.

Exceptions Signalled

None.

Pre conditions

A drag must be in progress.

Post conditions

None.

GetDragReference

This method is Macintosh specific.

Function Prototype

```
DragReference GetDragReference();
```

Protocol

Embedding, Drag-and-Drop. (Macintosh)

Protection

Public. This is a Macintosh specific function that returns the paltform specific DragReference associated with the current drag.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

This function returns the Drag Manager dragReference associated with the current drag. If no drag is under way, 0 is returned. Parts should use this

function to enable direct calls to the Drag Manager utilities to do window highliting.

Inputs

None

Outputs

DragReference	Paltform specific drag reference number

Exceptions Signalled

None.

Pre conditions

A drag must be in progess.

Post conditions

None.

GetStorageUnit

Function Prototype

```
XMPStorageUnit* GetStorageUnit();
```

Protocol

Embedding, Drag-and-Drop.

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

This function returns an XMPStorageUnit object.
The caller should copy data to the XMPDragAndDrop object through the XMPStorageUnit object.
To ensure that the XMPStorageUnit is empty, Clear() should be called immediately after GetStorageUnit() is called.

Inputs

None

Outputs

<return>	An empty XMPStorageUnit object for the caller to copy the data to.

Exceptions Signalled

None

Pre conditions

None.

Post conditions

An empty XMPStorageUnit is returned.

InitDragAndDrop

This method is only of interest to Container Application developers.

Function Prototype

```
void InitDragAndDrop();
```

Protocol

Initialization

Protection

Public. This method should only be called by XMPSession::InitSession.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

It initializes all the required system services which include a drag mechanism and a data transfer mechanism.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

this is an initialized XMPDragAndDrop object.

Purge

This method is only of interest to Container Application developers.

Function Prototype

```
XMPSize Purge(
        XMPSize size);
```

Protocol

Memory Management

Protection

Public. This method is only called by XMPSession::Purge.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Purges all the ephemeral structures in this object.

Inputs

size	Number of bytes to purge.

Outputs

<return>	Number of bytes actually purged.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

StartDrag

Function Prototype

```
XMPDropResult StartDrag(
        XMPFrame* srcFrame,
        XMPValueType imageType,
        XMPPtr image,
        XMPPart** destPart,
        XMPPtr refCon);
```

Protocol

Embedding, Drag-and-Drop.

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

This function initiates a drag. It shows the image supplied by the caller and allows the user to drag this image on the screen. Whenever the mouse passes over a Droppable Frame, it will notify the frame about the mouse location and the mouse state. If a drop occurs over a Droppable Frame, it will notify the frame.

If the call is implemented synchronously, it will also return the destination part.

Inputs

srcFrame	Frame in which the drag is initiated.
imageType	Type of the image being dragged.
image	Data of the image being dragged.
refCon	Extra information needed for dragging (mainly for platform specific data).

Outputs

<return>	kDropSucceed means a drop is successful. kDropFail means the drop is unsuccessful. kDropUnfinished means the drop is not completed yet.
destPart	If kDropSucceed is returned as the function result, destPart contains the destination part. Otherwise, the content of this parameter is undefined.

Exceptions Signalled

None

Pre conditions

XMPStorageUnit must contain client data from the Part initiating the drag.

Post conditions

None.

XMPDragItemIterator

Basic Class Documentation

XMPDragItemIterator is a companion of XMPDragAndDrop. Given the state of the XMPDragAndDrop object, XMPDragItemIterator allows a Part to iterate over all the XMPStorageUnits in the drag with a loop, using the iterator's First(), Next() and IsNotComplete() methods.

XMPDragItemIterator is implemented together with XMPDragAndDrop by platform implementors.

Theory of Operation

Once a drag is initiated, the XMPDragAndDrop object will notify a Frame when the mouse passes over it. Besides the mouse location, the XMPDragAndDrop object also supplies the Part with a XMPDragItemIterator. If the mouse is released over a Frame, it will be notified of the mouse location and be supplied with a XMPDragItemIterator. In both cases, the Part can use it to iterate over all the Drag Items and decide whether it can accept the Drop.

Invariants Maintained by Class

XMPDragItemIterator maintains a reference to the list of drag items registered with the XMPDragAndDrop object.

Other Persistent Properties

Member Functions

First

Function Prototype

```
XMPStorageUnit* First();
```

Protocol

Embeddeding, Drag and Drop.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the first storage unit in the drag item list.

Inputs

None

Outputs

<return>	The first storage unit in the drag item list. kXMPNULL if none.

Exceptions Signalled

None

Pre conditions

A constructed object of this class.

Post conditions

Result contains the first storage unit.

IsNotComplete

Function Prototype

```
XMPBoolean IsNotComplete();
```

Protocol

Embeddeding, Drag and Drop.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns kXMPTrue if there are more storage units, kXMPFalse otherwise.

Inputs

None

Outputs

<return>	kXMPTrue, if there are more windows

Exceptions Signalled

None

Pre conditions

A constructed object of this class.

Post conditions

Result contains kXMPTrue if there are more storage units, kXMPFalse otherwise.

Next

Function Prototype

```
XMPStorageUnit* Next();
```

Protocol

Embeddeding, Drag and Drop.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the next storage unit in the drag item list.

Inputs

None

Outputs

<return>	The next storage unit in the drag item list. kXMPNULL if none.

Exceptions Signalled

None

Pre conditions

First() has been called.

Post conditions

Result contains the next storage unit.

XMPEmbeddedFramesIterator

Basic Class Documentation

An XMPEmbeddedFramesIterator is used to iterate the embedded frames of a containing part.

This class has no base class. It is typically a friend of the part class whose frames it iterates.

This class is implemented by a part developer along with a containing part class.

Theory of Operation

XMPEmbeddedFramesIterator instances are created by their containing parts. A client calls XMPPart::CreateEmbeddedFramesIterator to obtain an iterator for a part's embedded frames.

Invariants Maintained by Class

Varies according to implementation.

Other Persistent Properties

Member Functions

XMPExtension

Basic Class Documentation

XMPExtension is the abstract base class from which object extension interfaces are derived. The base class itself has minimal behavior. It knows which object it is an extension of, and how to release resources in itself and in its object. Further behavior should be implemented in derived classes.

This class has no base class. Extensions will typically share friendship with their base object class.

Both platform vendors and part developers will use extensions to extend standard interfaces.

Theory of Operation

XMPExtension is the abstract base class from which object extension interfaces are derived. The base class itself has minimal behavior. It knows which object it is an extension of, and how to release resources in itself and in its object. Further behavior should be implemented in derived classes.

Invariants Maintained by Class

fBase contains a valid base object for this extension.

Other Persistent Properties

Member Functions

XMPExtension

Function Prototype

```
XMPExtension(
        XMPObject* base);
```

Protocol

Extensions

Protection

Protected.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Construct a new extension instance.

Inputs

base	The base object for this extension.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

fBase holds base object for this extension.

~XMPExtension

Function Prototype

```
~XMPExtension();
```

Protocol

Extensions

Protection

Protected.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Deallocate this instance and its storage.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

This instance must have been previously "released".

Post conditions

This instance is no longer usable by clients.

GetBase

Function Prototype

```
XMPObject* GetBase();
```

Protocol

Extensions

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return the object of which this instance is an extension.

Inputs

None.

Outputs

<return>	The base of this extension.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

InitExtension

Function Prototype

```
void InitExtension(
        XMPObject* base);
```

Protocol

Extensions

Protection

Protected.

Override policy

Derived class **must** override.

Derived class **must** call base class behavior, before derived class behavior.

Basic operation

Prepare this object for use.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

fBase is a valid base object

Post conditions

This object is ready for use by clients.

Release

Function Prototype

```
void Release();
```

Protocol

Extensions

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Tell my base object to release connection to me. Prepare to be deleted.

Inputs

None.

Outputs

None.

Exceptions Signalled

kXMPErrInvalidBase	Not a valid base object.

Pre conditions

fBase is a valid base object.

Post conditions

This extension is no longer usable.

XMPFacet

Basic Class Documentation

Ancestors: XMPFacet -> XMPObject

Part developers will not subclass XMPFacet. Platform implementors can subclass if needed, but that should not normally be necessary.

XMPFacet objects hold non-persistent information about the layout of a document. They are used to place a frame on a canvas for display and event dispatching. There will be one facet for each place a frame is visible in a window. Facets may exist for frames which have been previously visible, such as those that have been scrolled out of view, or may be eagerly created for frames which are expected to become visible soon. Facets which are not currently visible in a window are liable to be purged from memory in a low-memory condition.

Parts create facets for each place they display an embedded frame. In most cases, a containing part will only have one facet for an embedded frame in each facet of the containing part's display frame. But in some cases, a containing part will want to display the same embedded frame in several places. This can easily be accomplished by creating one facet for each place the containing part wants to display the embedded frame.

Facets are organized hierarchically. There is one root facet for each window or printing page, and all other facets in that window descend from it. All facets for the same frame must be contained within facets of their frame's containing frame.

There can be mulitple facets displaying the same frame. All parts should support multiple facets on a frame. In most cases all facets on the same frame will display their contents identically. For cases where the part wishes to display facets differently, it may store "partInfo" data in the facets to distinguish them.

A facet holds some extra information about the geometry of its frame. In addition to the frame's own frame shape, it also maintains a clip shape and an active shape. The clip shape describes the area in which the facet's part may display itself; it is controlled by the containing part. The active shape describes where the facet's part is sensitive to geometry-based UI events such as mouse clicks. The embedded part controls the active shape. Both shapes are represented in the frame's coordinate space. A facet also has an external transform, which describes where it is located within its containing part's coordinate space.

Any facet can have its own canvas. In that case, the facet and all its children will display themselves on that canvas. This can be used to do off-screen double-buffering, image combination, etc. Either the facet's part or the facet's containing part can decide to move the facet to its own canvas. Whichever part makes the decision must create the canvas and register as its owner. That part is then responsible for moving the data from the facet's canvas to the parent canvas.

A facet maintains some extra state information used for display and event dispatching. The "isSelected" flag indicates whether the facet is selected, so the dispatcher can determine whether to dispatch events into it. The "highlight" field indicates if the facet is part of a larger selection, and if so how its part needs to alter its imaging for selection highlighting.

Facets can be thought of as describing the layout environment of a frame. Since a facet can exist simultaneously in both a window and an offscreen canvas, there are sometimes two environments to consider. The aggregate clip shape and aggregate transforms are both distinguished by which environment, canvas or window, the part wants to display in. In most cases, the part will display in the canvas environment. But for real-time interaction like rubber-banding or dragging, the part may want to display directly in the window. In those cases, it should use the aggregates for the window environment.

Theory of Operation

Invariants Maintained by Class

An XMPFacet object always has a frame, and that frame cannot be changed during the lifetime of the facet. The clip shape and external transform must always be valid. Only the root facet of a window may have its window field set. Non-root facets must have a containing facet.

Other Persistent Properties

Member Functions

XMPFacet

Function Prototype

XMPFacet();

Protocol

None.

Protection

Public.

Override policy

Derived class **must** override.

Derived class **must** call base class behavior, before derived class behavior.

Basic operation

Class constructor.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

Must call InitFacet before returned instance is ready to use.

~XMPFacet

Function Prototype

~XMPFacet();

Protocol

None.

Protection

Public.

Override policy

Derived class **must** override.

Derived class **must** call base class behavior, after derived class behavior.

Basic operation

Class destructor.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

ActiveBorderContainsPoint

Function Prototype

```
XMPBoolean ActiveBorderContainsPoint(
        XMPPoint point);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return whether the point is within the active border adornment for this facet.

Inputs

point	The location to test. In window coordinates.

Outputs

<return>	True if the point is within the facet's border shape, false otherwise.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

ChangeActiveShape

Function Prototype

```
void ChangeActiveShape(
        XMPShape* activeShape);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Set the facet's active shape, and notify UI subsystem dependents. Only the facet's part should change its active shape.

Inputs

activeShape	The new active shape for the facet.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

ChangeClipShape

Function Prototype

```
void ChangeClipShape(
        XMPShape* clipShape);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Set the facet's clip shape, and notify the facet's part of the change via Part::ClipShapeChanged(). Invalidate any cached aggregate clip shape. Only the facet's containing part (or window) should change its clip shape.

Inputs

clipShape	The new clip shape for the facet.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

ChangeExternalTransform

Function Prototype

```
void ChangeExternalTransform(
        XMPTransform* transform);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Set the external transform of the facet, and notify the facet's part of the change via Part::ExternalTransformChanged(). Only a facet's containing part should change its external transform.

Inputs

transform	The new external transform for the facet.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

ChangeHighlight

Function Prototype

```
void ChangeHighlight(
        XMPHighlight highlight);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Set the highlight state of this facet, and notify its part via Part::HighlightChanged().

Inputs

highlight	The new highlight state of the facet. One of kXMPNoHighlight, kXMPFullHighlight, kXMPDimHighlight.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

ContainsPoint

Function Prototype

```
XMPBoolean ContainsPoint(
        XMPPoint point);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return whether the point is within the facet. Transform the point to frame coordinates and test versus the intersection of the clip and active shapes.

Inputs

point	The location to test. In window coordinates.

Outputs

<return>	True if the point is within the facet's shape, false otherwise.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

CreateEmbeddedFacet

Function Prototype

```
XMPFacet* CreateEmbeddedFacet(
        XMPFrame* frame,
        XMPShape* clipShape,
        XMPTransform* externalTransform,
        XMPFacet* siblingFacet,
        XMPFramePosition position);
```

Protocol

Facets

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Asks a facet to create an embedded facet. This should be requested by the facet's part when it needs to create a facet for one of its embedded frames. The new facet's part will be notified that a new facet has been added to its frame [XMPPart::FacetAdded(facet)].

Inputs

frame	The frame the new facet will display.
clipShape	The initial clip shape for the new facet.
externalTransform	The initial external transform for the new facet.
siblingFacet	An existing child facet of the receiver. May be kXMPNULL.
position	The desired position of the new facet relative to the siblingFacet. Legal values are kXMPFrameBehind or kXMPFrameInFront. If siblingFacet is kXMPNULL, the new facet will be placed at the front or back of all siblings.

Outputs

<return>	The new facet.

Exceptions Signalled

kXMPInvalidFacet	"siblingFacet" is not a child of the receiver.
kXMPErrInvalidPositionCode	"position" is not a valid position code.

Pre conditions

The receiving facet is a facet of the containingFrame of the "frame" parameter. "siblingFacet" is a child facet of the receiving facet.

Post conditions

The returned facet is a child facet of the receiving facet, with clipShape and externalTransform set to the indicated values.

CreateFacetIterator

Function Prototype

```
XMPFacetIterator* CreateFacetIterator(
        XMPTraversalType traversalType,
        XMPSiblingOrder siblingOrder);
```

Protocol

Facets

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Create an iterator for the children facet of the receiver. It is the caller's responsibility to delete the iterator when done using it. The iterator can be used to walk the entire tree of facets from the receiver on down, or just the receiver's immediate children.

Inputs

traversalType	may be kXMPTopDown, kXMPBottomUp, or kXMPChildrenOnly. The first two include the receiver in the traversal.
siblingOrder	May be kXMPFrontToBack or kXMPBackToFront. Controls the order in which facets are visited during traversal.

Outputs

<return>	The new iterator. Caller assumes responsibility for memory.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

New iterator has been created and is ready to use.

Draw

Function Prototype

```
void Draw(
        XMPShape* invalidShape);
```

Protocol

Imaging

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Tell the facet's part to draw itself in invalidShape in this facet. The facet transforms invalidShape into frame coordinates, and passes the request to the part via Part::Draw().

Inputs

invalidShape	The area in which the part should draw itself. In window coordinates.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

DrawActiveBorder

Function Prototype

```
void DrawActiveBorder();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Draw the facet's active border adornment.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

DrawChildren

Function Prototype

```
void DrawChildren(
        XMPShape* invalidShape);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Draw all this facet's child facets which need updating.

Inputs

invalidShape	The shape in which the facets should draw.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

DrawChildrenAlways

Function Prototype

```
void DrawChildrenAlways(
        XMPShape* invalidShape);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Draw all this facet's child facets, whether they need updating or not.

Inputs

invalidShape	The shape in which the facets should draw.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

DrawnIn

Function Prototype

```
void DrawnIn(
        XMPShape* shape);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Notify the facet it has been drawn in via another path than an update operation. Notify the owner of the facet's canvas to update its parent canvas with the altered contents.

Inputs

shape	The shape in which the facet has been drawn. In frame coordinates.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetActiveShape

Function Prototype

```
XMPShape* GetActiveShape();
```

Protocol

Access

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return the facet's active shape. If none has been set, return its frame's frame shape.

Inputs

None.

Outputs

<return>	The facet's active shape.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetAggregateClipShape

Function Prototype

```
XMPShape* GetAggregateClipShape();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return the shape which is the intersection of this facet's clip shape, and all containing facets' clip shapes on this facet's canvas.

Inputs

None.

Outputs

<return>	The facet's aggregate clip shape.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetCanvas

Function Prototype

```
XMPCanvas* GetCanvas();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return the canvas this facet's part should image on. If the facet has no canvas of its own, it will return its containing facet's canvas, recursively.

Inputs

None.

Outputs

<return>	The facet's canvas.

Exceptions Signalled

kXMPErrInvalidCanvas	Neither this facet or any of its parents had a canvas.

Pre conditions

None.

Post conditions

None.

GetClipShape

Function Prototype

XMPShape* GetClipShape();

Protocol

Access

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return the clip shape of the receiver.

Inputs

None.

Outputs

<return>	The facet's clip shape.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetContainingFacet

Function Prototype

XMPFacet* GetContainingFacet();

Protocol

Access

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return the containing facet of the receiver.

Inputs

None.

Outputs

<return>	The containing facet of the receiver. May be kXMPNULL.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetContentTransform

Function Prototype

XMPTransform* GetContentTransform();

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Calculate and return the facet's content transform. This transform is calculated by concatenating the facet's frame's internal transform with this facet's frame transform. This results in a transformation which describes the content coordinate space of this facet on its canvas.

Inputs

None.

Outputs

<return>	The facet's content transform.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetExternalTransform

Function Prototype

```
XMPTransform* GetExternalTransform();
```

Protocol

Access

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return the external transform of the facet.

Inputs

None.

Outputs

<return>	The facet's external transform.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetFrame

Function Prototype

```
XMPFrame* GetFrame();
```

Protocol

Access

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return the frame this facet displays.

Inputs

None.

Outputs

<return>	The facet's frame.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetFrameTransform

Function Prototype

```
XMPTransform* GetFrameTransform();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Calculate and return the facet's frame transform. This transform is calculated by concatenating the facet's external transform with all enclosing facets' internal and external transforms, up to the internal transform of the topmost facet on this facet's canvas. This results in a transformation which describes the frame coordinate space of this facet on its canvas.

Inputs

None.

Outputs

<return>	The facet's frame transform.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetHighlight

Function Prototype

```
XMPHighlight GetHighlight();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return the highlight state of the facet. The facet's part should use this information to draw its contents consistently with the selection highlighting in its containing part.

Inputs

None.

Outputs

<return> The highlight state of the facet.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetPartInfo

Function Prototype

XMPInfoType GetPartInfo();

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return the partInfo data a part has stored in this facet.

Inputs

None.

Outputs

<return> The facet's partInfo.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetWindow

Function Prototype

XMPWindow* GetWindow();

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return the window this facet is displayed in. This value may be kXMPNULL if the facet is a printing facet and does not appear in any window.

Inputs

None.

Outputs

<return> The facet's window.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetWindowAggregateClipShape

Function Prototype

XMPShape* GetWindowAggregateClipShape();

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return the shape which is the intersection of this facet's clip shape, and all containing facets' clip shapes in this facet's window.

Inputs

None.

Outputs

<return> The facet's aggregate clip shape.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetWindowContentTransform

Function Prototype

XMPTransform* GetWindowContentTransform();

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Calculate and return the facet's window content transform. This transform is calculated by concatenating the facet's frame's internal transform with its window frame transform. This results in a transformation which describes the content coordinate space of this facet in its window.

Inputs

None.

Outputs

<return> The facet's window content transform.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetWindowFrameTransform

Function Prototype

XMPTransform* GetWindowFrameTransform();

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Calculate and return the facet's window frame transform. This transform is calculated by concatenating the facet's external transform with all enclosing facets' internal and external transforms. This results in a transformation which describes the frame coordinate space of this facet in its window.

Inputs

None.

Outputs

<return>	The facet's window frame transform.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

HasCanvas

Function Prototype

```
XMPBoolean HasCanvas();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return whether this facets has its own canvas.

Inputs

None.

Outputs

<return>	True if the facet has its own canvas, or false if not.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

InitFacet

Function Prototype

```
void InitFacet(
        XMPFrame* frame,
        XMPShape* clipShape,
        XMPTransform* externalTransform);
```

Protocol

Initialization

Protection

Public.

Override policy

Derived class **must** override.

Derived class **must** call base class behavior, before derived class behavior.

Basic operation

Initialize a newly constructed facet. This MUST be called before the facet can be used.

Inputs

frame	The facet's frame. This value never changes over the life of the facet.
clipShape	The initial clipShape of the facet.
externalTransform	The initial externalTransform of the facet.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

Facet is newly constructed.

Post conditions

Facet can be used for display or event dispatching.

Invalidate

Function Prototype

```
void Invalidate(
        XMPShape* invalidShape);
```

Protocol

Imaging

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Cause the underlying graphics system to mark the area in invalidShape as needing repainting. This area is marked on this facet's canvas, and all parent canvases, so changes in this facet can be reflected in those canvases.

Inputs

invalidShape	The area to mark as invalid. In frame coordinates.

Outputs

None.

Exceptions Signalled

kXMPErrInvalidFacet	Can't find facet for parent canvas.

Pre conditions

None.

Post conditions

None.

InvalidateActiveBorder

Function Prototype

```
void InvalidateActiveBorder();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Mark the area under the facet's active border adornment as needing redrawing. This can be used to erase the active border.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

IsSelected

Function Prototype

```
XMPBoolean IsSelected();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return the facet's isSelected flag. Indicates whether the facet is selected within its container. This information is used by the dispatcher to determine where to send mouse events.

Inputs

None.

Outputs

<return>	Value of the facet's isSelected flag.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

MoveBefore

Function Prototype

```
void MoveBefore(
        XMPFacet* child,
        XMPFacet* sibling);
```

Protocol

Facets

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Move one of a facet's child facets in front of a sibling facet.

Inputs

child	The child facet to reposition.
sibling	The facet to move "child" in front of. If the value is kXMPNULL, "child" is moved to the front of all its siblings.

Outputs

None.

Exceptions Signalled

kXMPErrInvalidFacet	"child" or "sibling" is not a valid child facet of the receiver.

Pre conditions

"child" and "sibling" are both children of the receiver.

Post conditions

"child" has been repositioned in front of "sibling" or all siblings.

MoveBehind

Function Prototype

```
void MoveBehind(
        XMPFacet* child,
        XMPFacet* sibling);
```

Protocol

Facets

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Move one of a facet's child facets behind a sibling facet.

Inputs

child	The child facet to reposition.
sibling	The facet to move "child" behind. If the value is kXMPNULL, "child" is moved behind all its siblings.

Outputs

None.

Exceptions Signalled

kXMPErrInvalidFacet	"child" or "sibling" is not a valid child facet of the receiver.

Pre conditions

"child" and "sibling" are both children of the receiver.

Post conditions

"child" has been repositioned behind "sibling" or all siblings.

RemoveFacet

Function Prototype

```
void RemoveFacet(
        XMPFacet* facet);
```

Protocol

Facets

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Asks a facet to remove one of its child facets. This request should be made by a facet's part prior to removing one of its embedded frames, or optionally when scrolling an embedded frame out of view. The removed facet's part will be notified that a facet has been removed from its frame [XMPPart::FacetRemoved(facet)]. The caller should then delete the removed facet.

Inputs

facet	The child facet to remove.

Outputs

None.

Exceptions Signalled

kXMPErrInvalidFacet	"facet" is not a child of the receiver.

Pre conditions

"facet" must be a child of the receiver.

Post conditions

"facet" should not be used again. Ready to be deleted by the caller.

SetCanvas

Function Prototype

```
void SetCanvas(
        XMPCanvas* canvas);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Set the canvas of this facet. After this operation, this facet and its children should image on that canvas.

Mac only: The facet does NOT assume responsibility for deallocating the canvas' storage.

Inputs

canvas	The new canvas.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

SetPartInfo

Function Prototype

```
void SetPartInfo(
        XMPInfoType partInfo);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Set the partInfo data for this facet. PartInfo should only be set by a facet's part.

Inputs

partInfo	The new partInfo for this facet.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

SetSelected

Function Prototype

```
void SetSelected(
        XMPBoolean isSelected);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Set the facet's isSelected flag.

Inputs

isSelected	The new value of the flag. True if the facet is selected within its containing part.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

SetWindow

Function Prototype

```
void SetWindow(
        XMPWindow* window);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Set the facet's window. Only the root facet of a window should have its window set; its children will inherit this value.

Inputs

window	The window for the facet.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

Update

Function Prototype

```
void Update(
        XMPShape* invalidShape);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Update the facet's canvas in the invalidShape by drawing this facet and any of its children whose clipShape intersects invalidShape.

Inputs

invalidShape	The shape in which the facet's canvas must be repainted. In window coordinates.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

Validate

Function Prototype

```
void Validate(
        XMPShape* validShape);
```

Protocol

Imaging

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Cause the underlying graphics system to mark the area in validShape as no longer needing repainting.

Inputs

validShape	The shape to mark as valid. In frame coordinates.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

XMPFacetIterator

Basic Class Documentation

Ancestors: none

XMPFacetIterator is a companion to class XMPFacet. It allows iteration of all the children of a facet.

Parts, container applications, or other classes may need to iterate facets. All can instatiate this class.

There are three ways to use an XMPFacetIterator, depending on the traversalType specified when the iterator is created. kXMPTopDown starts with the specified facet, and walks depth-first down the sub-tree from that as the root. kXMPBottomUp starts at the leftmost leaf, and walks depth-first up the tree to the root. kXMPChildrenOnly simply iterates only the direct children of the specified facet.

In a top-down traversal, the method SkipChildren() can be used to advance the next item to skip over the current item's children.

Theory of Operation

Invariants Maintained by Class

Other Persistent Properties

Member Functions

XMPFacetIterator

Function Prototype

```
XMPFacetIterator(
        XMPFacet* facet,
        XMPTraversalType traversalType,
        XMPSiblingOrder siblingOrder);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Construct an iterator on facet and its child facets.

Inputs

facet	Root of the sub-tree of facets the iterator will traverse.
traversalType	kXMPTopDown: start at "facet" and walk down depth-first. kXMPBottomUp: start at bottom of tree and walk up to root. kXMPChildrenOnly: only the direct children of "facet".
siblingOrder	kXMPFrontToBack, kXMPBackToFront

Outputs

None.

Exceptions Signalled
None.

Pre conditions
None.

Post conditions
None.

~XMPFacetIterator

Function Prototype
~XMPFacetIterator();

Protocol
None.

Protection
Public.

Override policy
Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation
Destructor.

Inputs
None.

Outputs
None.

Exceptions Signalled
None.

Pre conditions
None.

Post conditions
None.

First

Function Prototype
XMPFacet* First();

Protocol
None.

Protection
Public.

Override policy
Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation
Return the first facet specified by the traversalType and siblingOrder of this iterator.

Inputs
None.

Outputs
<return> The first facet in the sequence.

Exceptions Signalled
None.

Pre conditions
None.

Post conditions
None.

IsNotComplete

Function Prototype

XMPBoolean IsNotComplete();

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return whether the last item returned from First() or Next() was valid.

Inputs

None.

Outputs

<return>	True if iterator is not yet complete, false otherwise.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

Next

Function Prototype

XMPFacet* Next();

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Answer the next facet in the sequence.

Inputs

None.

Outputs

<return>	The next facet.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

SkipChildren

Function Prototype

void SkipChildren();

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

If traversalType is kXMPTopDown, this causes the sequence to skip any children of the current item, if any. For other traversal types, it has no effect.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

XMPFocusModule

Basic Class Documentation

XMPFocusModule is an abstract base class for focus modules, which are the means of extensibility used by the arbitrator. The arbitrator maintains a table of focus modules indexed by focus type.
XMPFocusModule has no base class.
XMPFocusModule is implemented by the platform vendor.
Related classes are XMPArbitrator, XMPFocusSet and XMPFocusSetIterator.

Theory of Operation

Instances of XMPFocusModule are registered with the arbitrator, and associated with one or more focus types. See XMPArbitrator.

Invariants Maintained by Class

Stores the owner frame for one or more foci.

Other Persistent Properties

Member Functions

XMPFocusModule

Function Prototype

XMPFocusModule();

Protocol

Creation and Deletion

Protection

Public. Parts and container apps would typically only create focus modules that they temselves define.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Constructs a focus module. The InitFocusModule method must also be called.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

The object is constructed, but not ready for use until InitFocusModule is called.

~XMPFocusModule

Function Prototype

~XMPFocusModule();

Protocol

Creation and Deletion

Protection

Public. The arbitrator deletes installed focus modules.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Frees the memory allocated by this object.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

A valid instance.

Post conditions

The memory used by this object is freed.

InitFocusModule

Function Prototype

```
void InitFocusModule(
        XMPSession* session);
```

Protocol

Creation and Deletion

Protection

Public. Parts and container apps would typically only create focus modules that they temselves define.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Initializes the focus module.

Inputs

session	The OpenDoc session object.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

A constructed instance.

Post conditions

An initialize and usable instance.

IsFocusExclusive

Function Prototype

```
XMPBoolean IsFocusExclusive(
        XMPTypeToken focus);
```

Protocol

Focus Testing

Protection

Public. Called by Arbitrator. Can be called by parts.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns kXMPTrue if the specified focus is exclusive

Inputs

focus	A token for a registered focus.

Outputs

<return>	kXMPTrue if the specified focus is exclusive, kXMPFalse otherwise.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

No effect on internal state.

XMPFocusOwnerIterator

Basic Class Documentation

XMPFocusOwnerIterator is an abstract class. Focus modules for non-exclusive foci must implement a subclass of this class.
XMPFocusOwnerIterator has no base class.
Related classes are XMPArbitrator and XMPFocusModule

Theory of Operation

XMPFocusOwnerIterator allows a dispatch module or other client to iterate over the owners of a non-exclusive focus in a loop, using the iterator's First(), Next() and IsNotComplete() methods.

Invariants Maintained by Class

The iterator stores a reference to the focus module which stores the list of owners.
Subclasses may add additional invariants.

Other Persistent Properties

Member Functions

First

Function Prototype

```
XMPFrame* First();
```

Protocol

Iteration

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Should return the first frame which is an owner of this non-exclusive focus.

Inputs

None.

Outputs

<return>	The first owner frame

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

Internal position is advanced.

IsNotComplete

Function Prototype

```
XMPBoolean IsNotComplete();
```

Protocol

Iteration

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Should return kXMPTrue if the iteration is not complete, kXMPFalse otherwise.

Inputs

None.

Outputs

<return>	kXMPTrue if iteration is not complete, kXMPFalse otherwise.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

No effect on internal state.

Next

Function Prototype

```
XMPFrame* Next();
```

Protocol

Iteration

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Should return the next owner frame of this non-exclusive focus.

Inputs

None.

Outputs

<return>	The next owner frame

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

Internal position is advanced.

XMPFocusSet

Basic Class Documentation

An XMPFocusSet is used to store a set of tokenized focus types.
XMPFocusSet has no base class.
Related classes are XMPFocusSetIterator and XMPArbitrator.
XMPFocusSet is platform-independent.
XMPFocusSet participates in the Part Activation protocol.

Theory of Operation

A part activates itself by requesting a set of foci from the arbitrator. The XMPFocusSet class is used to store the set of foci. The foci are of type XMPFocusType (ISO strings) but the elements of the set are tokenized strings (XMPTypeToken). The XMPSession object contains methods for tokenizing.

XMPFocusSet has methods for adding and removing foci, and for testing membership. The companion iterator class, XMPFocusSetIterator, provides enumeration of the elements of the set.

Invariants Maintained by Class

An instance of XMPFocusSet contains a possibly empty set of foci . No duplicates are allowed.

Other Persistent Properties

Member Functions

XMPFocusSet

Function Prototype

XMPFocusSet();

Protocol

Creation and Deletion

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Constructs the focus set. InitFocusSet must also be called.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

The set is constructed, but not ready for use until InitFocusSet is called.

~XMPFocusSet

Function Prototype

~XMPFocusSet();

Protocol

Creation and Deletion

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Frees the memory allocated by this object.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

A valid instance.

Post conditions

The memory used by this object is freed.

Add

Function Prototype

```
void Add(
        XMPTypeToken focus);
```

Protocol

Elements

Protection

Public. Called by part editors.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Adds the specified focus to the set, if it is not already present. Allocates a collection to store the foci, if necessary.

Inputs

focus A token for a focus to add to the set.

Outputs

None

Exceptions Signalled

kXMPErrOutOfMemory

Pre conditions

A constructed object of this class.

Post conditions

The set contains the new focus.

Contains

Function Prototype

```
XMPBoolean Contains(
        XMPTypeToken focus);
```

Protocol

Elements

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns kXMPTrue if the specified focus is an element of the set, kXMPFalse otherwise.

Inputs

focus	A token for a focus to test for membership

Outputs

<return>	kXMPTrue, if the set contains the focus, kXMPFalse otherwise.

Exceptions Signalled

None

Pre conditions

A constructed object of this class.

Post conditions

The function result contains kXMPTrue if the specified focus is an element of the set, kXMPFalse otherwise.

CreateIterator

Function Prototype

```
XMPFocusSetIterator* CreateIterator();
```

Protocol

Iteration

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Creates and returns an iterator for this focus set. The caller is responsible for deleting the iterator.

Inputs

None.

Outputs

<return>	The iterator

Exceptions Signalled

None.

Pre conditions

A constructed object of this class.

Post conditions

Result contains new iterator.

InitFocusSet

Function Prototype

```
void InitFocusSet();
```

Protocol

Creation and Deletion

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Initializes the set.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

A constructed instance.

Post conditions

Instance is usable.

Remove

Function Prototype

```
void Remove(
        XMPTypeToken focus);
```

Protocol

Elements

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Removes the specified focus from the set, if it is present.

Inputs

focus — A token for a focus to remove from the set

Outputs

None

Exceptions Signalled

None

Pre conditions

A constructed object of this class.

Post conditions

The specified focus is no longer in the set.

XMPFocusSetIterator

Basic Class Documentation

XMPFocusSetIterator is a companion of XMPFocusSet, used to iterate over the elements of a focus set.
XMPFocusSetIterator has no base class.
XMPFocusSetIterator is platform-independent.
Related classes are XMPArbitrator, XMPFocusModule and XMPFocusSet.

Theory of Operation

Given a focus set, XMPFocusSetIterator allows the developer to iterate over the set in a loop, using the iterator's First(), Next() and IsNotComplete() methods.

Invariants Maintained by Class

The iterator contains a reference to the set, or to the underlying collection used to implement the set.
The iterator also stores the current element, or a reference to a lower-level collection iterator which contains that state.

Other Persistent Properties

Member Functions

~XMPFocusSetIterator

Function Prototype

~XMPFocusSetIterator();

Protocol

Creation and Deletion

Protection

Public. Parts use delete operator.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Frees memory allocated by this class.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

The memory used by this object is freed.

First

Function Prototype

XMPTypeToken First();

Protocol

Iteration

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Allocates an iterator supplied by the implementation collection class. Resets the iterator to the beginning, returns the first element, and sets it as the current element.

Inputs

None.

Outputs

<return>	The first focus in the set

Exceptions Signalled

kXMPErrOutOfMemory	Not enough memory to allocate the LinkedListIterator

Pre conditions

A constructed object of this class.

Post conditions

The first element is returned, and the iterator is advanced to be ready to return the next element.

IsNotComplete

Function Prototype

```
XMPBoolean IsNotComplete();
```

Protocol

Iteration

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns kXMPTrue if the iteration is not complete.

Inputs

None.

Outputs

<return>	kXMPTrue, if the iteration is not complete.

Exceptions Signalled

None.

Pre conditions

A constructed object of this class.

Post conditions

Result contains kXMPTrue if the iteration is complete, kXMPFalse otherwise.

Next

Function Prototype

```
XMPTypeToken Next();
```

Protocol

Iteration

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the next element of the set.

Inputs

None.

Outputs

<return>	The next focus in the set

Exceptions Signalled

None.

Pre conditions

First has been called.

Post conditions

Returns the next element. Advances the iterator state to the next element.

XMPFrame

Basic Class Documentation

Ancestors: XMPFrame -> XMPPersistentObject -> XMPRefCountObject -> XMPObject

An XMPFrame object (along with its facets) describes the geometric boundary between an embedded part and its containing part. A part may be displayed in multiple frames, and each frame represents a particular view of its parts content. Each frame may also appear in a window in multiple places, one for each facet of the frame.

Frames are created by containing parts as places to embed other parts. Once a part has been embedded, its frame mediates negotiations between the embedded part and its containing part.

Frames maintain a large amount of state information. Some of this data is kept on behalf of the embedded part, some for the containing part, and some small amount for the UI and Imaging subsystems.

A frame holds a reference to its containing frame. This will be kXMPNULL if the frame is the root frame of a window. This value doesn't usually change, but can if the frame was created before it is known where it will be embedded. A frame does not hold direct references to its embedded frames. Only the frame's part knows which frames are embedded in it.

A frame also has a list of facets which describe where it is visible in a window. This list may be empty if the frame is scrolled out of view. Since the facet hierarchy is navigable, it is possible to find some of a frame's embedded frames, by asking the frame's facet to iterate its child facets, and thence to their frames. This technique is fine for finding frames which are visible in a window, but it is unreliable for finding all the frames embedded in a frame, as not all may be visible and thus have facets.

A frame has two basic shapes which define its geometry. The frameShape is the area the containing part has given to the embedded part. The usedShape is the area the embedded part actually needs. Where appropriate, the containing part will wrap content to the contour of the usedShape.

The internalTransform is the third piece of the frame's geometry. It describes how the part's content is scrolled or otherwise transformed within the frame. There are two coordinate systems in the frame to consider. There is the coordinate space of the frame itself, and there is the coordinate space of the frame's content. The internalTransform is the mapping from the content coordinates to the frame coordinates.

A frame of course has a reference to its part. The frame ensures that it is added to the part as one of its display frames when it is created, and removed from the part's set of display frames when it is deleted. A frame's part may store arbitrary, uninterpreted data in the frame's partInfo field. Only the part itself interprets or manipulates this data, but the partInfo is stored with the frame, and only internalized when needed for that frame.

There are two fields which describe how a part is displayed within a frame. The viewType determines whether the frame is a full frame, an icon, a thumbnail, etc. The presentation describes which kind of view of a part is shown in the frame. Examples are: table, bar chart or pie chart; drawing or palette; text or outline. Part's MUST support all standard viewTypes, but the set of presentations varies from part to part. A containing part may request a particular presentation, but the embedded part need not honor that request.

Frame's have a groupID field the containing part may use to group the frames. In cases where a containing part embeds mulitple frames displaying the same part, it may want to group them for group editing, etc.

Frames also have a number of flags:
isRoot - the frame is the root frame of a window
isSubframe - the frame is part of a conglomerate frame (its containing frame displays the same part as the frame itself)
isOverlaid - the frame floats about its containing part's other content, and thus need not negotiate for space
isFrozen - frozen frames cannot be modified by geometry-based UI events
isDroppable - the frame's part supports drag-and-drop events in this frame
isDragging - this frame is currently being dragged, don't try to drop items into it or its embedded frames
usesCachedImage - the frame displays a static image becuase there is no part editor to display its part
doesPropagateEvents - the frame's part delegates unhandled events to its containing part

Theory of Operation

Invariants Maintained by Class

A frame always has a part, and is always a display frame of that part.

A frame always has a valid frameShape. The usedShape need not be set, in which case it will default to be the same as the frameShape. A frame always has a valid internalTransform, which defaults to the identity transformation.

The isRoot, isSubframe and isOverlaid flags are immutable for the lifetime of the frame.

Other Persistent Properties

Member Functions

ChangeFrameShape

Function Prototype

```
void ChangeFrameShape(
        XMPShape* shape);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Called by containing part to change the frame's frameShape. The frame sets its frameShape field to "shape" and notifies its part of the change via Part::FrameShapeChanged().

Inputs

shape	The new frameShape. In frame coordinates.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

ChangeInternalTransform

Function Prototype

```
void ChangeInternalTransform(
        XMPTransform* transform);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Called by the frame's part to change its internalTransform. The frame sets its internalTransform to be "transform", and notifies its facets of the change via Facet::InternalTransformChanged().

Inputs

transform	The new internalTransform.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

ChangeLinkStatus

Function Prototype

```
void   ChangeLinkStatus(
          XMPLinkStatus status);
```

Protocol

Linking

Protection

Public. Parts call this for any XMPFrames that are involved in a link when a link is created, broken, moved, etc.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Set the link status for the frame to the given value. Parts involved in moving or creating frames must set this attribute on frames they own. A frame should then look to its containing frame to see if it should set itself to the requested status or not. It should try to set itself to the net effect of whatever state it's in. The frame should then call XMPPart::LinkStatusChanged to notify the part of the change so that the part may call XMPFrame::ChangeLinkStatus for other owned frames and/or embedded frames

Inputs

None

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

The frame's link status may or may not have changed and frame will call the LinkStatusChanged method of its part.

ChangePart

Function Prototype

```
void ChangePart(
        XMPPart* part);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Remove all of this frame's facets from its old part via Part::FacetRemoved(). Set its part to be "part". Add all its facets to the new part via Part::FacetAdded().

Inputs

part — The frame's new part.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

ChangePresentation

Function Prototype

```
void   ChangePresentation(
        XMPTypeToken presentation);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Called by containing part to request the frame to change its presentation. Sets the presentation to "presentation", and notifies the frame's part of the change via Part::PresentationChanged(). If the part does not support the requested presentation, it may then change the presentation again using the Frame::SetPresentation() method.

Inputs

presentation	The frame's presentation.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

ChangeUsedShape

Function Prototype

```
void ChangeUsedShape(
        XMPShape* shape);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Called by the frame's part to change its usedShape. The frame sets its usedShape to "shape", and notifies its containing part of the change via Part::UsedShapeChanged().

Inputs

shape	The new usedShape. In frame coordinates.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

ChangeViewType

Function Prototype

```
void   ChangeViewType(
        XMPTypeToken viewType);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Called by containing part to request the frame to change its viewType. Sets the viewType to "viewType", and notifies the frame's part of the change via Part::ViewTypeChanged(). If the part does not support the requested viewType, it may then change the viewType again using the Frame::SetViewType() method.

Inputs

viewType	The frame's new viewType.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

CloneTo

Function Prototype

```
XMPStorageUnit* CloneTo(
        XMPDraftKey key,
        XMPDraft* destDraft);
```

Protocol

Storage

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Externalizes this XMPFrame object and recursively clones any of its references to other Storage Units or Parts.

Note that the actual copying may not be completed until after EndClone is finished.

Inputs

key	XMPDraftKey identifying the Clone transaction.
destDraft	destDraft

Outputs

<return>	XMPStorageUnit referring to the Storage Unit to where this Frame is cloned.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

Close

Function Prototype

```
void Close();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Prepare frame for being removed from its draft, but don't make the removal persistent. This call is used to clean up frames when a document is being closed after its final save. The frame notifies its part it is being closed via Part::CloseDisplayFrame(), then releases its references to its part and its containingFrame. During CloseDisplayFrame(), the frame's part will call this method recursively on its embedded frames.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

Frame has a valid part, and is a display frame of that part.

Post conditions

Frame makes no references to other frames or parts.

ContentChanged

Function Prototype

```
void    ContentChanged(
            XMPChangeID change);
```

Protocol

Linking

Protection

Public. Called by the part displayed by this frame object.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Called by the part displayed by the frame object if the frame is affected by a content change in the part. Causes the EmbeddedFrameChanged() method of all containing parts in the frame hierarchy to be called, thru the root part of the window displaying this frame. This method allows a containing part maintaining a source link to be aware when an embedded part has changed, so it can update the link.Parts should avoid calling this at every content change.They can probably call after a reasonable pause and when the part loses the selection focus.

Inputs

change — A unique change ID.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

The EmbeddedFrameChange method of its part will be called.

CreateFacetIterator

Function Prototype

```
XMPFrameFacetIterator* CreateFacetIterator();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Create an iterator for this frame's facets.

Inputs

None.

Outputs

<return>	The iterator. Caller assumes storage responsibility.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

DoesPropagateEvents

Function Prototype

```
XMPBoolean DoesPropagateEvents();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return whether this frame delegates unhandled events to its containing frame.

Inputs

None.

Outputs

<return>	The doesPropagateEvents flag.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

DrawActiveBorder

Function Prototype

```
void DrawActiveBorder();
```

Protocol

None.

Protection

Public. Called by Arbitrator

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Ask frame to draw its active border adornment as needing to be redrawn. The frame forwards the request to all its facets via Facet::DrawActiveBorder().

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

Externalize

Function Prototype

```
void Externalize();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Write out the state of this frame onto its storage unit, suitable for subsequent reconstruction of the frame.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

Frame has a valid storage unit.

Post conditions

None.

FacetAdded

Function Prototype

```
void FacetAdded(
        XMPFacet* facet);
```

Protocol

None.

Protection

Public. Only called by facet being added.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Notify the frame it has a new facet. The frame adds the facet to its list of facets, and notifies its part of the new facet via Part::FacetAdded().

Inputs

facet — The new facet.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

FacetRemoved

Function Prototype

```
void FacetRemoved(
        XMPFacet* facet);
```

Protocol

None.

Protection

Public. Only called by facet.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Notify the frame one of its facets has been removed. The frame removes the facet from its list of facets, and notifies its part via Part::FacetRemoved().

Inputs

facet	The facet being removed.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetContainingFrame

Function Prototype

```
XMPFrame* GetContainingFrame();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return the containing frame of this frame. Will return kXMPNULL if this is a root frame.

Inputs

None.

Outputs

<return>	This frame's containing frame.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetFrameGroup

Function Prototype

XMPULong GetFrameGroup();

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return the groupID of this frame.

Inputs

None.

Outputs

<return>	The groupID of this frame.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetFrameShape

Function Prototype

XMPShape* GetFrameShape();

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return this frame's frameShape.

Inputs

None.

Outputs

<return>	The frame's frameShape. In frame coordinates. Caller may not alter or delete.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetInternalTransform

Function Prototype

XMPTransform* GetInternalTransform();

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return the internalTransform of the frame. This transform specifies how the content displayed in the frame is transformed relative to the frame's coordinate space.

Inputs

None.

Outputs

<return>	The frame's internalTransform. Caller may not alter or deallocate.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetLinkStatus

Function Prototype

```
XMPLinkStatus GetLinkStatus();
```

Protocol

Linking

Protection

Public. Frames will want to call this for their enclosing frames when ChangeLinkStatus is invoked. This will determine how they handle ChangeLinkStatus.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the XMPLink status of the frame.

Inputs

None.

Outputs

<return>	The link status of this frame

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetPart

Function Prototype

```
XMPPart* GetPart();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return this frame's part.

Inputs

None.

Outputs

<return>	This frame's part.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetPartInfo

Function Prototype

```
XMPInfoType GetPartInfo();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return the partInfo data the part has stored in this frame.

Inputs

None.

Outputs

<return>	The frame's partInfo data.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetPresentation

Function Prototype

```
XMPTypeToken GetPresentation();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return the presentation kind of this frame.

Inputs

None.

Outputs

<return>	This frame's presentation kind.

Exceptions Signalled

None.

Pre conditions
None.
Post conditions
None.

GetUsedShape

Function Prototype
XMPShape* GetUsedShape();
Protocol
None.
Protection
Public.
Override policy
Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.
Basic operation
Return the usedShape of this frame.
Inputs
None.
Outputs

<return>	The usedShape. In frame coordinates. Caller may not alter or deallocate.

Exceptions Signalled
None.
Pre conditions
None.
Post conditions
None.

GetViewType

Function Prototype
XMPTypeToken GetViewType();
Protocol
None.
Protection
Public.
Override policy
Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.
Basic operation
Return the viewType of this frame.
Inputs
None.
Outputs

<return>	The viewType of this frame.

Exceptions Signalled
None.
Pre conditions
None.
Post conditions
None.

Invalidate

Function Prototype
void Invalidate(
XMPShape* invalidShape);

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Ask frame to mark "invalidShape" as needing to be redrawn. The frame forwards the request to all its facets via Facet::Invalidate().

Inputs

invalidShape The shape to be redrawn. In frame coordinates.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

InvalidateActiveBorder

Function Prototype

```
void InvalidateActiveBorder();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Ask frame to mark the shape of its active border adornment as needing to be redrawn. This call is used to erase the active border. The frame forwards the request to all its facets via Facet::InvalidateActiveBorder().

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

IsDragging

Function Prototype

```
XMPBoolean IsDragging();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return whether this frame is being dragged, so the Dispatcher knows not to drag it into itself or its contents.

Inputs

None.

Outputs

<return>	The frame's isDragging flag.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

IsDroppable

Function Prototype

```
XMPBoolean IsDroppable();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return whether this frame's part supports drag-and-drop.

Inputs

None.

Outputs

<return>	The frame's isDroppable flag.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

IsFrozen

Function Prototype

```
XMPBoolean IsFrozen();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return whether this frame is frozen.

Inputs

None.

Outputs

<return>	The isFrozen flag of this frame.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

IsOverlaid

Function Prototype

XMPBoolean IsOverlaid();

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

None.

Inputs

None.

Outputs

<return> The frame's IsOverlaid flag.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

IsRoot

Function Prototype

XMPBoolean IsRoot();

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

None.

Inputs

None.

Outputs

<return> The frame's IsRoot flag.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

IsSubframe

Function Prototype

XMPBoolean IsSubframe();

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

None.

Inputs

None.

Outputs

<return>	The frame's isSubframe flag.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

Purge

Function Prototype

```
XMPSize Purge(
        XMPSize size);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Ask the frame to free up to "size" bytes of memory by releasing non-essential internal data structures.

Inputs

size	The number of bytes to attempt to free.

Outputs

<return>	The number of bytes actually freed.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

Release

Function Prototype

```
void Release();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Called by another object which had a reference to this frame upon releasing that reference. Decrements the frame's reference count. If there are no more references to the frame, ask the draft to release its storage.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

Frame has a positive reference count.

Post conditions

None.

Remove

Function Prototype

```
void Remove();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Prepare frame for being removed from its draft. The frame notifies its part it is being removed via Part::RemoveDisplayFrame(), then releases its references to its part and its containingFrame. During RemoveDisplayFrame(), the frame's part will call this method recursively on its embedded frames. If the frame's reference count falls to 1, ask the draft to remove the frame.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

Frame has a valid part, and is a display frame of that part.

Post conditions

Frame makes no references to other frames or parts. Draft may have removed frame.

RequestFrameShape

Function Prototype

```
XMPShape* RequestFrameShape(
        XMPShape* shape);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Called by the frame's part to request a new frameShape. The frame forwards the request (with the shape still in this frame's frame coordinates) to the containing part via Part::RequestFrameShape(). The containing part responds with the shape it will allow the frame to have. The frame stores this shape as its new frameShape, and returns it to the part so the part knows what its new shape is.

Inputs

shape	The requested shape. In frame coordinates.

Outputs

<return>	The new frameShape. In frame coordinates. Caller may not alter or deallocate.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

SetContainingFrame

Function Prototype

```
void  SetContainingFrame(
          XMPFrame* frame);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Set the containing frame of this frame. Only necessary if frame is being moved, as will usually be created with correct value.

Inputs

frame	The new containing frame.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

SetDragging

Function Prototype

```
void  SetDragging(
          XMPBoolean isDragging);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Set whether this frame is in process of being dragged.

Inputs

isDragging	The new value for the frame's isDragging flag.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

SetDroppable

Function Prototype

```
void  SetDroppable(
          XMPBoolean isDroppable);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Called by the frame's part to indicate whether it supports drag-and-drop in this frame.

Inputs

isDroppable	The new value for this frame's isDroppable flag.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

SetFrameGroup

Function Prototype

```
void  SetFrameGroup(
          XMPULong groupID);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Set the groupID of this frame. This should only be done by the containing part.

Inputs

groupID	The new groupID.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

SetFrozen

Function Prototype

```
void  SetFrozen(
          XMPBoolean isFrozen);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Set whether this frame is frozen.

Inputs

isFrozen	The new value for the isFrozen flag.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

SetPartInfo

Function Prototype

```
void  SetPartInfo(
          XMPInfoType partInfo);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Set the frame's partInfo data. Should only be called by this frame's part.

Inputs

partInfo	The frame's new partInfo.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

SetPresentation

Function Prototype

```
void  SetPresentation(
          XMPTypeToken presentation);
```

Protocol

None.

Protection

Public. Should only be called by the frame's part.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Set the presentation kind of this frame to "presentation". Should only be called by frame's part.

Inputs

presentation	The frame's presentation kind.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

SetPropagateEvents

Function Prototype

```
void  SetPropagateEvents(
          XMPBoolean doesPropagateEvents);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Set whether this frame should delegate unhandled events to its containing frame.

Inputs

doesPropagateEvents	New value for doesPropagateEvents flag.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

SetSubframe

Function Prototype

```
void  SetSubframe(
          XMPBoolean isSubframe);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Set whether this frame is a subframe of a containing frame.

Inputs

isSubframe	The value for the isSubframe flag.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

SetViewType

Function Prototype

```
void   SetViewType(
           XMPTypeToken viewType);
```

Protocol

None.

Protection

Public. Should only be called by frame's part.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Set the viewType of this frame to "viewType". Should only be called by frame's part.

Inputs

viewType	The frame's new viewType.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

Validate

Function Prototype

```
void Validate(
           XMPShape* validShape);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Ask frame to mark "validShape" as no longer needing to be redrawn. The frame forwards the request to all its facets via Facet::Validate().

Inputs

validShape	The shape which no loger needs to be redrawn. In frame coordinates.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

XMPFrameFacetIterator

Basic Class Documentation

Ancestors: none

XMPFrameFacetIterator lists the facets of a frame.

Theory of Operation

Invariants Maintained by Class

Other Persistent Properties

Member Functions

XMPFrameFacetIterator

Function Prototype

```
XMPFrameFacetIterator(
        XMPFrame* frame);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Construct an iterator on the facets of a frame.

Inputs

frame	The frame who's facets to iterate.

Outputs

<return>	<return>

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

~XMPFrameFacetIterator

Function Prototype

```
~XMPFrameFacetIterator();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Destructor.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

First

Function Prototype

XMPFacet* First();

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return the first facet of the frame.

Inputs

None.

Outputs

<return>	The first facet in the sequence.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

IsNotComplete

Function Prototype

XMPBoolean IsNotComplete();

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return whether the last item returned from First() or Next() was valid.

Inputs

None.

Outputs

<return>	True if iterator is not yet complete, false otherwise.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

Next

Function Prototype

XMPFacet* Next();

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Answer the next facet in the sequence.

Inputs

None.

Outputs

<return> The next facet.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

XMPLink

Basic Class Documentation

This class represents the destinations of an OpenDoc link. Instances of this class will be created and maintained by XMPDrafts whenever the user creates a dynamic link between parts. The platform vendor implements this class. This class is not to be subclassed by developers. XMPLink is a subclass of XMPPersistentObject.

For further information on the implementation of OpenDoc links, part developers are referred to the documentation for the companion class XMPLinkSource, and the class XMPLinkSpec. Specific methods of class XMPPart and XMPFrame are also relevant to a complete understanding of linking.

Document shell developers and platform vendors are referred to the documentation on the XMPLinkManager.

Theory of Operation

A destination part creates a link by grabbing a link specification from the clipboard or drag-and-drop container, and passing that description to its draft's GetLink method. GetLink returns an XMPLink object through which the destination retrieves the content of the link. See the description of XMPLinkSource for a complete description of the creation of a link.

If the destination part wishes to be notified of changes to the link's content, it registers itself with the link object by calling XMPLink::RegisterDependent. If the link's content has changed since the part last updated, the link object calls the part's LinkUpdated method to notify the part to read the link data. The part will call XMPLink::GetContentStorageUnit to access the data.

A destination part's registration with a link is not permanent. Each time a part containing a link destination is initialized, it should re-register with the link object. A destination part may explicitly unregister itself with a link by calling XMPLink::UnregisterDependent. This should be performed when the user breaks the link at the destination, for example, or if the linked content is deleted.

Invariants Maintained by Class

Dependent parts must be maintained in a list for there may be more than one.

Other Persistent Properties

Member Functions

Externalize

Function Prototype

```
void Externalize();
```

Protocol

Storage

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Externalizes persistent data

Inputs

None.

Outputs

None.

Exceptions Signalled

<Storage exceptions> Storage exceptions may be signalled.

Pre conditions

None.

Post conditions

The link's persistent storage unit is consistent with the link.

GetChangeID

Function Prototype

```
XMPChangeID GetChangeID();
```

Protocol

Linking

Protection

Public. Can be used by parts containing a destination of this link to determine if new content is available.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the current change identification of the link content. This method may be used by destinations to determine if new content is available. Parts can also receive automatic notification of new content by calling the RegisterDependent() method. The returned value should be tested for equality only, as there is no implicit ordering of change ids.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetContentStorageUnit

Function Prototype

```
XMPStorageUnit* GetContentStorageUnit(
        XMPLinkKey key);
```

Protocol

Linking, Storage

Protection

Public. No restrictions.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns a reference to the XMPStorageUnit containing the content of the link.

Inputs

key A valid key returned by Lock().

Outputs

<result>	A reference to the content storage unit.

Exceptions Signalled

kXMPErrInvalidLinkKey

Pre conditions

The key argument is a valid key returned by Lock().

Post conditions

The returned storage unit is guaranteed to be valid until the key is relinquished by Unlock().

GetStatus

Function Prototype

```
XMPError GetStatus();
```

Protocol

Linking

Protection

Public. Should be called by parts when the user explicitly attempts to get updates from the link.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the status of the link. A non-zero value indicates a non-fatal problem which the part may ignore or bring to the user's attention. The most common problem is a broken cross-document destination. This situation may be temporary if the source document re-establishes the cross-document link.

Inputs

None.

Outputs

<result>	Zero if the link is functioning normally. Non-zero if some error ocurred.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

Lock

Function Prototype

```
XMPBoolean Lock(
        XMPULong wait,
        XMPLinkKey* key);
```

Protocol

Thread Safety

Protection

Public. Parts must call this method to aquire a key required by other methods.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Acquire exclusive access to the content storage unit of the link for the current thread. The value kXMPTrue is returned if the lock is granted; the key argument is set to a valid link key. Access is granted if the current thread already holds the

lock. Nested calls to Lock should be balanced by an equal number of calls to Unlock to relinquish the lock.

Inputs

wait	The interval to wait for access to be granted. A value of zero means no wait, a value of ULONG_MAX means an indefinite wait. Other values are platform-dependent.

Outputs

key	If the result is kXMPTrue, a valid key value required by routines that access or modify the link's content. If the result is kXMPFalse, key is an undefined invalid key.
<result>	kXMPTrue if access is granted and kXMPFalse if denied.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

If kXMPTrue is returned, the current thread has exclusive access to the link's content, and the key parameter is valid.

RegisterDependent

Function Prototype

```
void RegisterDependent(
        XMPPart* clientPart,
        XMPChangeID id);
```

Protocol

Linking

Protection

Public. Parts containing the destination of this link call this method to receive notification of changes to the link's content.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Puts the clientPart reference on a list for notification of changes to the link's content.

Inputs

clientPart	A destination part wishing to be informed of changes to the contents of the link.
id	The identification of the last known change. The constant kChangeUnknown may be supplied if the part does not keep track of link change identifications. The clientPart's LinkUpdated() method is called immediately if the link's current change id differs from the argument.

Outputs

None

Exceptions Signalled

kXMPErrOutOfMemory	Not enough memory

Pre conditions

None.

Post conditions

None.

ShowSourceContent

Function Prototype

```
void ShowSourceContent();
```

Protocol

Linking

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Causes the source part of the link to become active, makes the source content visible, and displays the link border. Throws an exception if the source part of the link cannot be located, or if the link has been broken at the source.

Inputs

None.

Outputs

None.

Exceptions Signalled

kXMPErrLinkBroken	Cannot locate the source of the link.

Pre conditions

None.

Post conditions

None.

Unlock

Function Prototype

```
void Unlock(
        XMPLinkKey key);
```

Protocol

Thead safety

Protection

Public. Parts must call this method when done accessing or modifying the content of a link.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Relinquish a key returned by Lock(). Exclusive access to the link's content by this thread is relinquished if this thread did not already have exclusive access when the key was granted.

Inputs

key	A valid key returned by Lock().

Outputs

None.

Exceptions Signalled

kXMPErrInvalidLinkKey

Pre conditions

The key argument is a valid key returned by Lock().

Post conditions

The key is no longer valid. Exclusive access to the link's content by this thread is relinquished if this thread did not already have exclusive access when the key was granted.

UnregisterDependent

Function Prototype

```
void UnregisterDependent(
        XMPPart* clientPart);
```

Protocol

Linking

Protection

Public. The destination XMPPart will call this.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Removes the previously registered XMPPart reference from our internal list. Return without error if the part was not previously registered.

Inputs

clientPart	A destination part that had previously registered itself as a dependent of the link.

Outputs

None

Exceptions Signalled

None

Pre conditions

clientPart should have previously registered with this link.

Post conditions

clientPart is not registered with this link.

XMPLinkManager

Basic Class Documentation

The Link Manager coordinates the creation and maintanence of cross-document links. There is one Link Manager object per session. The Link Manager is a non-persistent object instantiated by the session, which provides access to it. The Link Manager is called by link objects, draft objects, and by document shells. It is not visible to parts. Because the implementation of cross-document links is platform-dependent, the Link Manager is a platform-specific class.

Document shell developers should understand the following Link Manager methods: AnyLinkImported(), NewSectionID(), ReserveSectionID(), and UnsavedExportedLinks(). In addition, the Link Manager depends on the document shell to call DraftOpened(), DraftSaved(), and DraftClosing() at the proper times.

Theory of Operation

The Link Manager provides (1) the establishment of cross-document links, (2) the updating of cross-document links in accordance with the user interface guidelines, and (3), on the Macintosh, coordination of use of the Edition Manager with the document shell. It is dependent on the document shell to provide proper updating of cross-document links.

When a cross-document link is established, the Link Manager of the destination draft communicates with the Link Manager of the source draft to get the link created. Neither the source nor destination parts are aware that the link is cross-document; this fact is a secret shared by the LinkManager and the Link and LinkSource objects in both drafts.

When a draft is opened, the Link Manager ensures that destinations of links from other documents have the opportunity to update to the latest content. It also maintains a list of cross-document link sources that should be updated when the draft is saved.

The Link Manager provides coordination of Edition Manager section ids with Macintosh document shells that support Publish and Subscribe. The shell can reserve section ids already in use by calling ReserveSectionID(). The shell can get a new section ID unique for the life of the document by calling NewSectionID().

The Link Manager also provides important information to the document shell. The shell should call AnyLinkImported() when it would otherwise close the document without saving; a return value of kXMPTrue indicates that links were updated and that the document should be saved without user interaction. When the user closes or reverts a draft without saving, the shell should call UnsavedExportedLinks(); a return value of kXMPTrue indicates that new cross-document links would be abandoned if the draft isn't saved.

Invariants Maintained by Class

Other Persistent Properties

kXMPPropReservedSectionIDs -- the list of reserved section identifiers, present in the document properties storage unit.

kXMPPropLastSectionID -- the section identifier last returned by NewSectionID(), present in the document properties storage unit.

Member Functions

AnyLinkImported

This method is only of interest to Container Application developers.

Function Prototype

```
XMPBoolean AnyLinkImported(
        XMPDraft* draft);
```

Protocol

Linking

Protection

Public. For use by container apps only. Parts do not call.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns kXMPTrue if one or more links to source in other documents have automatically updated since the draft was last saved. The shell should silently save the imported content if no other changes to the draft were made by the user.

Inputs

draft	The draft of interest.

Outputs

<result>	kXMPTrue if one or more links have automatically updated, and kXMPFalse otherwise.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

DraftClosing

This method is only of interest to Container Application developers.

Function Prototype

```
void DraftClosing(
        XMPDraft* draft);
```

Protocol

Linking

Protection

Public. For use by container apps only. Parts do not call.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Must be called PRIOR to reverting or closing a draft. The link manager will delete any edition files created after the last save, breaking cross-document links at the source. The shell can call UnsavedExportedLinks() to determine if closing the document would delete edition files.

Inputs

draft	The draft about to be closed.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

DraftOpened

This method is only of interest to Container Application developers.

Function Prototype

```
void DraftOpened(
        XMPDraft* draft);
```

Protocol

Linking

Protection

Public. For use by container apps only. Parts do not call.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Must be called after the argument draft is opened. The link manager will internalize each link object in the draft, allowing those subscribing to or publishing edition files to register with the Edition manager.

Inputs

draft	The draft opened.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

DraftSaved

This method is only of interest to Container Application developers.

Function Prototype

```
void DraftSaved(
        XMPDraft* draft);
```

Protocol

Linking

Protection

Public. For use by container apps only. Parts do not call.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Must be called after every save. Following each save, links that publish a cross-document link will update their edition file. This method will throw an exception if the edition files cannot be updated; the shell should report this error to the user. The draft should be considered unsaved.

The link manager will prohibit this document from publishing a link to another document before the first call to this routine for any draft in the document. A document must be in a permanent location (not a temporary one) in order to reliably track Edition files on the Macintosh.

Inputs

draft	The draft saved.

Outputs

None.

Exceptions Signalled

<platform errors>	An edition file could not be sucessfully written.

Pre conditions

None.

Post conditions

None.

NewSectionID

This method is Macintosh specific.

This method is only of interest to Container Application developers.

Function Prototype

```
XMPULong NewSectionID(
        XMPDocument* document);
```

Protocol

Linking

Protection

Public. For use by container apps only. Parts do not call.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Return a new ID guaranteed to be unique for the lifetime of the argument document.

Inputs

document	The document in which to reserve an ID.

Outputs

<result>	A unique ID.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

ReserveSectionID

This method is Macintosh specific.

This method is only of interest to Container Application developers.

Function Prototype

```
XMPBoolean ReserveSectionID(
        XMPULong sectionID,
        XMPDocument* document);
```

Protocol

Linking

Protection

Public. For use by container apps only. Parts do not call.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Guarantees the argument section ID will not be issued by NewSectionID() for the lifetime of the document. ReserveSectionID returns kXMPTrue if the argument

sectionID is not already reserved and kXMPFalse otherwise. If kXMPFalse is returned, the caller should request a different ID or call NewSectionID().

Inputs

sectionID	The ID to be reserved.
document	The document in which sectionID is reserved.

Outputs

<result>	kXMPTrue if the argument sectionID is not in use and not already reserved, and kXMPFalse otherwise.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

UnsavedExportedLinks

This method is only of interest to Container Application developers.

Function Prototype

```
XMPBoolean UnsavedExportedLinks(
        XMPDraft* draft);
```

Protocol

Linking

Protection

Public. For use by container apps only. Parts do not call.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns kXMPTrue if cross-document links have been established since the draft was last saved. Should be called by the document shell prior to reverting or closing the document. If reverting or closing the document would cause newly created cross-document links to be abandoned, the document shell should alert the user before proceeding with the operation.

Inputs

draft	The draft of interest.

Outputs

<result>	kXMPTrue if cross-document links have been established and kXMPFalse otherwise.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

XMPLinkSource

Basic Class Documentation

This class represents the source of an OpenDoc link. These will be created and maintained by XMPDrafts whenever the user creates a dynamic link between parts. The platform vendor will implement this class. This class is not to be subclassed by the developer. XMPLinkSource is a subclass of XMPPersistentObject.

For further information on the implementation of OpenDoc links, part developers are referred to the documentation for the companion class XMPLink, and the class XMPLinkSpec. Specific methods of class XMPPart and XMPFrame are also relevant to a complete understanding of linking.

Document shell developers and platform vendors are referred to the documentation on the XMPLinkManager.

Theory of Operation

A link is a first class, persistent, one-way conduit for data from one part, the source part, to another part, the destination part (which may be the same part). When source part data is being cut or dragged, the source part should write out a link specification (see the XMPLinkSpec class) in addition to the content that it writes. The link specification is a signal that a link can be made to the source part. The destination part determines the value types available for linking by examining the value types present in the content property.

When a link is requested by the destination part, the following events occur:

The destination part asks its draft for an XMPLink object via the XMPDraft::GetLink method. If the source part is in the same document as the destination part, the draft calls the source part's XMPPart::CreateLink method. If the source part is in a different document, the draft calls the link manager's XMPLinkManager::CreateLink method to get a cross-document link. Ultimately, the XMPDraft::GetLink method returns an XMPLink object to the destination part.

The source part, whether in the same draft or in a different draft, is asked to create an XMPLink object via the XMPPart::CreateLink function. The part may refuse this request in which case the link cannot be created; normally the source part would refuse only in the event of an error.

If the source part has not already created a link to the content described by the link specification, the source part calls its draft's CreateLinkSource method to create an XMPLinkSource and XMPLink object pair. The draft object associated with the source part is responsible for the ephemeral and persistent storage of both link objects. The source part keeps a reference to the XMPLinkSource object, and returns a reference to the XMPLink object. Source parts reference only XMPLinkSource objects, and destination parts reference only XMPLink objects. These reference restrictions are important in assuring the proper behavior when linked content is copied or moved.

If the source part is in a different draft, the source part returns the XMPLink object to the Link Manager. The source draft Link Manager creates a platform-dependent cross-document link, and returns the identity of the link to the

destination draft Link Manager. The destination draft Link Manager calls the destination draft's CreateLinkSource method to create an XMPLinkSource and XMPLink object pair in the destination draft, and returns the XMPLink object to the destination draft's GetLink method. The destination draft returns the XMPLink object to the destination part.

If the destination part wishes to be notified of changes to the link's content, it registers itself with the link object by calling XMPLink::RegisterDependent. The link object will immediately call XMPPart::LinkUpdated to force the part to read the link data. The part will call XMPLink::GetContentStorageUnit to access the data.

Whenever the source part wishes to inform the destination part of a change to a link's contents, it updates the storage unit associated with the link object and calls XMPLink::ContentChanged. The link will call XMPPart::LinkUpdated for all its dependent parts. Each dependent part will read the new data from the link's storage unit.

A destination part's registration with a link is not permanent. Each time a part containing a link destination is initialized, it should re-register with the link object. A destination part may explicitly unregister itself with a link by calling XMPLink::UnregisterDependent. This should be performed when the user breaks the link at the destination, for example, or if the linked content is deleted.

A part writes content to and reads content from a link in exactly the same manner it uses to write and read the clipboard, except that a part should not write a link specification to the link.

Invariants Maintained by Class

The link always owns a valid storage unit.

Other Persistent Properties

Member Functions

Clear

Function Prototype

```
void Clear(
        XMPChangeID id,
        XMPLinkKey key);
```

Protocol

Linking

Protection

Public. Parts call to remove the link's previous contents.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Empties the contents of the link. The key parameter should be obtained by aquiring the lock for this thread.
The change identification argument should be obtained by calling the XMPSession::UniqueChangeID method.

Inputs

id	The change identification associated with the new empty content.
key	A valid key returned by Lock().

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

The next call to GetContentStorageUnit returns a storage unit with no properties.

ContentChanged

Function Prototype

```
void ContentChanged(
        XMPChangeID id);
```

Protocol

Linking

Protection

Public. Called by parts after modifying the link's content.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Called by the source part after modifying the link's content to associate a new change identification with the content. All registered destinations are notified of the updated link contents. The source part should call the XMPSession::UniqueChangeID method to get a new change identification, unless the change resulted from an updated destination link. In that case, the source part should pass on the change ID associated with the destination link.

Inputs

id	The change identification associated with the new content.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

Externalize

Function Prototype

```
void Externalize();
```

Protocol

Storage

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Externalizes persistent data

Inputs

None.

Outputs

None.

Exceptions Signalled

<Storage exceptions> Storage exceptions may be signalled.

Pre conditions

None.

Post conditions

The link's persistent storage unit is consistent with the link.

GetChangeID

Function Prototype

```
XMPChangeID GetChangeID();
```

Protocol

Linking

Protection

Public. Can be used by Parts containing a destination of this link to determine if new content is available.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the current change identification of the link content. The returned value should be tested for equality only, as there is no implicit ordering of change ids.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetContentStorageUnit

Function Prototype

```
XMPStorageUnit* GetContentStorageUnit(
        XMPLinkKey key);
```

Protocol

Linking, Storage

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns a reference to the XMPStorageUnit containing the content of the link.

Inputs

key A valid key returned by Lock().

Outputs

<result> A reference to the content storage unit.

Exceptions Signalled

kXMPErrInvalidLinkKey

Pre conditions

The key argument is a valid key returned by Lock().

Post conditions

The returned storage unit is guaranteed to be valid until the key is relinquished by Unlock().

GetLink

Function Prototype

```
XMPLink* GetLink();
```

Protocol

Linking

Protection

Public. Called by the source part to obtain the associated XMPLink object.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Return the link object associated with this link source. The source part calls this method to obtain a result to return from its CreateLink method.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

IsAutoExport

Function Prototype

```
XMPBoolean IsAutoExport();
```

Protocol

Linking

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns kXMPTrue if updates are automatically exported to other documents when the link's draft is saved. This setting defaults to kXMPTrue.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

Lock

Function Prototype

```
XMPBoolean Lock(
        XMPULong wait,
        XMPLinkKey* key);
```

Protocol

Thread Safety

Protection

Public. Parts must call this method to aquire a key required by other methods.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Acquire exclusive access to the content storage unit of the link for the current thread. The value kXMPTrue is returned if the lock is granted; the key argument is set to a valid link key. Access is granted if the current thread already holds the lock. Nested calls to Lock must be balanced by an equal number of calls to Unlock to relinquish the lock.

Inputs

wait	The interval to wait for access to be granted. A value of zero means no wait, a value of ULONG_MAX means an indefinite wait. Other values are platform-dependent.

Outputs

<result>	kXMPTrue if access is granted and kXMPFalse if denied.
key	If the result is kXMPTrue, a valid key value required by routines that access or modify the link's content. If the result is kXMPFalse, key is an undefined invalid key.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

If kXMPTrue is returned, the current thread has exclusive access to the link's content, and the key parameter is valid.

SetAutoExport

Function Prototype

```
void SetAutoExport(
        XMPBoolean automatic);
```

Protocol

Linking

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

If automatic is kXMPTrue, export updates to other documents when the link's draft is saved. If automatic is kXMPFalse, updates are exported when the ContentChanged method is called.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

ShowSourceContent

Function Prototype

```
void ShowSourceContent();
```

Protocol

Linking

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Causes the source part of the link to become active, makes the source content visible, and displays the link border. Throws an exception if the source part of the link cannot be located, or if the link has been broken at the source.

Inputs

None.

Outputs

None.

Exceptions Signalled

kXMPErrLinkBroken	Cannot locate the source of the link.

Pre conditions

None.

Post conditions

None.

Unlock

Function Prototype

```
void Unlock(
        XMPLinkKey key);
```

Protocol

Thead safety

Protection

Public. Parts must call this method when done accessing or modifying the content of a link.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Relinquish a key returned by Lock(). Exclusive access to the link's content by this thread is relinquished if this thread did not already have exclusive access when the key was granted.

Inputs

key	A valid key returned by Lock().

Outputs

None.

Exceptions Signalled

kXMPErrInvalidLinkKey

Pre conditions

The key argument is a valid key returned by Lock().

Post conditions

The key is no longer valid. Exclusive access to the link's content by this thread is relinquished if this thread did not already have exclusive access when the key was granted.

XMPLinkSpec

Basic Class Documentation

This class represents potential links to part content. This class is not to be subclassed by the developer.

For further information on the implementation of OpenDoc links, part developers are referred to the documentation for the XMPLink and XMPLinkSource classes. Specific methods of class XMPPart and XMPFrame are also relevant to a complete understanding of linking.

Developers implementing a document shell and platform implementors are also referred to the documentation on the XMPLinkManager.

Theory of Operation

A link specification represents a potential link to part content. When source part data is being copied or dragged, the source part should write out a link specification along with the content that it writes. This will be a signal that a link can be made to the source part. By implication, the value types available in the content define the value types available through the link.

A recieving part creates a persistent link by calling the GetLink method of its XMPDraft object, passing in the link specification. If the source part is in the same document, the draft calls the source part's CreateLink method, passing back the private data supplied by the source part when the link specification was created. If the source part is in another document, the draft object forwards the link specification to the link manager, which creates a cross-document link.

An XMPLinkSpec is a non-persistent object that contains methods for writing itself to and from a pre-focused storage unit as a value.

A source part calls XMPDraft::CreateLinkSpec() to originate the link specification. The source part supplies an untyped block of data to identify the content. While this data may be anything whatsoever, parts supporting the object model may find that an object specifier is a natural choice. The link specification can be turned into a storage unit value via the CopyToProperty method. By convention, the link specification is written to a kXMPPropLinkSpec property. The part receiving a link specification value creates a new XMPLinkSpec object and calls its CopyFromProperty method to initialize it.

If a link specification is written to the clipboard, and the content identified by the link specification is changed in a manner that doesn't affect the clipboard, the source part should remove the link specification from the clipboard. The source part can use clipboard change ids to determine that its content is still on the clipboard.

A link spec value is meaningless after the document containing the part that created it is closed.

Invariants Maintained by Class

A link spec value identifies its source part across document boundaries.
A link spec value identifies its source part until the document containing the source part is closed.

Other Persistent Properties

Member Functions

XMPLinkSpec

Function Prototype

XMPLinkSpec();

Protocol

Creation and Deletion

Protection

Public. Source parts should call XMPDraft::CreateLinkSpec. The destination part calls this routine prior to reading a link specification value.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Creates an XMPLinkSpec object.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

The object is initialized.

~XMPLinkSpec

Function Prototype

~XMPLinkSpec();

Protocol

Creation and Deletion

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Destroys the XMPLinkSpec object.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

This link specification object doesn't exist.

ReadFromProperty

Function Prototype

```
void ReadFromProperty(
        XMPStorageUnit* su);
```

Protocol

Linking

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Initialize this link specification from a link spec value in the argument storage unit. The storage unit must be pre-focused to a property.

Inputs

su	A storage unit that has been pre-focused to a property containing a link specification value.

Outputs

None.

Exceptions Signalled

kXMPErrNoLinkSpecValue	Focused property does not contain a link specification value.
kXMPErrCorruptLinkSpecValue	Focused storage unit contains an invalid link specification value.
kXMPErrCorruptLinkSpecValue	Focused storage unit contains an invalid link specification value.

Pre conditions

The argument storage unit is pre-focused to a property containing a link spec value.

Post conditions

This object is initialized to the link specification in the pre-focused storage unit.

WriteToProperty

Function Prototype

```
void CopyToProperty(
        XMPStorageUnit* su);
```

Protocol

Linking

Protection

Public. Called by parts to put a link specification on the clipboard or drag-and-drop storage unit.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Write this object into a pre-focused storage unit. The argument storage unit must be pre-focused to a property.

Inputs

su	A storage unit that has been pre-focused to a property to contain the link specification as a value.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

The argument storage unit is focused to a property.

Post conditions

The argument storage unit contains a link specification value in its focused property.

XMPMenuBar

Basic Class Documentation

XMPMenuBar represents a composite menu bar, made up of menus from the shell and the active part.
XMPMenuBar is a subclass of XMPRefCntObject.
XMPMenuBar is implemented by platform vendors , and is platform-specific.
XMPMenuBar participates in the UI Events and Part Activation protocols.

Theory of Operation

Partly to facilitate interoperability with OLE 2.0., interactions with the menu bar are encapsulated in the XMPMenuBar class. The document shell creates a menu bar, adds menus to it, and makes it the base menu bar by calling XMPWindowState::SetBaseMenuBar(). Part Editors obtain a copy of the base menu bar by calling XMPWindowState::CopyBaseMenuBar(). They then add menus to this copy, and call its Display() method to make it the current menu bar.

Command Numbers and Menu Events:

Unlike MS Windows, the Macintosh Menu Manager identifies a chosen menu item using the menu ID and item number, rather than allowing the programmer to associate a unique position-independent ID with a given menu item.

The Macintosh implementation of OpenDoc allows the document shell application and part editors to register command IDs for menu items. A mouse down in the menu bar is turned into a menu event, with the message field containing the menu result, from which a part can obtain the command number. If no command is registered, a synthetic command is generated. If a command is not registered and not synthetic, a part should not be handling it.

Invariants Maintained by Class

Other Persistent Properties

Member Functions

XMPMenuBar

Function Prototype

XMPMenuBar();

Protocol

Creation and Deletion

Protection

Public. Part editors usually call XMPWindowState::CopyBaseMenuBar() to create a menu bar.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Constructs the menu bar object. InitMenuBar must also be called.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

Instance is in a safe state

~XMPMenuBar

Function Prototype

```
~XMPMacMenuBar();
```

Protocol

Creation and Deletion

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Frees the memory allocated by this instance.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

The memory allocated by this instance is freed, and the instance is no longer usable.

AddMenuBefore

Function Prototype

```
void AddMenuBefore(
        XMPMenuID menuID,
        XMPPlatformMenu menu,
        XMPPart* part,
        XMPMenuID beforeID);
```

Protocol

Insertion and Deletion

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Inserts a menu before the menu with the specified ID.

Inputs

menuID	The ID of the new menu
menu	The platform-specific menu structure
part	The part whichi owns the menu, or kXMPNULL if it is a shell menu.

beforeID — The ID of the menu before which to insert the new one.

Outputs

None.

Exceptions Signalled

kXMPErrOutOfMemory — Out of memory

Pre conditions

A valid initialized instance.

Post conditions

The collection of menus contains the specified menu.

AddMenuLast

Function Prototype

```
void AddMenuLast(
        XMPMenuID menuID,
        XMPPlatformMenu menu,
        XMPPart* part);
```

Protocol

Insertion and Deletion

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Appends a menu to the end of the menu bar.

Inputs

menuID — The ID of the new menu

menu — The platform-specific menu structure

part — The part which owns the menu, or kXMPNULL if it is a shell menu.

Outputs

None.

Exceptions Signalled

kXMPErrOutOfMemory — Out of memory

Pre conditions

A valid initialized instance.

Post conditions

The collection of menus contains the specified menu.

AddSubMenu

This method is Macintosh specific.

Function Prototype

```
void AddSubMenu(
        XMPMenuID menuID,
        XMPPlatformMenu menu,
        XMPPart* part);
```

Protocol

Insertion and Deletion

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Adds a submenu to the menu list (required on Macintosh)

Inputs

menuID	The ID of the new menu
menu	The platform-specific menu structure
part	The part whichi owns the menu, or kXMPNULL if it is a shell menu.

Outputs

None.

Exceptions Signalled

kXMPErrOutOfMemory	Out of memory

Pre conditions

A valid initialized instance.

Post conditions

The collection of menus contains the specified sub menu.

Copy

Function Prototype

```
XMPMenuBar* Copy();
```

Protocol

Creation and Deletion

Protection

Public. Call by WindowState::CopyBaseMenuBar. Parts don't typically call this method.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Clones this object, including its command table.

Inputs

None.

Outputs

<return>	A copy of this menu bar

Exceptions Signalled

None.

Pre conditions

A valid initialized menu bar.

Post conditions

No effect on this instance.

Display

Function Prototype

```
void Display();
```

Protocol

Display

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Installs this menu bar as the current menu bar.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

The current menu bar contains the menus in this instance.

GetCommand

This method is Macintosh specific.

Function Prototype

```
XMPCommandID GetCommand(
        XMPMenuID menu,
        XMPMenuItemID menuItem);
```

Protocol

Command IDs

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the command ID registered for the specified menu and menu item.

Inputs

menu	The menu ID.
menuItem	The menu item.

Outputs

<return>	A command number, or kXMPNoCommand

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

No effect on this instance.

GetMenu

Function Prototype

```
XMPPlatformMenu GetMenu(
        XMPMenuID menu);
```

Protocol

Accessing

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the platform-specific menu structure (menu handle on Macintosh) for the specified menu ID.

Inputs

menu	The ID of the desired menu.

Outputs

<return>	The menu handle

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetMenuAndItem

This method is Macintosh specific.

Function Prototype

```
void GetMenuAndItem(
        XMPCommandID command,
        XMPMenuID& menu,
        XMPMenuItemID& menuItem);
```

Protocol

Command IDs

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Given a command ID, returns the menu and item for that command, registered or synthetic.

Inputs

command	The command to "decompose"

Outputs

menu	The associated menu
menuItem	The associated menu item

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

No effect on this instance.

InitMenuBar

Function Prototype

```
InitMenuBar(
        XMPSession* session,
        XMPPlatformMenuBar menuBar);
```

Protocol

Creation and Deletion

Protection

Public. Part editors usually call XMPWindowState::CopyBaseMenuBar()

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Initializes the menu bar object. Must be called immediately after new().

Inputs

session	The OpenDoc session object.
menuBar	The shared systen-wide menu bar handle

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

IsCommandRegistered

This method is Macintosh specific.

Function Prototype

```
XMPBoolean IsCommandRegistered(
        XMPCommandID command);
```

Protocol

Command IDs

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns kXMPTrue of the specified command ID is registered in this menu bar.

Inputs

command	The command ID to check

Outputs

<return>	kXMPTrue, if the command is registered, otherwise kXMPFalse

Exceptions Signalled

None.

Pre conditions

A valid initialized instance

Post conditions

No effect on this instance.

IsCommandSynthetic

This method is Macintosh specific.

Function Prototype

```
XMPBoolean IsCommandSynthetic(
        XMPCommandID command);
```

Protocol

Command IDs

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns kXMPTrue if the specified command ID is synthetic, i.e. one manufactured from the menu and item IDs.

Inputs

command	The command ID to check

Outputs

<result>	kXMPTrue, if the command is synthetic.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance

Post conditions

No effect on this instance.

Purge

This method is only of interest to Container Application developers.

Function Prototype

```
XMPSize Purge(
        XMPSize size);
```

Protocol

Low Memory

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Tries to free up some memory

Inputs

size	The amount of memory requested

Outputs

<return>	The amount of memory freed.

Exceptions Signalled

None

Pre conditions

Constructed and initialized object.

Post conditions

Some internal structures which are readily rebuilt may have been freed.

RegisterCommand

This method is Macintosh specific.

Function Prototype

```
void RegisterCommand(
        XMPCommandID command,
        XMPMenuID menu,
        XMPMenuItemID menuItem);
```

Protocol

Command IDs

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Associates a Command ID with a specifed menu and menu item.

Inputs

command	A 32-bit command number
menu	A menu ID
menuItem	A menu item number

Outputs

None.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

The command/(menu,item) association is added to a table.

Release

Function Prototype

```
void Release();
```

Protocol

Reference Counting

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Overridden to delete this object if the reference count has gone to zero.

Inputs

None

Outputs

None

Exceptions Signalled

None

Pre conditions

???

Post conditions

???

RemoveMenu

Function Prototype

```
void RemoveMenu(
        XMPMenuID menu);
```

Protocol

Insertion and Deletion

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Removes the menu with the specified ID. The diplayed menu bar is not affected.

Inputs

menu — The id of the menu to remove

Outputs

None.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

The menu is no longer in this menu bar. The diplayed menu bar is not affected.

UnregisterAll

This method is Macintosh specific.

Function Prototype

```
void UnregisterAll();
```

Protocol

Command IDs

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Unregisters all command numbers in this menu bar.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance

Post conditions

No Command IDs are registered in this menu bar.

UnregisterCommand

This method is Macintosh specific.

Function Prototype

```
void UnregisterCommand(
        XMPCommandID command);
```

Protocol

Command IDs

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Remove the association between the specified command ID and its menu and item.

Inputs

command A command ID

Outputs

None.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

The association is removed.

XMPMessageInterface

Basic Class Documentation

This class encapsulates AppleEvent message sending functionality. It works in concert with the XMPSemanticInterface class. The XMPSession object allows access to an instance of this object for any clients who wish to use it. The platform vendor will implement. XMPMessageInterface is a subclass of XMPObject.

A minimum of documentation is given for functions that have counterparts in the AppleEvent Manager. Only differences are pointed out.
The OSErr return value normally returned by these functions will instead be signalled via an exception should it not be equal to noErr.
Only exceptions that are different from, or exceptions that occur in addition to the normal AppleEvent exceptions, are documented.

Theory of Operation

The functions of this class are direct counterparts to the associated AppleEvent functions. AppleEvents will be used to implement the member functions of this class. An extra parameter or type XMPPart has been added to many of the functions. This is used to specify the context that should be used when performing the function. A special value of XMPPart*, kXMPAppShell is used to denote the application shell "part".
Callback functions have been changed to accomodate parts. All such functions take an XMPPart* as a first parameter. This value will be kXMPAppShell for the system and application shell XMPSemanticInterface objects (see XMPSemanticInterface class documentation). It will contain a pointer to the particular instance of XMPPart (or a subclass) to which the XMPSemanticInterface object belongs.

Invariants Maintained by Class

None.

Other Persistent Properties

Member Functions

CreateEvent

Function Prototype

```
virtual void CreateEvent(
        AEEventClass theAEEventClass,
        AEEventID theAEEventID,
        AEAddressDesc target,
        XMPSLong transactionID,
        AppleEvent* result);
```

Protocol

Semantic Events

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

See AppleEvent Manager documentation. The returnID parameter has been removed since OpenDoc will use this to track the origin of a Send.

Inputs

theAEEventClass	<See AppleEvent Manager documentation>
theAEEventID	<See AppleEvent Manager documentation>

	target	<See AppleEvent Manager documentation>
	transactionID	<See AppleEvent Manager documentation>
Outputs		
	result	Apple event created
	<return>	The return id assigned to this event.
Exceptions Signalled		
	kXMPOutOfMemory	Couldn't allocate memory for the event.
	others	<See AppleEvent Manager documentation>

Pre conditions

None.

Post conditions

An event will be created.

CreatePartAddrDesc

Function Prototype

```
XMPVMethod void CreatePartAddrDesc(
        AEDesc*     theAddressDesc,
        XMPPart*    thePart);
```

Protocol

Semantic Events

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Create an address descriptor to be used when sending events to the part in question.

Inputs		
	thePart	The XMPPart* to be addressed.
Outputs		
	theAddressDesc	The descriptor to be initialized.
Exceptions Signalled		
	kXXMPErrNotAValidPart	XMPPart* was not valid.

Pre conditions

None.

Post conditions

The address descriptor is initialized.

CreatePartObjSpec

Function Prototype

```
XMPVMethod void CreatePartObjSpec(
        AEDesc* theObjSpec,
        XMPPart* thePart);
```

Protocol

Semantic Events

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Creates an object specifier that can be used to refer to the part in question. Should not be stored persistently. Is only valid while the XMPPart is instantiated.

Inputs

thePart	The part for which an object specifier will be constructed.

Outputs

theObjSpec	The object specifier to be initialized

Exceptions Signalled

kXXMPErrNotAValidPart	XMPPart* was not valid.

Pre conditions

None.

Post conditions

The object specifier is initialized

ProcessSemanticEvent

This method is only of interest to Container Application developers.

Function Prototype

```
virtual XMPBoolean ProcessSemanticEvent(
        const XMPEventData* theEvent);
```

Protocol

Semantic Events

Protection

Public. Probably only called from an XMPDispatchModule.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

See AppleEvent Manager documentation. This function is similiar to AEProcessAppleEvent, but it may not call AEProcessAppleEvent at all. It is responsible to for dispatching a semantic event to the correct object and using the corrent XMPSemanticInterface object owned by that object for further dispatching. The object will either be the application shell / document or a part.

Inputs

theEvent	<See AppleEvent Manager documentation>

Outputs

<return>	Whether the event was processed.

Exceptions Signalled

kXMPEResolveError	Couldn't determine correct object to dispatch to.

Pre conditions

None.

Post conditions

Event will be dispatched.

Purge

Function Prototype

```
XMPSize Purge(
        XMPSize howMuch);
```

Protocol

Semantic Events, XMPObject

Protection

Public.

Override policy

Derived class **can** override.

Derived class **must** call base class behavior, during derived class behavior.

Basic operation

None.

Inputs

howMuch	Amount of memory requested to be purged.

Outputs

<return>	Amount of memory that was released.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

Memory may have been freed.

Send

Function Prototype

```
virtual XMPSShort Send(
        XMPPart* thePart,
        AppleEvent* theAppleEvent,
        AppleEvent* reply,
        AESendMode sendMode,
        AESendPriority sendPriority,
        XMPULong timeOutInTicks,
        XMPIdleProcPtr idleProc,
        XMPEventFilterProcPtr filterProc);
```

Protocol

Semantic Events

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

See AppleEvent Manager documentation. The XMPPart* parameter specifies the sender of the event.

Inputs

thePart	The sending XMPPart or kXMPAppShell if the sender is the application shell / document.
theAppleEvent	<See AppleEvent Manager documentation>
sendMode	<See AppleEvent Manager documentation>
sendPriority	<See AppleEvent Manager documentation>
timeOutInTicks	<See AppleEvent Manager documentation>
idleProc	<See AppleEvent Manager documentation>
filterProc	<See AppleEvent Manager documentation>

Outputs

reply	<See AppleEvent Manager documentation>

Exceptions Signalled

kXMPEInvalidPart	Invalid part.
kXMPEInvalidSemanticEvent	Invalid AppleEvent (not created with XMPMessageInterface::CreateEvent).
others	<See AppleEvent Manager documentation>

Pre conditions

A valid AppleEvent created by XMPMessageInterface::CreateEvent is required.

Post conditions

Event will be sent.

XMPNameResolver

Basic Class Documentation

This class encapsulates structured name or object specifier resolution functionality. It works in concert with the XMPSemanticInterface class. The XMPSession object allows access to an instance of this object for any clients who wish to use it. The platform vendor will implement. XMPNameResolver is a subclass of XMPObject

A minimum of documentation is given for functions that have counterparts in the AppleEvent Manager. Only differences are pointed out.
The OSErr return value normally returned by these functions will instead be signalled via an exception should it not be equal to noErr.
Only exceptions that are different from, or exceptions that occur in addition to the normal AppleEvent exceptions, are documented.

Theory of Operation

The functions of this class are direct counterparts to the associated AppleEvent object model functions. A new version of the OSL, one supports "contexts" will be used to implement the member functions of this class. Resolution of object specifiers will take into account XMPParts working as pseudo-applications. This class provides private services tothe XMPMessageInterface class for getting and setting the current context part.

Invariants Maintained by Class

None.

Other Persistent Properties

Member Functions

CallObjectAccessor

Function Prototype

```
virtual void CallObjectAccessor(
        XMPPart* part,
        DescType desiredClass,
        XMPOSLToken* containerToken,
        DescType containerClass,
        DescType keyForm,
        AEDesc* keyData,
        XMPOSLToken* token);
```

Protocol

Content Model

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

See AppleEvent Manager documentation for AECallObjectAccessor. part is the part whose accessor is to be called.

Inputs

part	The part containing desired object accessor.
desiredClass	The object class of the desired Apple event objects.
containerToken	The token that identifies the container for the desired objects.

	containerClass	The object class of the container for the desired objects.
	keyForm	The key form specified by the object specifier record for the object or objects to be located.
	keyData	The key data specified by the object specifier record for the object or objects to be located.
	token	The token returned by the object accessor call.

Outputs

None

Exceptions Signalled

See	AppleEvent Mgr documentation.

Pre conditions

None.

Post conditions

Whatever side effects the invocation of the accessor produces.

CreateSwapToken

Function Prototype

```
virtual CreateSwapToken(
        XMPOSLToken* token,
        XMPFrame* frame);
```

Protocol

Content Model

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Construct an AppleEvent token that encodes an XMPFrame.

Inputs

frame	The XMPFrame represented by this token

Outputs

token	The token constructed.

Exceptions Signalled

kXMPErrOutOfMemory	Out of Memory

Pre conditions

None.

Post conditions

A new token is constructed.

CreateToken

Function Prototype

```
virtual CreateToken(
        XMPOSLToken* token
        DescType type,
        XMPPart* inPart);
```

Protocol

Content Model

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Initialize an AppleEvent token that has an encoded XMPPart context.

Inputs

type	The DescType for the AEDesc of this token
inPart	The XMPPart* specifying the root container for this token

Outputs

token	The initialized token.

Exceptions Signalled

kXMPErrOutOfMemory	Not enough memory to initialize token.

Pre conditions

None.

Post conditions

A new token is constructed.

DisposeToken

Function Prototype

```
virtual void DisposeToken(
        XMPOSLToken* theToken);
```

Protocol

Content Model

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

See AppleEvent Object Model documentation.

Inputs

theToken	The token to dispose.

Outputs

None.

Exceptions Signalled

kXMPErrInvalidPart	Token does not contain a valid part description.

Pre conditions

None.

Post conditions

Token is destroyed.

GetContainingFrame

Function Prototype

```
XMPFrame* GetContainingFrame();
```

Protocol

Semantic Events

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Parts will call from their object accessors to return the frame that was used to find that part during object specifier resolution. Return value may be kXMPNULL if this information is not available. frame is recorded the last time a part called CreateSwapToken. It will be invalidated at the next CreateSwapToken call.

Inputs

None

Outputs

Return	The containing frame of this part.

Exceptions Signalled

None

Pre conditions

CreateSwapToken must have been called for a valid XMPFrame* to be returned.

Post conditions

None.

GetContextInfo

This method is only of interest to Container Application developers.

Function Prototype

```
void GetContextInfo(
        XMPFrame** frame,
        XMPPart** part);
```

Protocol

Content Model

Protection

Public. Meant to be called by the shell only when it receives an AppleEvent with an object specifier in the direct parameter that has not been resolved.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

The shell application can make this call when it receives an AppleEvent that has a direct parameter which is an object specifier. This is a signal to the shell that it is being given a chance to handle an AppleEvent that was not handled by a part. The frame and part parameters represent the frame and part to which the AppleEvent was sent.

Inputs

None.

Outputs

frame	The last destination frame for the AppleEvent.
part	The last destination part for the AppleEvent.

Exceptions Signalled

kXMPErrInvalidPart	Not a valid part

Pre conditions

An AppleEvent and/or the resolution of the object specifier in its direct parameter must have failed.

Post conditions

None.

Purge

Function Prototype

```
XMPSize Purge(
        XMPSize howMuch);
```

Protocol

Content Model, XMPObject

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Free unused memory.

Inputs

howMuch	The memory requested to be freed.

Outputs

Return	The amount of memory that was freed.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

Some memory may have been freed.

Resolve

Function Prototype

```
virtual XMPPart* Resolve(
        XMPObjectSpec* theObject,
        XMPOSLToken* token,
        XMPPart* contextPart);
```

Protocol

Content Model

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

See AppleEvent Object Model documentation for AEResolve. A new type, XMPOSLToken is used for the token instead of an AEDesc. The fields of the struct are exactly the same however. The extra XMPPart* parameter is used to describe the context of the resolution, i.e., the part who is making the call.

Inputs

theObject	The object specifier to be resolved.
contextPart	The part from which the call is being made.

Outputs

token	The final token produced by the resolution.

Exceptions Signalled

kXMPOutOfMemory	Couldn't allocate needed internal structures.
errAENoSuchObject	Invalid XMPPart* or part does not support the Semantic Interface extension.
Others	Parts and the OSL may generate exceptions. These exceptions may be returned by Resolve.

Pre conditions

None.

Post conditions

A new token will have been created for the object described in theObject. Many part-specific routines will be called back to.

XMPNameSpace

Basic Class Documentation

This class represents a name space. Objects of this class are created by member functions of the XMPNameSpaceManager class. The platform vendor will implement. XMPNameSpace is a subclass of XMPObject.

Theory of Operation

This class allows registering XMPPtrs with three different types of keys: XMPISOStr, XMPOSType and XMPSLong. Values are hashed for quick lookup.

Invariants Maintained by Class

There is always a valid hash table instantiated to hold the key/value pairs. It may be empty.

Other Persistent Properties

Member Functions

Exists

Function Prototype

```
virtual XMPPtr Exists(
        XMPISOStr key);
```

Protocol

Name Binding

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns boolean for the existance of the given key.

Inputs

key	The key for the entry requested.

Outputs

Boolean	True if key exists.

Exceptions Signalled

None

Pre conditions

None.

Post conditions

None.

GetName

Function Prototype

```
virtual XMPISOStr GetName();
```

Protocol

Name Binding

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return the ISO string assigned to this object at initialization time.

Inputs

None

Outputs

Return — name of this name space.

Exceptions Signalled

None

Pre conditions

None.

Post conditions

None.

GetValue

Function Prototype

```
virtual XMPPtr GetValue(
        XMPISOStr key);
```

Protocol

Name Binding

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the XMPPtr corresponding to the given key

Inputs

key — The key for the entry requested.

Outputs

XMPPtr — The entry.

Exceptions Signalled

None

Pre conditions

None.

Post conditions

None.

Purge

Function Prototype

```
XMPSize Purge(
        XMPSize howMuch);
```

Protocol

Name Binding, XMPObject

Protection

Public.

Override policy

Derived class **can** override.

Derived class **must** call base class behavior, during derived class behavior.

Basic operation

Frees up unused memory.

Inputs

howMuch — The amount of memory to purge.

Outputs

Return — The amount of memory that was freed.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

Some memory may be freed for use.

Register

Function Prototype

```
virtual void Register(
        XMPISOStr key,
        XMPPtr value,
        XMPULong valueLength);
```

Protocol

Name Binding

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Register an XMPPtr given a key.

Inputs

key	The key to use for this XMPPtr
value	Client-defined pointer value.
valueLength	Size of value

Outputs

None

Exceptions Signalled

kXMPErrOutOfMemory	Not enough memory.
kXMPErrWrongType	Request for type other than type with which this name space was originally created.
kXMPErrInvalidValue	value was kXMPNULL.

Pre conditions

None.

Post conditions

The XMPPtr is registered under the key.

Unregister

Function Prototype

```
virtual void Unregister(
        XMPISOStr key);
```

Protocol

Name Binding

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Removes the given entry from the XMPNameSpace.

Inputs

key	Key for the entry to be removed.

Outputs

None

Exceptions Signalled

None

Pre conditions

The given entry must have been registered. No error is signalled, however, if the key does not already exist.

Post conditions

The given entry will be removed.

XMPNameSpaceManager

Basic Class Documentation

This class represents a a bottleneck for the creation of XMPNameSpaces to be used by the XMPBinding object. It keeps track of XMPNameSpaces created and destroyed so that they may later be found by name. Objects of this class are created by member functions of the XMPSession class and by the XMPMacSystemSession class. The platform vendor will implement. XMPNameSpaceManager is a subclass of XMPObject.

Theory of Operation

This class allows inheritance of XMPNameSpaces so that a search can take place in more than one XMPNameSpace.

Invariants Maintained by Class

A valid list of XMPNameSpaces (possibly empty) exists.

Other Persistent Properties

Member Functions

CreateNameSpace

Function Prototype

```
virtual XMPNameSpace* CreateNameSpace(
        XMPISOStr spaceName,
        XMPNameSpace* inheritsFrom,
        XMPULong numExpectedEntries);
```

Protocol

Name Binding

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Creates an XMPNameSpace object with specified parameters. An exception is signalled if an XMPNameSpace with the given name already exists.

Inputs

spaceName	The name the caller wishes to give to this XMPNameSpace
inheritsFrom	An XMPNameSpace to search if a GetValue fails in this one.
numExpectedEntries	The number of expected entries to be made in this XMPNameSpace.

Outputs

Return	Reference to XMPNameSpace created.

Exceptions Signalled

kXMPErrOutOfMemory	Not enough memory
kXMPErrKeyAlreadyExists	An XMPNameSpace with that name already exists.

Pre conditions

None.

Post conditions

A new XMPNameSpace object is created and recorded in a list.

DeleteNameSpace

Function Prototype

```
virtual void DeleteNameSpace(
        XMPNameSpace* theNameSpace);
```

Protocol

Name Binding

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Delete specificied XMPNameSpace object.

Inputs

theNameSpace	The XMPNameSpace to be destroyed

Outputs

None

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

The specified XMPNameSpace object will be destroyed.

HasNameSpace

Function Prototype

```
virtual XMPBoolean HasNameSpace(
        XMPISOStr spaceName);
```

Protocol

Name Binding

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Finds out whether a given XMPNameSpace exists.

Inputs

spaceName	the name of the XMPNameSpace to find.

Outputs

Return	If true, the XMPNameSpace exists.

Exceptions Signalled

None

Pre conditions

None.

Post conditions

None.

Purge

Function Prototype

```
XMPSize Purge(
        XMPSize howMuch);
```

Protocol

Name Binding, XMPObject

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Free unused memory

Inputs

howMuch	The amount request to free

Outputs

Return	The amount of memory that was able to be freed.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

Some memory may be freed.

XMPObject

Basic Class Documentation

This class is the common base class for most XMP classes. It provides a general mechanism for extensibility through Extension objects. XMPObject also provides a general memory recovery system with the Purge method.

Theory of Operation

XMPObject does not have any real operation in it, for the most part it serves as a template.

Invariants Maintained by Class

XMPObject can only be initialized once. If InitObject() is called again after the object is initialized, this method is a no-op and will return immediately. Any subclass of XMPObject should respect this behavior as all the subclasses should call their base classes in their Init methods.

Other Persistent Properties

Member Functions

XMPObject

Function Prototype

XMPObject();

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **must** call base class behavior, before derived class behavior.

Basic operation

Constructor of the class.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

this is an uninitialized XMPObject object.

~XMPObject

Function Prototype

~XMPObject();

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **must** call base class behavior, after derived class behavior.

Basic operation

Destructor of the class.

Inputs
None.
Outputs
None.
Exceptions Signalled
None.

Pre conditions
None.

Post conditions
None.

GetExtension

Function Prototype

```
XMPExtension* GetExtension(
        XMPName* extensionName);
```

Protocol
Extensions

Protection
Public.

Override policy
Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation
GetExtension is intended to return the extension object with name extensionName. If the object does not have an extension object with name extensionName then an exception is raised.
XMPObject has no extensions and always raises an exception. Derived classes should return appropriate Extension named extensionName or call the inherited method.

Inputs

extensionName	name of extension to get

Outputs

<return>	the requested Extension

Exceptions Signalled

kXMPErrInvalidExtension	not an extension that this object knows about

Pre conditions
this object has extension extensionName

Post conditions
return = extension extensionName

HasExtension

Function Prototype

```
XMPBoolean HasExtension(
        XMPName* extensionName);
```

Protocol
Extensions

Protection
Public.

Override policy
Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation
HasExtension returns whether or not the object has an extension with the name extensionName. XMPObject has no inherent Extensions of its own, and always

returns false. Derived classes should return whether or not they return a extension named extensionName.

Inputs

extensionName	name of extension to look for

Outputs

<return>	whether or not this object has the given extension

Exceptions Signalled

none

Pre conditions

extensionName is an Extension which this object has

Post conditions

None.

InitObject

Function Prototype

```
void   InitObject();
```

Protocol

Initialization

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

InitObject performs initialization operations for an object. InitObject is always called on an XMPObject immediately after the constructor is called. Derived classes should do whatever is necessary to set up the internal state of their objects to be consistent with class invariants.

If this method is called again after the first time on the same object, it will return immediately.

Inputs

None.

Outputs

none

Exceptions Signalled

none

Pre conditions

None.

Post conditions

this is an initialized XMPObject object.

Purge

Function Prototype

```
XMPSize Purge(
        XMPSize size);
```

Protocol

Memory Management

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Purge is intended for a class to free up memory including caches until it has freed up size or more bytes or freed up all the memory it can. The number of bytes actually freed should be returned. XMPObject does not allocate any memory and always return 0 bytes purged.

Inputs

size	the number of bytes to try to purge

Outputs

<return>	the number of bytes actually purged

Exceptions Signalled

none

Pre conditions

None.

Post conditions

At least size or as many as possible bytes have been freed up by the object

ReleaseExtension

Function Prototype

```
void   ReleaseExtension(
           XMPExtension* extension);
```

Protocol

Extensions

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

ReleaseExtension is intended to release the extension object passed in. If the extension passed in is not one which was acquired from the object using GetExtension then an exception is raised.
XMPObject has no extensions and always raises an exception. Derived classes should expect an extension which was previously acquired using GetExtension, and release it.

Inputs

extension	the extension to be released

Outputs

none

Exceptions Signalled

kXMPErrInvalidExtension	not an extension that this object knows about

Pre conditions

this object has given out the given extension

Post conditions

the given extension is no longer guaranteed to be valid

XMPPart

Basic Class Documentation

Ancestors: XMPPart -> XMPPersistentObject -> XMPRefCountObject -> XMPObject.
Descendants: any developer part editor implementation.

XMPPart is the abstract base class from which Part developers will derive the classes which implement their Part Editors. There can be no instances of XMPPart itself, but there will be instances of derived classes.

XMPPart objectifies the structure and behavior of Parts. Each Part in a Document is represented by one instance of an XMPPart. The methods defined in XMPPart must be implemented by all Part Handlers. One Part Handler supplies the implementation of behavior for all Parts of a given type.

Parts participate in many of the OpenDoc protocols. Devolopers of Part handlers must implement behavior for the Part's roles in all those behaviors. This provides a base structure for Parts, on which developers will construct the functionality of the Part's content. There are quite a few entry points into a Part, but many of these are fairly simple to implement.

Theory of Operation

Invariants Maintained by Class

The fSU field (inherited from XMPPersistentObject) of an XMPPart is initialized to point to the Part's storage unit. This field is never nil, and cannot be changed to point at a different storage unit.

Each Part must maintain a list of all frames in which it is displayed. The representation of this list is hidden by the API, and developers may choose whatever format works best for their implementation.

A Part which can contain other parts must maintain a list of all frames embedded within it. This list contains references to frames displaying all the parts embedded in the containing part. The representation of this list is hidden by the API, and developers may choose whatever format works best for their implementation.

Parts which support extensions, including a semantic interface, must manage the creation and deletion of their extension objects.

Other Persistent Properties

Member Functions

AttachSourceFrame

Function Prototype

```
void AttachSourceFrame(
        XMPFrame* frame,
        XMPFrame* sourceFrame);
```

Protocol

Frames

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Called by the object which requested creation of a frame, immediately after creation of the frame. This associates a "source frame" with a display frame of a part. This tells the part to keep two or more of its display frames synchronized.

A part will recieve this call just after a display frame has been added. Attaching a source frame should cause the display frame to look just like it, or if presentations differ, to be equivalent to it. This will cause duplication of embedded frames, and will ensure that the view in one frame is updated when content in the other is changed.

Inputs

frame	A display frame of the part.
sourceFrame	Another display frame of the part.

Outputs

None.

Exceptions Signalled

kXMPErrInvalidFrame	Either frame or sourceFrame is not a display frame of this part.

Pre conditions

Both frame and sourceFrame are display frames of the part.

Post conditions

None.

ChangeKind

Function Prototype

```
void ChangeKind(
        XMPType kind);
```

Protocol

Binding

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Ask a part to change into a new kind of content. For instance, ask an ASCII Text part to change into a Styled Text part.

The part should begin using the given kind as its primary kind if it is supported. The first type of the first value in the contents property of the part's storage unit should become the given kind.

Inputs

kind	The new kind for the part.

Outputs

None.

Exceptions Signalled

kXMPErrInvalidKind	Part does not support that kind of content.

Pre conditions

The part supports kind 'kind'.

Post conditions

The part's primary kind is 'kind'.
The first type of the first value in the contents property of the part's storage unit is 'kind'.
When 'ternalized, the part uses this representation.

CloneInto

Function Prototype

```
void CloneInto(
        XMPDraftKey key,
        XMPStorageUnit* storageUnit,
        XMPStorageUnit* initiatingFrame);
```

Protocol

Storage

Protection

Public. Mostly called by XMPFrame::CloneTo.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Externalizes this XMPPart into storageUnit.
In order to avoid copying extra Storage Units during the deep-copy, an initiatingFrameSU may be specified. If initiatingFrameSU is not kXMPNULL, only those Storage Units reachable from initiatingFrameSU will be copied. If initiatingFrameSU is kXMPNULL, all the Storage Units reachable from this Storage Unit will be copied.
Note that the actual copying may not be completed until after EndClone is finished.

Inputs

key	XMPDraftKey identifying the Clone transaction
storageUnit	XMPStorageUnit referring to the destination Storage Unit.
initiatingFrame	initiatingFrame

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

ContainingPartPropertiesChanged

Function Prototype

```
void ContainingPartPropertiesChanged(
        XMPFrame* frame,
        XMPStorageUnit* propertyUnit);
```

Protocol

Embedding

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Used by my ContainingPart to inform this part of changes to content properties of its proxy for this part.

Inspect "propertyUnit" for properties this part can understand. Where applicable, incorporate those properties into part's conent data. Ignore inapplicable properties, without signalling an error.

Inputs

frame	frame
propertyUnit	propertyUnit

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

CreateEmbeddedFrame

Function Prototype

```
XMPFrame* CreateEmbeddedFrame(
        XMPFrame* containingFrame,
        XMPShape* frameShape,
        XMPTransform* externalTransform,
        XMPPart* embedPart,
        XMPTypeToken viewType,
        XMPTypeToken presentation,
        XMPID frameGroupID,
        XMPBoolean isOverlaid);
```

Protocol

Frames

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Ask the part to create a new frame and embed a part in it.

If this part is a containing part, it should ask the draft to create a new frame, and embed the frame in its content. The containing part decides if it can fulfill the requests for frameShape and externalTransform - if not, it can create the frame where it wants and with the shape it chooses. If the frameGroupID param is zero, then the part should assign the frame a new, unique ID. If isOverlaid is true, the new frame should float above the part's content, and should not have to negotiate for space with the part. The viewType and presentation are just passed through to Draft::CreateFrame().

Inputs

containingFrame	The display frame of this part in which to embed the new frame.
frameShape	The requested shape of the new frame. In the frame's own coordinate space.
externalTransform	The requested location for the new frame in the part's conent.
embedPart	The part to embed in the new frame.

	viewType	View type for new frame.
	presentation	Presentation for new frame.
	frameGroupID	The ID of another embedded frame of the part, or zero to make a new one.
	isOverlaid	Whether the new frame should float above the part's content.

Outputs

<return>	The new frame.

Exceptions Signalled

kXMPErrCannotEmbed	This part does not support embedding.
kXMPErrOutOfMemory	Not enough memory to embed a part.

Pre conditions

None.

Post conditions

None.

CreateEmbeddedFramesIterator

Function Prototype

```
XMPEmbeddedFramesIterator* CreateEmbeddedFramesIterator();
```

Protocol

Embedding

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Create an object which will iterate all the embedded frames of this part.

Inputs

None.

Outputs

<return>	The iterator.

Exceptions Signalled

kXMPErrOutOfMemory	Not enough memory to allocate iterator.
kXMPErrCannotEmbed	This part does not support embedding.

Pre conditions

None.

Post conditions

None.

CreateLink

Function Prototype

```
XMPLink* CreateLink(
        XMPPtr data,
        XMPULong size);
```

Protocol

Linking

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

If a link already exists to the content identified by the data and size arguments, return the link object; otherwise, create a new link object, put in the initial data and return it to the caller.

Identify the content to be linked (by resolving the object specifier saved in the data parameter, or by some other means). Create a link object to represent the content data. The part must maintain information about what portion of its contents have been linked to so that it may notify link clients when that data has been changed. The link created is returned to the caller.

Inputs

data	A description of the content object to link to, as stored by this part in a link spec.
size	Size of the data in bytes.

Outputs

<return>	The link object.

Exceptions Signalled

kXMPErrCantLink	Part does not support linking protocol.
kXMPErrInvalidLinkData	Could not identify content for link.
kXMPErrOutOfMemory	Not enough memory to allocate link object.

Pre conditions

The data identifies some portion of this part's contents.

Post conditions

This part maintains a link to the identified content.

EmbeddedFrameSpec

Function Prototype

```
void EmbeddedFrameSpec(
        XMPFrame* embeddedFrame,
        XMPObjectSpec* spec);
```

Protocol

Embedding

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Create an object specifier for the embedded frame.

If this part is itself embedded, ask its containing part for the specifier for the part's display frame, then concatenate the specifier for the embedded frame to that.

Inputs

embeddedFrame	The frame to create a specifier for.

Outputs

spec	The frame's specifier.

Exceptions Signalled

kXMPErrInvalidFrame	embeddedFrame is not an embedded frame of this part.

Pre conditions

None.

Post conditions

None.

FocusAcquired

This method is only of interest to Container Application developers.

Function Prototype

```
void FocusAcquired(
        XMPTypeToken focus,
        XMPFrame* ownerFrame);
```

Protocol

Part Activation

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

A notification to the part that one of its frames has acquired the specified focus. Called when a document is opened, for example.

The part makes whatever response is necessary for acquiring the focus. For instance, acquiring the selection focus might cause a part to highlight its text selection.

Inputs

focus	The focus type to acquire.
ownerFrame	The frame which gets the focus.

Outputs

None.

Exceptions Signalled

kXMPErrInvalidFocus	Part cannot own this focus.
kXMPErrInvalidFrame	ownerFrame is not a display frame of this part.

Pre conditions

None.

Post conditions

None.

FocusLost

This method is only of interest to Container Application developers.

Function Prototype

```
void FocusLost(
        XMPTypeToken focus,
        XMPFrame* ownerFrame);
```

Protocol

Part Activation

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Notify the part that a focus has unexpectedly been lost.

Take whatever actions necessary to recover from having lost the focus, i.e. close connections, de-highlight selection, etc.

Inputs

focus	The focus which was lost.
ownerFrame	The frame from which the focus was lost.

Outputs

None.

Exceptions Signalled

kXMPErrInvalidFocus	Part doesn't own this focus.
kXMPErrInvalidFrame	ownerFrame is not a display frame of this part.

Pre conditions

None.

Post conditions

None.

GetContainingPartProperties

Function Prototype

```
XMPStorageUnit* GetContainingPartProperties(
        XMPFrame* frame);
```

Protocol

Embedding

Protection

Public. Called by an embedded part.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Used by an embedded Part to ask this Containing Part of changes to content properties of the proxy for it.

Create a storage unit, write properties containing part associates with the embedded frame into it, and return it.

Inputs

frame	An embedded frame of this part.

Outputs

<return>	A storage unit containing the container properties for the specified frame. Caller has responsibility for deallocating storage.

Exceptions Signalled

kXMPErrCannotEmbed	This part does not support embedding.
kXMPErrInvalidFrame	frame is not an embedded frame of this part.

Pre conditions

None.

Post conditions

None.

GetPrintResolution

Function Prototype

```
XMPULong GetPrintResolution(
        XMPFrame* frame);
```

Protocol

Imaging

Protection

Public. Called by part or app performing printing.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Return the minimum desired resolution in dots per inch this part requires for printing the contents of the specified frame.

Inputs

frame	The frame for which the resolution is needed.

Outputs

<return>	The desired resolution in dots per inch.

Exceptions Signalled

kXMPErrInvalidFrame	frame is not a display frame of this part.

Pre conditions

None.

Post conditions

None.

Open

Function Prototype

```
XMPID Open(
        XMPFrame* frame);
```

Protocol

Frames

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Open a presentation of the part in a frame in a new window.

Create and open a new window. Add this part as the root part of the window. Base the presentation in the new frame on that in the old frame, or a default one if none.

Inputs

frame	The frame which should be used as a model for the new presentation. kXMPNULL means the part should create a new frame and open it in the window.

Outputs

<return>	A unique id for the new window.

Exceptions Signalled

kXMPErrInvalidFrame	"frame" isn't one of my display frames.

Pre conditions

None.

Post conditions

None.

RemoveEmbeddedFrame

Function Prototype

```
void RemoveEmbeddedFrame(
        XMPFrame* embeddedFrame);
```

Protocol

Frames

Protection

Public. Called by embedded part.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Remove a frame viewing an embedded part.

Called by an embedded part to indicate it no longer needs the frame to display itself.

Inputs

embeddedFrame	embeddedFrame

Outputs

None.

Exceptions Signalled

kXMPErrCannotEmbed	This part does not support embedding.
kXMPErrInvalidFrame	frame is not an embedded frame of this part.

Pre conditions

None.

Post conditions

None.

RevealFrame

Function Prototype

```
void RevealFrame(
        XMPFrame* embeddedFrame);
```

Protocol

Embedding

Protection

Public. Called by embedded part.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Ask a part to make an embedded frame visible.

Scroll one of this part's display frame's to make the embedded frame visible therein. If this part has no visible frames, ask a containing part to reveal one of them. If no display frames for the part currently exist, or if this part's containing frame can't reveal the display frame, open a frame in a new window.

Inputs

embeddedFrame	An embedded frame of this part.

Outputs

None.

Exceptions Signalled

kXMPErrInvalidFrame	frame is not an embedded frame of this part.

Pre conditions

None.

Post conditions

None.

XMPPersistentObject

Basic Class Documentation

There are a number of XMP objects which require persistent storage (XMPFrame, XMPPart, XMPLink and XMPSourceLink). This functionality has been commonalized and centralized in the XMPPersistentObject class and all these classes are subclasses of XMPPersistentObject. All the objects from these classes are collectively known as persistent objects.

Part developers should not subclass this class. They should use XMPStorageUnits to store their persistent data. Platform developers may subclass this class to create persistent objects.

Theory of Operation

Every XMPPersistentObject has a XMPStorageUnit in which it stores itself persistently.
When a persistent object is created for the first time, an InitXXX method is called on it (where XXX is the name of the class, e.g., InitPart). In XMPXXX::InitXXX, the persistent object calls XMPPersistentObject::InitPersistentObject to initialize the XMPPersistentObject and then creates all its properties and values in the XMPStorageUnit.
When Externalize is called on the persistent object, it should write out its persistent state to the properties and values it created during InitXXX.
When InitXXXFromStorage is called on the persistent object, it should read in its persistent state from the properties and values in the XMPStorageUnit.
Note that a persistent object should Externalize only the data which it requires for InitXXXFromStorage. Therefore, any caches or anything else which can be derived or regenerated should not be externalized.

Invariants Maintained by Class

InitPersistentObjectFromStorage is called only once on XMPPersistentObject, when it is first created.
After that, every time the XMPPersistentObject is instantiated in memory, InitPersistentObject is called once.

Other Persistent Properties

kXMPPropCreateDate of type kXMPTime_T
kXMPPropModDate of type kXMPTime_T
kXMPPropModUser of type kXMPAppleTEXT

Member Functions

XMPPersistentObject

Function Prototype

```
XMPPersistentObject(
        XMPStorageUnit* storageUnit);
```

Protocol

None.

Protection

Public. Parts and Container Apps do not call this method directly. But whenever the Part or the Container App creates a persistent object (e.g. , Frame), this method is called.

Override policy

Derived class **can** override.

Derived class **must** call base class behavior, before derived class behavior.

Basic operation

set fSU = storageUnit

Inputs

None.

Outputs

none

Exceptions Signalled

none

Pre conditions

None.

Post conditions

fSU = storageUnit

~XMPPersistentObject

Function Prototype

~XMPPersistentObject();

Protocol

None.

Protection

Public. Parts and Container Apps do not call this method directly. But whenever a persistent object is deleted, this method is called.

Override policy

Derived class **can** override.

Derived class **must** call base class behavior, after derived class behavior.

Basic operation

release the object's storageUnit

Inputs

none

Outputs

none

Exceptions Signalled

none

Pre conditions

None.

Post conditions

None.

Externalize

Function Prototype

void Externalize();

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **can** override.

Derived class **must** call base class behavior, before derived class behavior.

Basic operation

Updates the moddate and modifiedUser properties of the persistent object.
A derived class should write out its persistent state to the properties and values it created during InitXXX.

Inputs

none

Outputs

none

Exceptions Signalled

none

Pre conditions

None.

Post conditions

None.

GetID

Function Prototype

XMPID GetID();

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the ID of this object.
Note that this ID is not persistent. Therefore, the same persistent object may have a different ID in every session.

Inputs

none

Outputs

<return>	ID of this object.

Exceptions Signalled

none

Pre conditions

None.

Post conditions

None.

GetStorageUnit

Function Prototype

XMPStorageUnit* GetStorageUnit();

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

returns the XMPStorageUnit associated with the persistent object

Inputs

none

Outputs

<return>	XMPStorageUnit associated with this object.

Exceptions Signalled

none

Pre conditions

None.

Post conditions

None.

InitPersistentObject

Function Prototype

```
void InitPersistentObject(
        XMPStorageUnit* storageUnit);
```

Protocol

None.

Protection

Public. Parts and Container Apps do not call this method directly. But whenever the Part or the Container App creates a persistent object (e.g. , Frame), this method is called.

Override policy

Derived class **must** override.

Derived class **must** call base class behavior, before derived class behavior.

Basic operation

Creates and sets the creationDate, modDate and modifiedUser properties of the persistent object. This method is called once on a persistent object, at the beginning of its persistent lifetime.
A derived class must call this method in its InitXXX method.

Inputs

storageUnit	the storageUnit of this persistent object

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

InitPersistentObjectFromStorage

Function Prototype

```
void InitPersistentObjectFromStorage(
        XMPStorageUnit* storageUnit);
```

Protocol

ObjectStorage

Protection

Public. Parts and Container Apps do not call this method directly. But whenever the Part or the Container App creates a persistent object (e.g. , Frame), this method is called.

Override policy

Derived class **must** override.

Derived class **must** call base class behavior, before derived class behavior.

Basic operation

Initializes the internal fields of this object.
A derived class must call this method in its InitXXXFromStorage method.

Inputs

storageUnit	XMPStorageUnit of this object

Outputs

none

Exceptions Signalled

none

Pre conditions

None.

Post conditions

None.

XMPPlatformTypeSet

Basic Class Documentation

XMPPlatformTypeSet is a collection class of XMPPlatformType objects with set semantics. XMPPlatformTypeSet has no base class.

Theory of Operation

XMPPlatformTypeSet is a collection of XMPPlatformTypes. Users of this class can add and remove types , query the existence of a type, and get the number of types in the class.

Parts may create instances of XMPPlatformTypeSet. Subclassing of XMPPlatformTypeSet is not anticipated.

The main client of this class are XMPClipboard and parts. The XMPClipboard class uses XMPPlatformTypeSet as parameters to its functions. Parts can use XMPPlatformTypeSet to store a set of Platform Types for repeated use.

XMPPlatformTypeSet uses OpenDoc's private collection class.

Invariants Maintained by Class

Every item in a XMPPlatformTypeSet object is unique. Therefore, a XMPPlatformTypeSet object contains a set of different XMPPlatformTypes.
A XMPPlatformTypeSet object can contain zero or more XMPPlatformTypes. In other words, the type set can either be empty or non-empty.
There is no particular ordering of XMPPlatformTypes in a XMPPlatformTypeSet object.

Other Persistent Properties

Member Functions

XMPPlatformTypeSet

Function Prototype

XMPPlatformTypeSet();

Protocol

Creation and Deletion

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **must** call base class behavior, before derived class behavior.

Basic operation

Class instance constructor.

Inputs

None.

Outputs

None.

Exceptions Signalled

kXMPErrOutOfMemory Could not construct this instance.

Pre conditions

None.

Post conditions

None.

~XMPPlatformTypeSet

Function Prototype

~XMPPlatformTypeSet();

Protocol

Creation and Deletion

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Class instance destructor.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

Add

Function Prototype

void Add(
XMPPlatformType type);

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Adds the argument platform type to the set. If the platform type is already present in the set, no action is taken.

Inputs

type	Platform type to be added.

Outputs

None.

Exceptions Signalled

kXMPErrOutOfMemory	Could not add the argument platform type to the set.

Pre conditions

None.

Post conditions

The set contains the argument platform type.

Contains

Function Prototype

XMPBoolean Contains(
XMPPlatformType type);

Protocol

None.

Protection
Public.
Override policy
Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.
Basic operation
Test this set for the presence of the argument platform type.
Inputs

type	A platform type to test for inclusion in this set.

Outputs

<return>	kXMPTrue if the argument platform type is in the set and kXMPFalse otherwise.

Exceptions Signalled
None.
Pre conditions
None.
Post conditions
None.

Count

Function Prototype
XMPULong Count();
Protocol
None.
Protection
Public.
Override policy
Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.
Basic operation
This function returns the number of platform types in the set. If the set is empty, zero is returned.
Inputs
None.
Outputs

<return>	Number of platform types in the set.

Exceptions Signalled
None.
Pre conditions
None.
Post conditions
None.

CreatePlatformTypeSetIterator

Function Prototype
XMPPlatformTypeSetIterator* CreatePlatformTypeSetIterator();
Protocol
Iteration
Protection
Public.
Override policy
Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return an iterator for this set. The client should dispose of the iterator using the delete operator.

Inputs

None.

Outputs

<return>	An iterator for this set.

Exceptions Signalled

kXMPErrOutOfMemory	Could not create the iterator.

Pre conditions

None.

Post conditions

None.

Difference

Function Prototype

```
void Difference(
        XMPPlatformTypeSet* typeSet);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Modify this object to be the set difference between this object and the argument set; that is, exclude all platform types in the argument typeSet from this set. Platform types in the argument set which are absent from this set are ignored.

Inputs

typeSet	The set of platform types to be removed.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

This set does not contain any platform types in the argument set.

Remove

Function Prototype

```
void Remove(
        XMPPlatformType type);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Removes the argument platform type from the set. If the type is not in the set, no action is taken.

Inputs

type	Platform type to be removed.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

The argument platform type is not in the set.

Union

Function Prototype

```
void Union(
        XMPPlatformTypeSet* typeSet);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Modify this object to be the set union of all platform types in this object and the argument set; that is, add all platform types in the argument typeSet not already present in this set. If an exception is thrown, this set is left in an indeterminate state.

Inputs

typeSet	The set of platform types to be added.

Outputs

None.

Exceptions Signalled

kXMPErrOutOfMemory	Cannot form the set union with the argument set.

Pre conditions

None.

Post conditions

This set contains every type in the argument set.

XMPPlatformTypeSetIterator

Basic Class Documentation

XMPPlatformTypeSetIterator is an iterator for the class XMPPlatformTypeSet. XMPPlatformTypeSetIterator has no base class.

Theory of Operation

This class is used to iterate over the elements of a XMPPlatformTypeSet instance. Iteration is forward only. The iteration can be restarted at any time by calling the First method. If the XMPPlatformTypeSet instance is modified during the iteration, First or Next will throw an exception. When an exception is thrown, the iterator cannot be used further, and should be destroyed.

Parts should create an XMPPlatformTypeSetIterator by calling the CreatePlatformTypeSetIterator method of the XMPPlatformTypeSet instance to be iterated over. The iterator should be disposed using the delete operator.

Subclassing of XMPPlatformTypeSetIterator is not anticipated.

The implementation of XMPPlatformTypeSetIterator uses OpenDoc's private collection class.

Invariants Maintained by Class

Other Persistent Properties

Member Functions

~XMPPlatformTypeSetIterator

Function Prototype

~XMPPlatformTypeSetIterator();

Protocol

Creation and Deletion

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Class instance destructor.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

First

Function Prototype

XMPPlatformType First();

Protocol

Iteration

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Initiates iteration and returns an arbitrary first element from the set. After creating an iterator using XMPPlatformTypeSet::CreatePlatformTypeSetIterator, call this method once, then call Next repeatedly to iterate through all elements of the set. First can be called at any time to restart the iteration.

Inputs

None.

Outputs

<return>	An arbitrary XMPPlatformType object in the set. If the set is empty kXMPNULL is returned, although the client should call IsNotComplete to test for completion.

Exceptions Signalled

kXMPErrIteratorOutOfSync	The XMPPlatformTypeSet instance was changed during the iteration.

Pre conditions

None.

Post conditions

None.

IsNotComplete

Function Prototype

```
XMPBoolean IsNotComplete();
```

Protocol

Iteration

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Test if iteration is complete.

Inputs

None.

Outputs

<return>	Returns kXMPTrue if the last call to First or Next returned a valid XMPPlatformType element, and kXMPFalse otherwise.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

Next

Function Prototype

```
XMPPlatformType Next();
```

Protocol

Iteration

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns an arbitrary next element from the set. First should be called prior to this routine to start the iteration. Next can be called repeatedly until IsNotComplete returns kXMPTrue.

Inputs

None.

Outputs

<return>	An arbitrary element from the set not yet returned by this method nor by First since the iteration was started. If the iteration is complete kXMPNULL is returned, although the client should call IsNotComplete to test for completion.

Exceptions Signalled

kXMPErrIteratorOutOfSync	The XMPPlatformTypeSet instance was changed during the iteration.

Pre conditions

First has been called to start the iteration.

Post conditions

None.

XMPRefCntObject

Basic Class Documentation

This is the base class for all XMP classes which require reference counting. Often it is convenient to share one copy of an object instead of making multiple copies of the object. When an object is shared, there needs to be a mechanism for tracking the number of references to it so that the object can be properly disposed of when there are no references to it. XMPRefCntObject provides this general functionality by keeping track of the number of references to every instance.

Many OpenDoc classes are subclasses of XMPRefCntObject -- XMPContainer, XMPDocument, XMPDraft, XMPStorageUnit and XMPWindow. However, the clients of these classes do not have to deal with the creation and destruction of the XMPRefCntObject as these operations are hidden in the factory class. For example, in order to create a new XMPContainer object, the client should call XMPStorageSystem::CreateContainer. XMPStorageSystem::CreateContainer then makes the appropriate calls to instantiate and initialize the XMPContainer object.

XMPRefCntObject can be subclassed by part developers or platform developers. However, this class only handles ref-counting. The actual garbage collection is done by the factory object which instantiates this XMPRefCntObject. For example, when the refcount of a XMPContainer goes down to 0, XMPStorageSystem is notified (through XMPStorageSystem::ReleaseContainer) so that it can delete the XMPContainer.

Theory of Operation

Invariants Maintained by Class

The reference count of any XMPRefCntObject object is always >=0.
Any XMPRefCntObject is valid (i.e., its pointer can be used) if its refCount > 0.

Other Persistent Properties

Member Functions

XMPRefCntObject

Function Prototype

XMPRefCntObject();

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **must** call base class behavior, before derived class behavior.

Basic operation

Creates the object.

Inputs

none

Outputs

none

Exceptions Signalled

none

Pre conditions

None.

Post conditions

None.

~XMPRefCntObject

Function Prototype

~XMPRefCntObject();

Protocol

Cleanup

Protection

Public.

Override policy

Derived class **can** override.

Derived class **must** call base class behavior, after derived class behavior.

Basic operation

Deletes this object.

Inputs

none

Outputs

none

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

IncrementRefCount

Function Prototype

void IncrementRefCount();

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **can** override.

Derived class **must** call base class behavior, before derived class behavior.

Basic operation

Increment the reference count of this object by 1.

Inputs

none

Outputs

none

Exceptions Signalled

none

Pre conditions

reference count of this object = x

Post conditions

reference count of this object = x+1

InitRefCntObject

Function Prototype

void InitRefCntObject();

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **must** call base class behavior, before derived class behavior.

Basic operation

Initializes this object and sets its refCount to 1.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

refCount of this object == 1

Release

Function Prototype

```
void Release();
```

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **can** override.

Derived class **must** call base class behavior, after derived class behavior.

Basic operation

Decrements its reference count by 1.

Inputs

none

Outputs

none

Exceptions Signalled

kXMPErrZeroRefCount	The reference count cannot be decremented because the reference count is already 0.

Pre conditions

reference count of this object = x, where x>0

Post conditions

reference count of this object = x-1

XMPSemanticInterface

Basic Class Documentation

This class encapsulates the functionality necessary to install semantic event handlers and object accessors. These calls resemble their AppleEvent and AppleEvent object model counterparts very closely. An XMPSemanticInterface object can be thought as a handler table to be used by OpenDoc when processing semantic events. Every XMPPart instance must own one of these, and the XMPSession object keeps an instance for the application shell and owns the "system" instance as well. The platform vendor will implement.

The member functions of this class work similarly to the corresponding "AE<MemberFunctionName>" functions of the AppleEvent manager and the Object Support Library.

XMPSemanticInterface is an extension class and descends from XMPExtension

A minimum of documentation is given for functions that have counterparts in the AppleEvent Manager. Only differences are pointed out.

The isSysHandler parameter is missing from all these routines. Instead, clients wishing to install system callback functions should call the appropriate functions of the XMPSession's XMPSemanticInterface object.

The OSErr return value normally returned by these functions will instead be signalled via an exception should it not be equal to noErr.

Only exceptions that are different from, or exceptions that occur in addition to the normal AppleEvent exceptions, are documented.

Theory of Operation

XMPSemanticInterface interacts primarily with XMPMessageInterface, XMPNameResolver, XMPSession, and XMPPar.

Each XMPPart object and the XMPSession will instantiate one or more of these for its own use. The owning object will install the necessary AppleEvent handlers and object accessor routines required to access its content model and to respond to the required AppleEvents.

The application shell will use the XMPSemanticInterface returned by XMPSession::GetShellSemanticInterface. Any client wishing to alter the system semantic interface should use the XMPSemanticInterface returned by XMPSession::GetSemanticInterface

To send an AppleEvent to a part, the sending part calls XMPMessageInterface::Send with an AppleEvent whose direct parameter is an object specifier that contains some description of the receiving part. The MessageInterface object will resolve the direct parameter to identify the destination part and then use that part's XMPSemanticInterface object to dispatch the event. The XMPMessageInterface object may need to use the XMPSemanticInterface object of a number of parts if the resolution of the object specifier reveals the final part destination is an embedded part. The XMPSemanticInterface object will walk down the hierarchy of parts from the root part to the final destination part.

A part must be able to identify any of its embedded parts if requested. This is how we locate an embedded part if one is specified in the object specifier.

Callback functions have been changed to accomodate parts. All such functions take an XMPPart* as a first parameter. This value will be NULL for the system and document XMPSemanticInterface objects. It will contain a pointer to the particular instance of XMPPart (or a subclass) to which the XMPSemanticInterface object belongs.

Invariants Maintained by Class

None.

Other Persistent Properties

Member Functions

XMPSemanticInterface

Function Prototype

```
XMPSemanticInterface(
        XMPObject* base);
```

Protocol

Semantic Events

Protection

Public. Only parts should create XMPSemanticInterfaces. The application shell should obtain its copy throughthe XMPSession.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Initialize the object.

Inputs

base	the object whose XMPSemanticInterface this is.

Outputs

None

Exceptions Signalled

None

Pre conditions

None.

Post conditions

Object is instantiated.

GetCoercionHandler

Function Prototype

```
virtual void GetCoercionHandler(
        DescType fromType,
        DescType toType,
        XMPCoercionHandler* handler,
        XMPSLong*  handlerRefcon,
        XMPBoolean* fromTypeIsDesc);
```

Protocol

Semantic Events

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

See AppleEvent Manager documentation.

Inputs

fromType	The descriptor type of the data coerced by the handler.
toType	The descriptor type of the resulting data.

Outputs

handler	A pointer to the coercion handler.
handlerRefCon	A reference constant that is passed by the Apple Event Manager to the handler each time the handler is called.

fromTypeIsDesc	Specifies the form of the data to be coerced. If the value of this parameter is true, the coercion handler expects the data to be passed as a descriptor. If the value is false, the coercion handler expects a pointer to the data.

Exceptions Signalled

memFullErr	Not enough room in heap zone
errAEHandlerNotFound	No coercion handler found

Pre conditions

None.

Post conditions

None.

GetEventHandler

Function Prototype

```
virtual void GetEventHandler(
        AEEventClass theAEEventClass,
        AEEventID theAEEventID,
        XMPEventHandlerProcPtr* handler,
        XMPSLong* handlerRefcon);
```

Protocol

Semantic Events

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

See AppleEvent Manager documentation.

Inputs

theAEEventClass	The event class for the Apple event or events to be dispatched for this entry.
theAEEventID	The event ID for the Apple event or events to be dispatched for this entry.

Outputs

handler	A pointer to an Apple event handler for this dispatch table entry.
handlerRefCon	A reference constant that is passed by the Apple Event Manager to the handler each time the handler is called.

Exceptions Signalled

errAEHandlerNotFound	No handler found for an Apple event

Pre conditions

None.

Post conditions

None.

GetObjectAccessor

Function Prototype

```
virtual void GetObjectAccessor(
        DescType  desiredClass,
        DescType containerType,
        XMPAccessorProcPtr* theAccessor,
        XMPSLong* accessorRefcon);
```

Protocol

Content Model

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

See AppleEvent Manager documentation.

Inputs

desiredClass	The object class of the Apple event objects to be located by the object accessor function for this table entry.
containerType	The descriptor type of the token used to specify the container for the desired objects.

Outputs

theAccessor	A pointer to the object accessor function for this table entry.
accessorRefCon	A reference constant passed by the Apple Event Manager to the object accessor function whenever the function is called.

Exceptions Signalled

See	AppleEvent Mgr documentation.

Pre conditions

None.

Post conditions

None.

GetOSLSupportFlags

Function Prototype

```
virtual XMPShort GetOSLSupportFlags();
```

Protocol

Semantic Events

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Get the flags indicating amount of support for the OSL.

Inputs

None

Outputs

flags	The flags indicating amount of support for the OSL.

Exceptions Signalled

None

Pre conditions

None.

Post conditions

None.

GetSpecialHandler

Function Prototype

```
virtual void GetSpecialHandler(
        AEKeyword  functionClass,
        XMPSpecialHandlerPtr* handler);
```

Protocol

Semantic Events

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

See AppleEvent Manager documentation.

Inputs

functionClass	The keyword for the special handler that is installed.

Outputs

handler	A pointer to the special handler.

Exceptions Signalled

memFullErr	Not enough room in heap zone
errAENotASpecialFunction	Wrong keyword for a special function

Pre conditions

None.

Post conditions

None.

InitSemanticInterface

Function Prototype

```
void InitSemanticInterface(
        XMPObject* base);
```

Protocol

Semantic Events

Protection

Public. Only parts should instantiate these. The application shell can get its copy through the XMPSession.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Initialize the object.

Inputs

None

Outputs

None

Exceptions Signalled

kXMPErrOutOfMemory	Not enough room in heap zone.

Pre conditions

None.

Post conditions

fEventHandlerTable, fObjectAccessorTable, and fCoercionHandlerTable have been succesfully instantiated.

InstallAdjustMarksProc

Function Prototype

```
InstallAdjustMarksProc(
        XMPMarkPtr   adjustMarksProc,
        XMPSLong     refCon);
```

Protocol

Semantic Events

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Install a reference to a callback function that gets called to unmark objects that have previously been marked.

Inputs

countProc	the procedure pointer
refCon	a reference constant

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

The callback function and refCon are stored and available for future use by the Semantic Events subsystem.

InstallCoercionHandler

Function Prototype

```
virtual void InstallCoercionHandler(
        DescType fromType,
        DescType toType,
        XMPCoercionHandler handler,
        XMPSLong handlerRefcon,
        XMPBoolean fromTypeIsDesc);
```

Protocol

Semantic Events

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

See AppleEvent Manager documentation.

Inputs

fromType	The descriptor type of the data coerced by the handler.
toType	The descriptor type of the resulting data.
handler	A pointer to the coercion handler.
handlerRefcon	A reference constant that is passed by the Apple Event Manager to the handler each time the handler is called.
fromTypeIsDesc	Specifies the form of the data to be coerced. If the value of this parameter is true, the coercion handler expects the data to be passed as a descriptor. If the value is false, the coercion handler expects a pointer to the data.

Outputs

None.

Exceptions Signalled

memFullErr	Not enough room in heap zone

Pre conditions

None.

Post conditions

The function will be installed in the coercion handler table.

InstallCompareProc

Function Prototype

```
InstallCompareProc(
        XMPCompareProcPtr    compareProc,
        XMPSLong       refCon);
```

Protocol

Semantic Events

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Install a reference to a callback function that compares objects.

Inputs

compareProc	the procedure pointer
refCon	a reference constant

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

The callback function and refCon are stored and available for future use by the Semantic Events subsystem.

InstallCountProc

Function Prototype

```
InstallCountProc(
        XMPCountProcPtr    countProc,
        XMPSLong       refCon);
```

Protocol

Semantic Events

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Install a reference to a callback function that counts objects.

Inputs

countProc	the procedure pointer
refCon	a reference constant

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

The callback function and refCon are stored and available for future use by the Semantic Events subsystem.

InstallDisposeTokenProc

Function Prototype

```
InstallDisposeTokenProc(
        XMPDisposeTokenProcPtr    disposeTokenProc,
        XMPSLong       refCon);
```

Protocol

Semantic Events

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Install a reference to a callback function that disposes tokens.

Inputs

countProc	the procedure pointer
refCon	a reference constant

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

The callback function and refCon are stored and available for future use by the Semantic Events subsystem.

InstallEventHandler

Function Prototype

```
virtual void InstallEventHandler(
        AEEventClass  theAEEventClass,
        AEEventID theAEEventID,
        XMPEventHandlerProcPtr handler,
        XMPSLong handlerRefcon);
```

Protocol

Semantic Events

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

See AppleEvent Manager documentation.

Inputs

theAEEventClass	The event class for the Apple event or events to be dispatched for this entry.
theAEEventID	The event ID for the Apple event or events to be dispatched for this entry.
handler	A pointer to an Apple event handler for this dispatch table entry.

handlerRefCon — A reference constant that is passed by the Apple Event Manager to the handler each time the handler is called.

Outputs

None.

Exceptions Signalled

memFullErr — Not enough room in heap zone

Pre conditions

None.

Post conditions

The function will be installed in the event handler table.

InstallGetErrDescProc

Function Prototype

```
InstallGetErrDescProc(
        XMPMarkPtr    getErrDescProc,
        XMPSLong      refCon);
```

Protocol

Semantic Events

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Install a reference to a callback function that gets called to return a pointer to a client's descriptor record that will be used to write the descriptor for the last object being worked on when an error occurred.

Inputs

countProc — the procedure pointer

refCon — a reference constant

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

The callback function and refCon are stored and available for future use by the Semantic Events subsystem.

InstallGetMarkTokenProc

Function Prototype

```
InstallGetMarkTokenProc(
        XMPGetMarkTokenProcPtr    getMarkTokenProc,
        XMPSLong      refCon);
```

Protocol

Semantic Events

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Install a reference to a callback function that gets a token that the client will use to mark set of objects.

Inputs

countProc	the procedure pointer
refCon	a reference constant

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

The callback function and refCon are stored and available for future use by the Semantic Events subsystem.

InstallMarkProc

Function Prototype

```
InstallMarkProc(
        XMPMarkPtr    markProc,
        XMPSLong      refCon);
```

Protocol

Semantic Events

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Install a reference to a callback function that gets called repeatedly to mark a set of objects.

Inputs

countProc	the procedure pointer
refCon	a reference constant

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

The callback function and refCon are stored and available for future use by the Semantic Events subsystem.

InstallObjectAccessor

Function Prototype

```
virtual void InstallObjectAccessor(
        DescType desiredClass,
        DescType containerType,
        XMPAccessorProcPtr theAccessor,
        XMPSLong accessorRefcon);
```

Protocol

Content Model

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

See AppleEvent Manager documentation.

Inputs

desiredClass	The object class of the Apple event objects to be located by the object accessor function for this table entry.
containerType	The descriptor type of the token used to specify the container for the desired objects.
theAccessor	A pointer to the object accessor function for this table entry.
accessorRefCon	A reference constant passed by the Apple Event Manager to the object accessor function whenever the function is called.

Outputs

None

Exceptions Signalled

See	AppleEvent Manager documentation See AppleEvent Manager documentation
kXMPErrOutOfMemory	out of memory

Pre conditions

None.

Post conditions

The function will be installed in the object acessor table. An existing function will be replaced.

InstallSpecialHandler

Function Prototype

```
virtual void InstallSpecialHandler(
        AEKeyword  functionClass,
        XMPSpecialHandlerPtr handler);
```

Protocol

Semantic Events

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

See AppleEvent Manager documentation.

Inputs

functionClass	The keyword for the special handler that is installed.
handler	A pointer to the special handler.

Outputs

None.

Exceptions Signalled

paramErr	handler pointer is nil or odd
memFullErr	Not enough room in heap zone
errAENotASpecialFunction	Wrong keyword for a special function

Pre conditions

None.

Post conditions

The function will be installed in the special handler table.

RemoveCoercionHandler

Function Prototype

```
virtual void RemoveCoercionHandler(
        DescType fromType,
        DescType toType,
        XMPCoercionHandler handler);
```

Protocol

Semantic Events

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

See AppleEvent Manager documentation.

Inputs

fromType	The descriptor type of the data coerced by the handler.
toType	The descriptor type of the resulting data.
handler	A pointer to the coercion handler.

Outputs

None.

Exceptions Signalled

memFullErr	Not enough room in heap zone
errAEHandlerNotFound	No coercion handler found

Pre conditions

None.

Post conditions

The function is removed from the coercion handler table.

RemoveEventHandler

Function Prototype

```
virtual void RemoveEventHandler(
        AEEventClass  theAEEventClass,
        AEEventID theAEEventID,
        XMPEventHandlerProcPtr handler);
```

Protocol

Semantic Events

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

See AppleEvent Manager documentation.

Inputs

theAEEventClass	The event class for the Apple event or events to be dispatched for this entry.
theAEEventID	The event ID for the Apple event or events to be dispatched for this entry.
handler	A pointer to an Apple event handler for this dispatch table entry.

Outputs

None.

Exceptions Signalled

errAEHandlerNotFound	No handler found for an Apple event

Pre conditions

None.

Post conditions

The function is removed from the event handler table.

RemoveObjectAccessor

Function Prototype

```
virtual void RemoveObjectAccessor(
        DescType desiredClass,
        DescType containerType,
        XMPAccessorProcPtr theAccessor);
```

Protocol

Content Model

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

See AppleEvent Manager documentation.

Inputs

desiredClass	The object class of the Apple event objects located by the object accessor function.
containerType	The descriptor type of the token that identifies the container for the objects located by the object accessor function.
theAccessor	A pointer to the object accessor function you want to remove.

Outputs

None.

Exceptions Signalled

See	AppleEvent Mgr documentation.

Pre conditions

None.

Post conditions

The object accessor function is removed from the object accessor table.

RemoveSpecialHandler

Function Prototype

```
virtual void RemoveSpecialHandler(
        AEKeyword functionClass,
        XMPSpecialHandlerPtr handler);
```

Protocol

Semantic Events

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

See AppleEvent Manager documentation.

Inputs

functionClass	The keyword for the special handler that is installed.

handler — A pointer to the special handler.

Outputs

None.

Exceptions Signalled

memFullErr — Not enough room in heap zone

errAENotASpecialFunction — Wrong keyword for a special function

Pre conditions

None.

Post conditions

The function is removed from the special handler table.

SetOSLSupportFlags

Function Prototype

```
virtual void SetOSLSupportFlags(
        XMPShort flags);
```

Protocol

Semantic Events

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Set the flags indicating the support for the OSL that this XMPSemanticInterface provides. See the documentation for the OSL routine, AEResolve.

Inputs

flags — The OSL support flags. See the documentation for AEResolve.

Outputs

None

Exceptions Signalled

None

Pre conditions

None.

Post conditions

None.

~XMPMacSemanticInterface

Function Prototype

```
XMPVMethod ~XMPMacSemanticInterface();
```

Protocol

Semantic Events

Protection

Public. See notes for XMPMacSemanticInterface

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Destroy the object

Inputs

None

Outputs

None

Exceptions Signalled

None

Pre conditions

None.

Post conditions

Object is destroyed.

XMPSession

Basic Class Documentation

This class encapsulates access to OpenDoc globals as well as initialization and shutdown of the OpenDoc environment. It calls the constructor for a number of unique global objects and must have access to those classes' constructors. Platform vendor should implement. XMPSession is a subclass of XMPObject. Note that XMPSession is a subclass of XMPBaseSession. This class contains some of the public API available to developers. However, the existence of XMPBaseSession is an artifact of the C++ language. Please use this document as a guide to the OpenDoc session-level functionality. In particular, the method GetBinding is private to OpenDoc.

Theory of Operation

Initialization of OpenDoc is through the function OpenXMPSession and shutdown through the function XMPSession::Close. OpenXMPSession creates a number of global objects, saving references to them, and XMPSession::Close destroys these objects. Accessor functions return the value of the cached references.

Invariants Maintained by Class

All member variables always contain valid references.

Other Persistent Properties

Member Functions

Close

This method is only of interest to Container Application developers.

Function Prototype

```
void Close();
```

Protocol

System

Protection

Public. Should only be called once for each OpenXMPSession call. Usually called by the application shell.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Shuts down OpenDoc environment.

Inputs

None

Outputs

None

Exceptions Signalled

None

Pre conditions

None.

Post conditions

OpenDoc environment will be shut down.

GetArbitrator

Function Prototype

```
virtual XMPArbitrator* GetArbitrator();
```

Protocol

System

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Return reference to the global object.

Inputs

None

Outputs

Return	Reference to the global object

Exceptions Signalled

None

Pre conditions

None.

Post conditions

None.

GetClipboard

Function Prototype

virtual XMPClipboard* GetClipboard();

Protocol

System

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Return reference to the global object.

Inputs

None

Outputs

Return	Reference to the global object

Exceptions Signalled

None

Pre conditions

None.

Post conditions

None.

GetDispatcher

Function Prototype

virtual XMPDispatcher* GetDispatcher();

Protocol

System

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Return reference to the global object.

Inputs

None

Outputs

Return — Reference to the global object

Exceptions Signalled

None

Pre conditions

None.

Post conditions

None.

GetDragAndDrop

Function Prototype

```
virtual XMPDragAndDrop* GetDragAndDrop();
```

Protocol

System

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Return reference to the global object.

Inputs

None

Outputs

Return — Reference to the global object

Exceptions Signalled

None

Pre conditions

None.

Post conditions

None.

GetLinkManager

Function Prototype

```
XMPLinkManager* GetLinkManager();
```

Protocol

Data Interchange

Protection

Public. Should not be called by parts.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Return reference to the global object.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetMessageInterface

Function Prototype

virtual XMPMessageInterface* GetMessageInterface();

Protocol

System

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Return reference to the global object.

Inputs

None

Outputs

Return — Reference to the global object

Exceptions Signalled

None

Pre conditions

None.

Post conditions

None.

GetNameResolver

Function Prototype

virtual XMPNameResolver* GetNameResolver();

Protocol

System

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Return reference to the global object.

Inputs

None

Outputs

Return — Reference to the global object

Exceptions Signalled

None

Pre conditions

None.

Post conditions

None.

GetNameSpaceManager

Function Prototype

XMPNameSpaceManager* GetNameSpaceManager();

Protocol

System

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Return reference to the global object.

Inputs

None

Outputs

Return	Reference to the global object

Exceptions Signalled

None

Pre conditions

None.

Post conditions

None.

GetSemanticInterface

Function Prototype

```
virtual XMPSemanticInterface* GetSemanticInterface();
```

Protocol

System

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns cached reference to system's XMPSemanticInterface. This is not a unique object. Other OpenDoc object may own one or more instances of XMPSemanticInterface. This object can be used to install system level event handlers and object accessors.

Inputs

None

Outputs

Return	Reference to system's copy of the the XMPSemanticInterface.

Exceptions Signalled

None

Pre conditions

None.

Post conditions

None.

GetShellSemtInterface

This method is only of interest to Container Application developers.

Function Prototype

```
XMPSemanticInterface* GetShellSemtInterface();
```

Protocol

System

Protection

Public. Only the shell should use this method.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Return a reference to an XMPSemanticInterface object that the application shell can use.

Inputs

None.

Outputs

Return	a reference to the shell's XMPSemanticInterface object

Exceptions Signalled

None

Pre conditions

None.

Post conditions

None.

GetStorageSystem

Function Prototype

```
virtual XMPStorageSystem* GetStorageSystem();
```

Protocol

System

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Return reference to the global object.

Inputs

None

Outputs

Return	Reference to the global object

Exceptions Signalled

None

Pre conditions

None.

Post conditions

None.

GetTranslation

Function Prototype

```
virtual XMPTranslation* GetTranslation();
```

Protocol

System

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Return reference to the global object.

Inputs

None

Outputs

Return	Reference to the global object

Exceptions Signalled

None

Pre conditions

None.

Post conditions

None.

GetType

Function Prototype

```
XMPBoolean GetType(
        XMPTypeToken token,
        XMPType* type);
```

Protocol

Name Binding

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Get the XMPType corresponding to the given XMPTypeToken. If the token does not exist, kXMPFalse is retuned. kXMPTrue is returned otherwise.

Inputs

token	The XMPTypeToken of interest.

Outputs

Return	Whether the type for the given token exists in the token table.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetUndo

Function Prototype

```
virtual XMPUndo* GetUndo();
```

Protocol

System

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Return reference to the global object.

Inputs

None

Outputs

Return	Reference to the global object

Exceptions Signalled

None

Pre conditions

None.

Post conditions

None.

GetWindowState

Function Prototype

virtual XMPWindowState* GetWindowState();

Protocol

System

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Return reference to the global object.

Inputs

None

Outputs

Return	Reference to the global object

Exceptions Signalled

None

Pre conditions

None.

Post conditions

None.

OpenXMPSession

This method is only of interest to Container Application developers.

Function Prototype

XMPSession* OpenXMPSession ();

Protocol

System

Protection

Public. Usually called by the application shell.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Creates new XMPSession object; returns reference.

Inputs

None

Outputs

return	reference to the XMPSession object created by this call.

Exceptions Signalled

kXMPErrInitFailed	OpenDoc could not be initialized.

Pre conditions

This should only be called once per process.

Post conditions

The OpenDoc environment will be initialized.

Purge

Function Prototype

```
XMPSize Purge(
        XMPSize size);
```

Protocol
System, XMPObject
Protection
Public.
Override policy
Derived class **cannot** override.

Derived class **cannot** call base class behavior.
Basic operation
Make memory available.
Inputs

size	The size requested to be freed.

Outputs

Return	The amount of memory that was able to be freed.

Exceptions Signalled
None
Pre conditions
None.
Post conditions
Some memory may be freed for use.

RemoveEntry

Function Prototype

```
void RemoveEntry(
        XMPType type);
```

Protocol
Name Binding
Protection
Public.
Override policy
Derived class **cannot** override.

Derived class **cannot** call base class behavior.
Basic operation
Remove an XMPType from the type/token table. Any unique ID previously generated for this type can now be reused. No error is signalled if the type was not previously tokenized.
Inputs

type	The XMPType to be removed from the type/token table.

Outputs
None.
Exceptions Signalled
None.
Pre conditions
None.
Post conditions
The type/token pair corresponding to this type will be removed from the token table.

Tokenize

Function Prototype

```
XMPTypeToken Tokenize(
        XMPType type);
```

Protocol
Name Binding

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns a unique XMPTypeToken for a given XMPType. Original token is returned if type has been previously tokenized.

Inputs

type	The XMPType to register

Outputs

Return	The unique token for this type.

Exceptions Signalled

kXMPErrOutOfMemory	out of memory

Pre conditions

None.

Post conditions

A new unique token is generated if the type has not yet been tokenized.

UniqueChangeID

Function Prototype

```
XMPChangeID UniqueChangeID();
```

Protocol

Data Interchange

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Return a change identification unique to this session and unlikely to be repeated on the network.

Inputs

None.

Outputs

<result>	A unique change identification.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

Result is an identifier with a high likelyhood of uniqueness.

XMPShape

Basic Class Documentation

Base class: XMPObject.
Class XMPShape has no friends.
This class represents a geometric shape or area of a document. It is mostly used to define an XMPFrame's and XMPFacet's various shapes for frame negotiation, clipping, hit testing, etc. More background information can be found in the XMPFrame and XMPFacet class documentation.

Theory of Operation

An XMPShape starts out life uninitialized, with no shape data, and must be initialized before use. Public methods are defined to get and set the shape data in various formats, both universal (rectangle, polygon) and platform specific (QuickDraw Region, QuickDraw GX shape, …) The most universal representation of a shape is a polygon, and this is how it is stored in a document. However, some platform-specific shape types (like Regions) — which are needed for ephemeral shapes such as clipping regions that are closely tied to the native windowing system — may not be representable as polygons. A shape has a Geometry Mode which governs whether it is required to maintain a polygonal representation, or whether this representation can be abandoned for the sake of efficiency.

Invariants Maintained by Class

An XMPShape is an "envelope" that points to a private XMPRealShape object. Most of the calls to XMPShape are delegated to the XMPRealShape. There are various subclasses of the abstract XMPRealShape class; these represent rectangles, polygons, QuickDraw Regions, and QuickDraw GX shapes. During an XMPShape's life it may use various different XMPRealShapes to represent its data, depending on the particular shape. The client does not need to worry about this.

Other Persistent Properties

Member Functions

XMPShape

Function Prototype

XMPShape();

Protocol

Imaging

Protection

Public. No restrictions.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, before derived class behavior.

Basic operation

Creates an XMPShape object with no shape data. This must be followed by a call to SetRectangle, SetPlatformShape, CopyFrom or ReadShape (or ~XMPShape, of course.)

Inputs

None.

Outputs

<return> an XMPShape object.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

Shape is allocated but uninitialized (fShape==NULL.)

~XMPShape

Function Prototype

```
~XMPShape();
```

Protocol

Imaging

Protection

Public. No restrictions.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, after derived class behavior.

Basic operation

Deletes an XMPShape object and any private shape data.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

Shape, fShape (if any) and fShape's private data (if any) are deleted.

ContainsPoint

Function Prototype

```
XMPBoolean ContainsPoint(
        XMPPoint point);
```

Protocol

Imaging

Protection

Public. No restrictions.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Determines whether the shape contains a point.

Inputs

point	XMPPoint given in this shape's coordinate space.

Outputs

<return>	kXMPTrue if aPoint is in this shape, kXMPFalse otherwise.

Exceptions Signalled

None.

Pre conditions

Shape is initialized.

Post conditions

None.

Copy

Function Prototype

```
XMPShape* Copy();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns a copy of this shape. This is a deep copy: the new shape does not share any data with the original and both can be modified independently. This is safer than creating a new shape and calling CopyFrom on it, since you don't have to worry about deleting the new shape on the way out if the CopyFrom fails.

Inputs

None.

Outputs

<return>	A new shape identical to the receiver

Exceptions Signalled

kXMPErrOutOfMemory

Pre conditions

Shape is initialized.

Post conditions

None.

CopyFrom

Function Prototype

```
void CopyFrom(
        XMPShape* sourceShape);
```

Protocol

Imaging

Protection

Public. No restrictions.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Makes this shape an exact copy of sourceShape. This is a deep copy: the shape does not share any data with sourceShape and both can be modified independently.

Inputs

sourceShape	Pointer to another XMPShape object. Caller's storage responsability.

Outputs

None.

Exceptions Signalled

kXMPErrOutOfMemory	Not enough memory to copy shape data.

Pre conditions

sourceShape is initialized.

Post conditions

Shape is initialized and identical to sourceShape.

CopyPolygon

Function Prototype

```
XMPPolygon* CopyPolygon();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns a new XMPPolygon describing this shape. Caller is responsible for deleting this polygon afterward.
Not all shapes may have a polygonal representation so this call may fail with a kXMPErrNoGeometry exception. You can call HasGeometry first to check.
Some shapes (i.e. curves) may be only approximated by a polygon. Don't expect it to be 100% exact.

Inputs

None.

Outputs

<return>	Pointer to a new XMPPolygon.

Exceptions Signalled

kXMPErrOutOfMemory	
kXMPErrNoGeometry	Shape cannot be described as a polygon.

Pre conditions

Shape is initialized. Shape has geometry.

Post conditions

None.

GetBoundingBox

Function Prototype

```
void GetBoundingBox(
        XMPRect *bounds );
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the bounding box of this shape (the smallest rectangle that contains this shape.) After the call the XMPRect pointed to by bounds will be set to the bounding box.

Inputs

bounds	Pointer to the XMPRect into which to copy the bounding box.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

Shape is initialized.

Post conditions

*bounds is set to the shape's bounding box.

GetGeometryMode

Function Prototype

XMPGeometryMode GetGeometryMode();

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the current geometry mode of this shape. See the discussion of the fMode field.

Inputs

None.

Outputs

<return>	Shape's geometry mode.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetGXShape

This method is Macintosh specific.

Function Prototype

inline gxShape GetGXShape ();

Protocol

None.

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

This is a Mac-specific inline method that returns a QuickDraw GX shape equivalent to the shape. The caller should not modify this shape, but should dispose it when finished with it.

This call is identical to: (gxShape)this->GetPlatformShape(kXMPQuickDrawGX);

Inputs

None.

Outputs

<return>	Reference to a GX shape. Cannot be modified but should be disposed by caller.

Exceptions Signalled

None.

Pre conditions

QuickDraw GX is installed.

Shape is initialized.

Post conditions

None.

GetPlatformShape

Function Prototype

```
XMPPlatformShape GetPlatformShape(
        XMPGraphicsSystem
        );
```

Protocol

Imaging

Protection

Public. No restrictions.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the data of this shape object, for a particular graphics system. The format of the data is graphics system dependent. Whether or not this data is a copy depends on the implementation and the graphics system.
For kXMPQuickDraw, the data is a RgnHandle, which belongs to the shape and should not be modified or disposed by the caller.
For kXMPQuickDrawGX, it's a gxShape. The caller should not modify the shape, but should dispose it (i.e. lower its ref-count) when done.

Inputs

XMPGraphicsSystem	The graphics system of the data you want to get.

Outputs

<return>	Graphics-system-specific shape data (see method description)

Exceptions Signalled

kXMPErrInvalidGraphicsSystem	Graphics system unknown or not installed.

Pre conditions

Shape is initialized.

Post conditions

None.

GetQDRegion

This method is Macintosh specific.

Function Prototype

```
inline RgnHandle GetQDRegion ();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

This is a Mac-specific inline method that returns a Region equivalent to the shape. The region is owned by the shape and the caller must not modify or dispose it.
This call is identical to: (RgnHandle)this->GetPlatformShape(kXMPQuickDraw);

Inputs

None.

Outputs

<return>	QD Region representing the shape

Exceptions Signalled

None.

Pre conditions

Shape is initialized.

Post conditions

None.

HasGeometry

Function Prototype

```
XMPBoolean HasGeometry();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns true if the shape's geometric information (i.e. its polygonal shape) is accessible. A call to CopyPolygon is valid only if this is true. What makes a shape non-geometric is implementation specific. See SetGeometryMode.

Inputs

None.

Outputs

<result>	kXMPTrue if the shape can be represented as a polygon, else kXMPFalse.

Exceptions Signalled

None.

Pre conditions

Shape is initialized.

Post conditions

None.

Intersect

Function Prototype

```
void Intersect(
        XMPShape* sectShape);
```

Protocol

Imaging

Protection

Public. No restrictions.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Intersects this shape with sectShape. This shape becomes the intersection; fShape is unchanged.

Inputs

sectShape	another XMPShape object to intersect the receiver with.

Outputs

None.

Exceptions Signalled

kXMPErrOutOfMemory	not enough memory to intersect shapes.

kXMPErrNoGeometry	Geometry mode is kXMPNeedsGeometry, but diffShape has no geometry.

Pre conditions

Shape is initialized.
sectShape is initialized.

Post conditions

This shape becomes the intersection of sectShape and its previous shape.
sectShape is unchanged.

InverseTransform

Function Prototype

```
XMPShape* InverseTransform(
        XMPTransform* transform);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Runs this shape through the opposite of the given transform. This is the inverse operation of XMPShape::Transform. Shapes with no geometry may not be transformable except by simple offsets.

Inputs

transform	XMPTransform to apply the inverse of.

Outputs

<result>	The receiver.

Exceptions Signalled

kXMPErrNoShapeGeometry	Shape does not have enough geometric information to be transformed in this way.

Pre conditions

Shape is initialized.
transform is non-singular (i.e. invertible)

Post conditions

Shape is transformed.

IsEmpty

Function Prototype

```
XMPBoolean IsEmpty();
```

Protocol

Imaging

Protection

Public. No restrictions

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns true if the shape is empty (has no area).

Inputs

None.

Outputs

<result>	kXMPTrue if the shape is empty, otherwise kXMPFalse.

Exceptions Signalled

None.

Pre conditions

Shape is initialized.

Post conditions

None.

IsRectangular

Function Prototype

```
XMPBoolean IsRectangular();
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns kXMPTrue if the shape is a rectangle, kXMPFalse otherwise. Empty shapes are rectangular.

Inputs

None.

Outputs

<result>	kXMPTrue if the shape is rectangular, else kXMPFalse

Exceptions Signalled

None.

Pre conditions

Shape is initialized.

Post conditions

None.

IsSameAs

Function Prototype

```
XMPBoolean IsSameAs(
        XMPShape* compareShape);
```

Protocol

Imaging

Protection

Public. No restrictions.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns true if the two shapes are identical (describe the same area.) Warning: This is imperfectly implemented at alpha and may return false negatives for polygon shapes. It may also return false if the shapes are ever so slightly different due to rounding errors.

Inputs

compareShape	shape to compare against.

Outputs

<return>	kXMPTrue if shapes are equivalent or equal, kXMPFalse otherwise.

Exceptions Signalled

kXMPErrOutOfMemory	not enough memory to compare shapes.

Pre conditions

Shape is initialized.
compareShape is initialized.

Post conditions

None.

ReadShape

Function Prototype

```
XMPShape* ReadShape(
        XMPStorageUnit *su );
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Reads shape data from a storage unit into this shape. The storage unit must be pre-focused on a particular property. The kXMPPolygon value will be read if it exists, otherwise a platform-dependent shape value may be read if it exists, otherwise this object will be deleted and NULL will be returned.
The proper usage of this call is: s = s->ReadShape(su);

Inputs

su	Prefocused storage unit to read from.

Outputs

<return>	The receiver if the data was read successfully; otherwise NULL.

Exceptions Signalled

None.

Pre conditions

Storage unit is prefocused to a property.

Post conditions

Shape is initialized [if data was found in su] otherwise this shape is deleted.

SetGeometryMode

Function Prototype

```
void SetGeometryMode(
        XMPGeometryMode mode );
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Sets the geometry mode (fMode) of this shape. See the discussion of the fMode field. Setting the mode to kXMPNeedsGeometry will throw a kXMPErrNoGeometry exception if the shape has no geometry.

Inputs

mode	The geometry mode to use.

Outputs

None.

Exceptions Signalled

kXMPErrNoGeometry	If mode is set to kXMPNeedsGeometry but shape has no geometry.

Pre conditions

None.

Post conditions

None.

SetGXShape

This method is Macintosh specific.

Function Prototype

```
inline void SetGXShape (
        gxShape shape );
```

Protocol

None.

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

This is a Mac-specific inline method that sets the shape to a QuickDraw GX shape passed in by the caller. The gxShape is consumed by the shape and the caller must not use or dispose it afterward.
This call is identical to: this->SetPlatformShape(kXMPQuickDrawGX,(XMPPlatformShape)shape);

Inputs

shape	A reference to a QuickDraw GX shape of type polygon, path, empty or full.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

QuickDraw GX is installed.

Post conditions

Shape owns the gx shape passed in.

SetPlatformShape

Function Prototype

```
void SetPlatformShape(
        XMPGraphicsSystem graphicsSystem,
        XMPPlatformShape platformShape);
```

Protocol

Imaging

Protection

Public. No restrictions.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Sets the shape's graphics system dependent data. Whether or not the data is copied is also platform and graphics-system dependent. In the 1.0 Macintosh implementation, the data is not copied and the caller must not hang onto it after the call.

Inputs

graphicsSystem	On Mac, kXMPQuickDraw or kXMPQuickDrawGX.
platformShape	On Mac, this is a RgnHandle or gxShape.

Outputs

None.

Exceptions Signalled

kXMPErrInvalidGraphicsSystem	not a valid graphics system.

Pre conditions

graphicsSystem is a supported graphics system of the platform.
platformShape is a valid platform-specific shape for this graphics system.

Post conditions

Shape is initialized.

SetPolygon

Function Prototype

```
XMPShape* SetPolygon(
        XMPPolygon *polygon );
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Sets the shape based on the given polygon. The XMPPolygon is not modified or used by the shape; the caller is still responsible for it afterward. This call may be used to initialize a just-created XMPShape.

Inputs

polygon	XMPPolygon to use for this shape.

Outputs

<result>	The receiver.

Exceptions Signalled

None.

Pre conditions

polygon points to a valid XMPPolygon.

Post conditions

Shape is initialized to a copy of the given polygon.

SetQDRegion

This method is Macintosh specific.

Function Prototype

```
inline void SetQDRegion (
        RgnHandle rgn );
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

This is a Mac-specific inline method that sets the shape to a Region passed in by the caller. The region is consumed by the shape and the caller must not use or dispose it afterward.
This call is identical to: this->SetPlatformShape(kXMPQuickDraw,(XMPPlatformShape)rgn);

Inputs

rgn — The QuickDraw region to set.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

Shape is equivalent to rgn.
Shape either owns or has disposed the region passed in.

SetRectangle

Function Prototype

```
XMPShape* SetRectangle(
        XMPRect *rect );
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Changes this shape to be a rectangle with the given coordinates. This call may be used to initialize a shape right after it's created.

Inputs

rect — Rectangle to set this shape equal to.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

rect points to a valid XMPRect structure.

Post conditions

Shape is initialized to a rectangle identical to rect.

Subtract

Function Prototype

```
XMPShape* Subtract(
        XMPShape* diffShape);
```

Protocol

Imaging

Protection

Public. No restrictions.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Substracts diffShape from this shape; result is stored in this shape.

Inputs

diffShape	Shape to subtract from this shape.

Outputs

<result>	The receiver.

Exceptions Signalled

kXMPErrOutOfMemory	not enough memory to subtract shapes.
kXMPErrNoShapeGeometry	Geometry mode is kXMPNeedsGeometry, but diffShape has no geometry.

Pre conditions

Shape is initialized.
diffShape is initialized.

Post conditions

This shape is set to the value (this - diffShape).
diffShape is unaltered.

Transform

Function Prototype

```
XMPShape* Transform(
        XMPTransform* transform);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Runs this shape through a transformation (an offset, scale, rotation, skew, etc.)
Shapes with no geometry may not be transformable except by simple offsets.

Inputs

transform	XMPTransform to apply to this shape.

Outputs

<result>	The receiver.

Exceptions Signalled

kXMPErrNoGeometry	Shape doesn't have enough geometric information for this transformation.

Pre conditions

Shape is initialized.

Post conditions

Shape is transformed.

Union

Function Prototype

```
XMPShape* Union(
        XMPShape* unionShape);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Unions this shape with sectShape. This shape becomes the union; unionShape is unchanged.

Inputs

unionShape	Shape to combine with this shape.

Outputs

<result>	The receiver.

Exceptions Signalled

kXMPErrNoGeometry	Geometry mode is kXMPNeedsGeometry, but diffShape has no geometry.

Pre conditions

Shape is initialized.
unionShape is initialized.

Post conditions

This shape becomes the intersection of sectShape and its previous shape.
sectShape is unchanged.

WriteShape

Function Prototype

```
void WriteShape(
        XMPStorageUnit *su );
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Writes shape data to a specific value of a storage unit. The storage unit must already be focused to a particular property. If possible the data will be written as a polygon, type kXMPPolygon. If the shape is not geometric, the data may be written in some platform dependent format (or an exception may be thrown.)

Inputs

su	Storage unit (pre-focused) to write polygon data to.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

Shape is initialized.

Post conditions

Shape data is written to one or more values of a property of the storage unit.

XMPStorageSystem

Basic Class Documentation

When a OpenDoc session is launched, it creates a bunch of objects which are global for that session, one of these is an XMPStorageSystem object.When the OpenDoc session ends, this XMPStorageSystem object is deleted.
The class documented here (XMPAbsStorageSystem) is the abstract base class for any XMPStorageSystem. Platform Implementors should subclass XMPAbsStorageSystem to provide the functionality of the OpenDoc Storage System on their platforms.

Theory of Operation

There is only one XMPStorageSystem object per session.
The XMPStorageSystem object is purely an in-memory object, and has no persistent state.
Containers are created and retrieved by calling CreateContainer and GetContainer of the XMPStorageSystem object.
The XMPContainer objects in turn provide access to Documents, which provide access to Drafts which provide access to Storage Units.
The XMPStorageSystem object is responsible for guaranteeing that there is only one XMPContainer object associated with each physical Container. Each Container is responsible for guaranteeing that there is only one XMPDocument object associated with each Document within it. Each Document is responsible for guaranteeing that there is only one XMPDraft objectassociated with each Draft within it. Each Draft in turn is responsible for guaranteeing that there is only oneXMPStorageUnit objectassociated with each StorageUnit, Part, Frame, LinkSource and Link within it.

Invariants Maintained by Class

There is at most one XMPContainer object for every XMPContainerType, XMPContainerID tuple requested via GetContainer or CreateContainer.
Until a requested Container has been Released, the XMPContainer object will be valid. After a requested Container has been Released, the XMPContainer object which referred to that Container is not valid.
It is the responsibility of XMPStorageSystem to maintain these invariants, not XMPContainer, since XMPStorageSystem maintains the collection of XMPContainers.

Other Persistent Properties

Member Functions

GetSession

Function Prototype

```
XMPSession*  GetSession();
```

Protocol

ObjectStorage

Protection

Public. Parts can call this method even though they should really be calling XMPStorageUnit::GetSession().

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the current XMPSession object.

Inputs

none

Outputs

<return>	XMPBaseSession in which this XMPStorageSystem runs.

Exceptions Signalled

none

Pre conditions

None.

Post conditions

None.

NeedSpace

Function Prototype

```
void NeedSpace(
        XMPSize memSize,
        XMPBoolean doPurge);
```

Protocol

Memory

Protection

Public. This method can be called by any class. However, it should be used cautiously as this may be a slow operation.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

This is called to check for or preflight some memory, i.e. This should be called when an object anticipates the usage of a large memory block. It can also be used as a maintainence call for cleaning up the heap.
doPurge indicates whether or not this object should try to purge memory. This is done by calling Purge on its XMPContainers and transitively their XMPDocuments, XMPDrafts and persistent objects and XMPStorageUnits .
This call is not guaranteed to generate the memory requested.

Inputs

memSize	size of memory block desired
doPurge	whether or not to purge if necessary

Outputs

none

Exceptions Signalled

none

Pre conditions

None.

Post conditions

None.

XMPStorageUnit

Basic Class Documentation

A StorageUnit is the basic unit of persistent storage. It contains a list of properties, each of which has a unique name within the Storage Unit, and an ordered list of Values. XMPStorageUnit is used to manipulate a Storage Unit.
To keep clean and minimize the api, a set of common functions which can apply to the entire StorageUnit, a particular property, or a particular value have been abstracted out.
The class documented here is a abstract base class. Container Suite implementors should subclass this class to provide the functionality of a OpenDoc Storage Unit for their Container Suite. (For definition of Container Suite, please refer to documentation on XMPContainer).
Note: XMPStorageUnitView is almost identical to XMPStorageUnit, except that it has a fixed focus which is set upon creation of the XMPStorageUnitView object.

XMPStorageUnit is only instantiated by XMPDraft of the same Container Suite. When a Part or the Container App needs a Storage Unit, it will call XMPDraft::CreateStorage or XMPDraft::GetStorageUnit.

Theory of Operation

Properties and Values within a StorageUnit are not represented by objects. To indicate the context for calls which assume the context of the entire StorageUnit, a particular property, or a particular value, one uses the Focus method. Focusing can be absolute by passing a particular property name or value index, or relative by passing in positioncodes.
When focussed to a particular value, the StorageUnit provides a stream interface to that value.
This class is derived from XMPRefCntObject. A XMPStorageUnit object is instantiated when CreateStorageUnit or GetStorageUnit is called on its corresponding XMPDraft. When XMPStorageUnit is first constructed, its refCount is 1. Every time XMPDraft::GetStorageUnit is called on the same Storage Unit, the refCount of the XMPDraft object is incremented by 1. When XMPStorageUnit object is no longer needed, XMPStorageUnit::Release should be called.
The XMPDraft object is responsible for guaranteeing that there is only one XMPStorageUnit object
associated with each Storage Unit or persistent object within it.
A Property/Value combination is usually referred to as a context. When a context is applied to a XMPStorageUnit, this context is specifically called a focus. Many XMPStorageUnit operations require the establishment of a focus.
If a XMPStorageUnit is focused to a Value, the XMPStorageUnit also has a current offset. When the XMPStorageUnit is refocused (either to the same Value or to another Value), the current offset is reset to 0.
Storage Units can maintain persistent references to each other. These references are created and resolved using special XMPStorageUnit methods.

Invariants Maintained by Class

The focus of a StorageUnit is conserved over all methods except Focus, Remove and Add(Property/Value), which changes the focus to the (Property/Value) which was just added.

Other Persistent Properties

Member Functions

XMPStorageUnit

This method is only of interest to Container Application developers.

Function Prototype

XMPStorageUnit();

Protocol

Constructor

Protection

Public. Private within Container Suite. This method should only be called by XMPDraft::CreateStorageUnit or XMPDraft::GetStorageUnit.

Override policy

Derived class **must** override.

Derived class **must** call base class behavior, before derived class behavior.

Basic operation

This method is the constructor of the class.

Inputs

None.

Outputs

<return> this

Exceptions Signalled

none

Pre conditions

None.

Post conditions

The return result is an uninitialized XMPStorageUnit object.

~XMPStorageUnit

This method is only of interest to Container Application developers.

Function Prototype

~XMPStorageUnit();

Protocol

Cleanup

Protection

Public. Private within Container Suite. This method should only be called by XMPDraft.

Override policy

Derived class **must** override.

Derived class **must** call base class behavior, after derived class behavior.

Basic operation

This method is the destructor of the class. Note that it does not externalize the XMPStorageUnit object.

Inputs

none

Outputs

none

Exceptions Signalled

none

Pre conditions

None.

Post conditions

this is no longer a valid XMPStorageUnit object.

AddProperty

Function Prototype

```
XMPStorageUnit* AddProperty(
        XMPPropertyName propertyName);
```

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

If no property with the same name exist in this storage unit, it is added.
The focus is set to the given property.

Inputs

propertyName	Name of the property to add

Outputs

<return>	this

Exceptions Signalled

kXMPErrInvalidPropertyName	propertyName is kXMPNULL
kXMPErrCannotAddProperty	Failure to add the given Property.

Pre conditions

A Property with the given name may or may not exist in the Storage Unit.

Post conditions

XMPStorageUnit is focused to the Property of the given name.

AddValue

Function Prototype

```
XMPStorageUnit* AddValue(
        XMPValueType type);
```

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

If a Value with the given type does not exist in the property currently focused to, a Value with the given type is added.
The focus is set to the Value of the given type.

Inputs

type	type of value to add

Outputs

<return>	this

Exceptions Signalled

kXMPErrValueExists	Value exists alreday.
kXMPErrUnfocusedStorageUnit	This XMPStorageUnit is not focused to a Property.
kXMPErrInvalidType	type is kXMPNULL.
kXMPErrCannotAddType	Cannot add Type to Storage Unit.

Pre conditions

this XMPStorageUnit must be focused to a Property.
The given type may or may not exist in the focused Proeprty.

Post conditions

XMPStorageUnit is focused to the Value with the given type.

ClearAllPromises

Function Prototype

```
void    ClearAllPromises();
```

Protocol

Cleanup

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Removes all Values containing promises from this Storage Unit.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

CloneInto

Function Prototype

```
void   CloneInto(
           XMPDraftKey key,
           XMPStorageUnit* destStorageUnit,
           XMPStorageUnit* initiatingFrameSU);
```

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Deep-copy all Properties and Values of this Storage Unit to the specified destStorageUnit.

In order to avoid copying extra Storage Units during the deep-copy, an initiatingFrameSU may be specified. If initiatingFrameSU is not kXMPNULL, only those Storage Units reachable from initiatingFrameSU will be copied. If initiatingFrameSU is kXMPNULL, all the Storage Units reachable from this Storage Unit will be copied.

Note that the actual copying may not be completed until after EndClone is finished.

Inputs

key	XMPDraftKey identifying the Clone transaction.
initiatingFrameSU	All Storage Units cloned from this point on should be within the scope of this XMPFrame.
initiatingFrameSU	All Storage Units cloned from this point on should be within the scope of this XMPFrame.

Outputs

none

Exceptions Signalled

kXMPErrCloneNotStarted	A Clone was started unsuccessfully.
kXMPErrInvalidDraftKey	Invalid Clone.

Pre conditions

None.

Post conditions

None.

CloneTo

Function Prototype

```
XMPStorageUnit*   CloneTo(
        XMPDraftKey key,
        XMPDraft* destDraft,
        XMPStorageUnit* initiatingFrameSU);
```

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Deep-copy all Properties and Values of this Storage Unit to the newly created Storage Unit in the specified destDraft.
In order to avoid copying extra Storage Units during the deep-copy, an initiatingFrameSU may be specified. If initiatingFrameSU is not kXMPNULL, only those Storage Units reachable from initiatingFrameSU will be copied. If initiatingFrameSU is kXMPNULL, all the Storage Units reachable from this Storage Unit will be copied.
Note that the actual copying may not be completed until after EndClone is finished.

Inputs

initiatingFrameSU	All Storage Units cloned from this point on should be within the scope of this XMPFrame.
destDraft	Destination Draft to where this Stoage Unit is copied.
initiatingFrameSU	All Storage Units cloned from this point on should be within the scope of this XMPFrame.

Outputs

<return>	XMPStorageUnit referring to the new Storage Unit created in the destination draft.

Exceptions Signalled

kXMPErrCloneNotStarted	A Clone was started unsuccessfully.
kXMPErrInvalidDraftKey	Invalid Clone.

Pre conditions

None.

Post conditions

None.

CopyTo

Function Prototype

```
void   CopyTo(
        XMPStorageUnit* toSU);
```

Protocol
Storage Unit Manipulation
Protection
Public.
Override policy
Derived class **must** override.

Derived class **cannot** call base class behavior.
Basic operation
Shallow Copy (i.e., one-level) properties and values of this storageUnit to the given toSU.
Inputs
toSU the storage unit from which data is to be copied
Outputs
none
Exceptions Signalled
none
Pre conditions
None.
Post conditions
All properties and values in this XMPStorageUnit have their respective values in toSU.

CountProperties

Function Prototype
XMPULong CountProperties();
Protocol
Storage Unit Manipulation
Protection
Public.
Override policy
Derived class **must** override.

Derived class **cannot** call base class behavior.
Basic operation
Returns the number of Properties in this XMPStorageUnit.
Inputs
None
Outputs
<return> Number or Properties in this XMPStorageUnit.
Exceptions Signalled
None.
Pre conditions
None.
Post conditions
None.

CountValues

Function Prototype
XMPULong CountValues();
Protocol
Storage Unit Manipulation
Protection
Public.
Override policy
Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the number of Values in this focused XMPStorageUnit.

Inputs

None

Outputs

<return>	Number or Values in this focused XMPStorageUnit.

Exceptions Signalled

kXMPErrUnfocusedStorageUnit	This XMPStorageUnit is not focused to a Property.

Pre conditions

This XMPStorageUnit must be focused to a Property or a Value.

Post conditions

None.

CreateCursor

Function Prototype

```
XMPStorageUnitCursor* CreateCursor();
```

Protocol

Initialization

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Creates a XMPCursor object from the current focus of this XMPStorageUnit and returns it.

Inputs

none

Outputs

<return>	a newly created storageUnitCursor

Exceptions Signalled

kXMPErrUnfocusedStorageUnit	This XMPStorageUnit is not focused to a Property or a Value.

Pre conditions

None.

Post conditions

The returned XMPStorageUnitCursor object contains the same context as this XMPStorageUnit object.

CreateView

Function Prototype

```
XMPStorageUnitView* CreateView(
        XMPStorageUnitCursor* cursor);
```

Protocol

Initialization

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

If the cursor is kXMPNULL, this method uses this XMPStorageUnit and the focus of this XMPStorageUnit to create a new XMPStorageUnitView object. Otherwise, this method uses XMPStorageUnit and the specified cursor to create a new XMPStorageUnitView object.
The new XMPStorageUnitView object is returned as the function result.

Inputs

cursor	Cursor to create the current view. If cursor is kXMPNULL, the current focus is used.

Outputs

<return>	a StorageUnitView of this unit and the given curosr or the focus of this unit

Exceptions Signalled

none

Pre conditions

Either cursor is a valid XMPStorageUnitCursor or this XMPStorageUnit is focused to a Property or a Value.

Post conditions

The returned XMPStorageUnitView represents the appropriate context of this XMPStorageUnit.

DeleteValue

Function Prototype

```
void   DeleteValue(
           XMPULong length);
```

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Deletes length bytes at the current offset in the current value focus.
If the focused Value is a Promise Value, the Promise is resolved first before insertion is done.

Inputs

length	the length of bytes to delete from the focussed value

Outputs

none

Exceptions Signalled

kXMPErrUnfocusedStorageUnit	This XMPStorageUnit is not focused to a Value.

Pre conditions

this XMPStorageUnit object is focused to a particular value.

Post conditions

length bytes at the current offset in the current value focus have been deleted.

Exists

Function Prototype

```
XMPBoolean Exists(
        XMPPropertyName propertyName,
        XMPValueType valueType,
        XMPValueIndex valueIndex);
```

Protocol

Object Storage

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns kXMPTrue if a Property of the given Property Name exists and a Value with the specified valueType or valueIndex exists in the corresponding Property. Otherwise, returns kXMPFalse.
If valueType is kXMPNULL and valueIndex is 0, this method can be used for testing the existence of Property only.
valueIndex is ignored if valueType is not kXMPNULL.

Inputs

propertyName	Name of a Property.
valueType	Type of a Value.
valueIndex	Value Index.

Outputs

<return>	XMPBoolean showing whether the specified Property and Value exists.

Exceptions Signalled

kXMPErrInvalidName	Invalid Property Name.
kXMPErrInvalidType	Invalid Value Type.

Pre conditions

Property Name must be specified.

Post conditions

The focus of the XMPStorageUnit is not changed.

Exists

Function Prototype

```
XMPBoolean Exists(
        XMPStorageUnitCursor* cursor);
```

Protocol

Object Storage

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns kXMPTrue if the given context specified by the cursor exists. Otherwise, returns kXMPFalse.

Inputs

cursor	XMPStorageUnitCursor specifying a context (Property and/or a Value.)

Outputs

<return>	XMPBoolean showing whether the specified Property and Value exists.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

The focus of the XMPStorageUnit is not changed.

Externalize

Function Prototype

```
XMPStorageUnit* Externalize();
```

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Resolves all promises in this XMPStorageUnit and makes all changes in the Storage Unit persistent.

Inputs

none

Outputs

<return>	this

Exceptions Signalled

none

Pre conditions

None.

Post conditions

None.

Focus

Function Prototype

```
XMPStorageUnit* Focus(
        XMPPropertyName propertyName,
        XMPPositionCode propertyPosCode,
        XMPValueType valueType,
        XMPValueIndex valueIndex,
        XMPPositionCode valuePosCode);
```

Protocol

Storage Unit Manipulation

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

If the propertyName is specified (i.e., not kXMPNULL), it is used for focusing. Otherwise, the propertyPosCode is used to designate which property to focus on using the current focus.
If a valueType is specified (i.e., not kXMPNULL), it is used to focus the XMPStorageUnit. Both valueIndex and valuePosCode are ignored.
If valueType is not specified but valueIndex is specified (i.e., not 0), valueIndex is used to focus the XMPStorageUnit. valuePosCode is ignored in this case.
If neither valueType nor valueIndex is specified, the valuePosCode is used to designate which value to focus on using the current focus.

Inputs

propertyName	focus property name
propertyPosCode	focus relative property relation
valueType	focus value type

	valueIndex	focus value index
	valuePosCode	focus relative value relation
Outputs		
	<return>	this
Exceptions Signalled		
	kXMPErrInvalidProperty	No such Property.
	kXMPErrInvalidType	No Value with this Type.
	kXMPErrInvalidValueIndex	Invalid value index.
	kXMPErrInvalidPosCode	Invalid position code.

Pre conditions

None.

Post conditions

The StorageUnit is focused on the specified property and value.
The current offset is 0.

Focus

Function Prototype

```
XMPStorageUnit* Focus(
        XMPStorageUnitCursor* cursor);
```

Protocol

Storage Unit Manipulation

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Focus the Storage Unit using the property and value specified by the given XMPStorageUnitCursor object.

Inputs		
	cursor	the cursor to use for focusing
Outputs		
	<return>	this
Exceptions Signalled		
	kXMPErrInvalidPropertyName	No such Property.
	kXMPErrInvalidType	No Value with such Type.
	kXMPErrInvalidValueIndex	Invalid Value Index.
	kXMPErrInvalidPosCode	Invalid position code.

Pre conditions

None.

Post conditions

The StorageUnit is focused on the property and/or value specified by the XMPStorageUnitCursor object.
The current offset is 0.

GetDraft

Function Prototype

```
XMPDraft*  GetDraft();
```

Protocol

Getter

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the draft from which this XMPStorageUnit is instantiated.

Inputs

none

Outputs

<return>	XMPDraft from which this XMPStorageUnit object is created.

Exceptions Signalled

none

Pre conditions

None.

Post conditions

None.

GetGenerationNumber

Function Prototype

```
XMPULong   GetGenerationNumber();
```

Protocol

Storage Unit Manipulation

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Return the generation number of the current value focus.

Inputs

none

Outputs

<return>	the generation number of the current valuye

Exceptions Signalled

kXMPUnfocusedStorageUnit	This XMPStorageUnit object is not focused to a Value.

Pre conditions

this XMPStorageUnit object is focused to a value.

Post conditions

None.

GetID

Function Prototype

```
XMPID   GetID();
```

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the ID of this Storage Unit.

Inputs

none

Outputs

<return>	the id of the focus

Exceptions Signalled

none

Pre conditions

None.

Post conditions

None.

GetIDFromStorageUnitRef

Function Prototype

```
XMPStorageUnitID GetIDFromStorageUnitRef(
        XMPStorageUnitRef ref);
```

Protocol

Persistent Reference

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Resolves the given ref to a XMPStorageUnitID and returns the XMPStorageUnitID. If the specified ref cannot be resolved, an exception is raised.

Inputs

ref	XMPStorageUnitRef to be resolved.

Outputs

<return>	the StorageUnitID associated with the given rRef in this value

Exceptions Signalled

kXMPUnfocusedStorageUnit	this XMPStorageUnit object is not focused to a Value.
kXMPErrInvalidStorageUnitRef	Invalid Storage Unit Reference.

Pre conditions

this XMPStorageUnit object is focused to a value.

Post conditions

None.

GetName

Function Prototype

```
XMPName  GetName();
```

Protocol

Getter

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns a copy of the name of the Storage Unit. If the Storage Unit does not have a name, kXMPNULL is returned.

Inputs

none

Outputs

<return>	Name of the Storage Unit

Exceptions Signalled

none

Pre conditions

None.

Post conditions

None.

GetOffset

Function Prototype

```
XMPULong  GetOffset();
```

Protocol

Getter

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the current offset of the currently focused value.

Inputs

none

Outputs

<return>	the current offset of the currently focused value

Exceptions Signalled

kXMPErrUnfocusedStorageUnit	This XMPStorageUnit is not focused to a Value.

Pre conditions

this XMPStorageUnit object is focused to a value.

Post conditions

None.

GetPromiseValue

Function Prototype

```
XMPULong GetPromiseValue(
        XMPValueType valueType,
        XMPULong offset,
        XMPULong length,
        XMPValue value,
        XMPPart **sourcePart);
```

Protocol

Promise Manipulation

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Extracts data placed in the value when the promise was created. This does not fulfill the promise.

Inputs

valueType	Value Type of Value to be focused to.
offset	Offset from which the data should be retrieved.
length	Length of data to be retrieved.
value	Buffer into which the Promise data should be retrieved.
sourcePart	Pointer to the buffer where the sourcePart can be returned.

Outputs

<return>	the actual number of bytes read
sourcePart	Source Part which has put out the Promise.

Exceptions Signalled

kXMPErrUnfocusedStorageUnit	This XMPStorageUnit object is not focused to a Property.
kXMPErrInvalidType	valueType is kXMPNULL.
kXMPErrCannotAddType	Invalid valueType.
kXMPErrInvalidValue	Cannot focus to specified Value with valueType.
kXMPErrNotPromise	The Value does not contain a Promise.

Pre conditions

The current storage unit must be focused to a valid property containing a promise of the same type specified in the valueType parameter.

Post conditions

The promise data (not the data the promise represents) is copied into the supplied value parameter.

GetProperty

Function Prototype

```
XMPPropertyName    GetProperty();
```

Protocol

Getter

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the name of the currently focused property.

Inputs

None.

Outputs

<return>	Property name of the current focus.

Exceptions Signalled

kXMPErrUnfocusedStorageUnit	this XMPStorageUnit is not focused to a Property.

Pre conditions

this XMPStorageUnit object is focused to a property or value.

Post conditions

None.

GetSession

Function Prototype

```
XMPSession* GetSession();
```

Protocol

Getter

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the session in which this XMPStorageUnit object is running.
This is a utility routine Parts should use to retrieve the current XMPSession object.

Inputs

None.

Outputs

<return>	XMPSession object in which this XMPStorageUnit object is running.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetSize

Function Prototype

```
XMPULong  GetSize();
```

Protocol

Value Manipulation

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the size of the current focus. If the current focus is the whole Storage Unit, the aggregate size of all the Properties and their Values is returned. If the current focus is a Property, the aggregate size of all the Values in the Property is returned. If the current focus is a Value, only the size of the Value is returned. If the current focus is a Promise Value, the Promise is resolved first before the Value size is evaluated.

Inputs

none

Outputs

<return>	Desired size.

Exceptions Signalled

kXMPUnfocusedStorageUnit

Pre conditions

None.

Post conditions

None.

GetStorageUnitRefIterator

Function Prototype

```
XMPStorageUnitRefIterator* GetStorageUnitRefIterator();
```

Protocol

Iteration

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns an iterator for iterating over all the persistent references created in the currently focused Value.

Inputs

None.

Outputs

<return>	Iterator used to iterate over all persistent references in the focused Value.

Exceptions Signalled

None.

Pre conditions

this XMPStorageUnit is focused to a Value.

Post conditions

None.

GetStrongStorageUnitRef

Function Prototype

```
XMPStorageUnitRef GetStrongStorageUnitRef(
        XMPStorageUnit* embeddedSU);
```

Protocol

Persistent Reference

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns a persistent reference to the Storage Unit refered to by embeddedSU. This persistent reference may be stored within the current value focus of this XMPStorageUnit and later used to retrieve the given Storage Unit. The returned XMPStorageUnitRef is strong in the sense that the Storage Unit it refers to will be copied in the CloneTo operation.

Inputs

embeddedSU	XMPStorageUnit object referring to the Storage Unit whose ref is desired.

Outputs

<return>	persistent reference to the Storage Unit referred to by embeddedSU.

Exceptions Signalled

kXMPUnfocusedStorageUnit

Pre conditions

this XMPStorageUnit object is focused to a value.
Storage Units referred to by this XMPStorageUnit and embeddedSU must be in the same Draft.

Post conditions

None.

GetType

Function Prototype

```
XMPValueType GetType();
```

Protocol

Getter

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the type of the currently focused value.

Inputs

none

Outputs

<return> Type of the focused particular value

Exceptions Signalled

kXMPErrInvalidValue
kXMPErrUnfocusedStorageUnit.

Pre conditions

this XMPStorageUnit object is focused to a value.

Post conditions

None.

GetValue

Function Prototype

```
XMPULong   GetValue(
        XMPULong length,
        XMPValue value);
```

Protocol

Value Manipulation

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Reads length bytes from the currently focused value at the current offset into the buffer value.
If offset+length is bigger than the size of the Value, only the bytes from the offset (inclusive) to the end of the Value will be read rendering the actual number of bytes read smaller than length.

Inputs

length Number of bytes to read into value
value a buffer of length>= length

Outputs

<return> Number of bytes actually read

Exceptions Signalled

none

Pre conditions

this XMPStorageUnit object is focused to a value.
value is a buffer of length length or more.

Post conditions

value contains length bytes from offset in the currently focused value.

GetWeakStorageUnitRef

Function Prototype

```
XMPStorageUnitRef GetWeakStorageUnitRef(
        XMPStorageUnit* embeddedSU);
```

Protocol

Persistent Reference

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns a persistent reference to the Storage Unit refered to by embeddedSU. This persistent reference may be stored within the current value focus and later used to retrieve the given Storage Unit. This XMPStorageUnitRef is weak in the sense that the Storage Unit it refers to will only be copied over in the CloneTo operation only if there is another Strong Storage Unit Ref referring to it.

Inputs

embeddedSU	Storage Unit for which a ref is desired

Outputs

<return>	a ref to the given storageUnit

Exceptions Signalled

kXMPUnfocusedStorageUnit	this XMPStorageUnit is not focused to a Value.

Pre conditions

this XMPStorageUnit object is focused to a value.

Post conditions

None.

IncrementGenerationNumber

Function Prototype

```
XMPULong   IncrementGenerationNumber();
```

Protocol

Generation

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Increment and return the generation number of the current value focus.

Inputs

none

Outputs

<return>	Generation number of the focused Value.

Exceptions Signalled

kXMPErrUnfocusedStorageUnit	this XMPStorageUnit is not focused to a Value.

Pre conditions

this XMPStorageUnit object is focused to a value.

Post conditions

None.

IncrementRefCount

Function Prototype

```
void IncrementRefCount();
```

Protocol

Reference counting

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Increments the reference count of this object.

Inputs

none

Outputs

none

Exceptions Signalled

none

Pre conditions

None.

Post conditions

The ref count of this XMPStorageUnit object is incremented by 1.

InitStorageUnit

This method is only of interest to Container Application developers.

Function Prototype

```
void InitStorageUnit(
        XMPDraft* draft,
        XMPStorageUnitID suid);
```

Protocol

Constructor

Protection

Public. Private within Storage System. This method should only be called by XMPDraft::CreateStorageUnit.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Initializes this XMPStorageUnit object.

Inputs

draft	the draft of this storage unit
suid	the id of this storage unit

Outputs

none

Exceptions Signalled

none

Pre conditions

this is a valid XMPStorageUnit.

Post conditions

this is an initialized XMPStorageUnit object.
this XMPStorageUnit is focused to all Properties.

InsertValue

Function Prototype

```
void   InsertValue(
           XMPULong length,
           XMPValue value);
```

Protocol

Storage Unit Manipulation

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Insert length bytes into the currently focused value at the current offset from the buffer value. If the focused Value is a Promise Value, the Promise is resolved first before insertion is done.

Inputs

length	Number of bytes to write into the focussed value

value Buffer of size>= length

Outputs

none

Exceptions Signalled

kXMPErrUnfocusedStorageUnit this XMPStorageUnit is not focused to a Value.

Pre conditions

this XMPStorageUnit object is focused to a value.
value is a buffer of size length or more.

Post conditions

length bytes of value have been inserted at the current offset in the currently focused value.

Internalize

This method is only of interest to Container Application developers.

Function Prototype

XMPStorageUnit* Internalize();

Protocol

ObjectStorage

Protection

Public. Private within Storage System. This method should only be called by XMPDraft.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Brings into ephemeral storage from persistent storage, all properties and their values in the current focus.

Inputs

none

Outputs

<return> this

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

All properties and values in the current focus have been brought into ephemeral storage from persistent storage.

IsPromiseValue

Function Prototype

XMPBoolean IsPromiseValue();

Protocol

Promise

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns kXMPTrue if focused Value contains a promise. Otherwise, returns kXMPFalse.

Inputs

None.

Outputs

<return>	kXMPTrue if the focused value contains a promise. otherwise false.

Exceptions Signalled

None.

Pre conditions

this XMPStorageUnit is focused to a Value.

Post conditions

None.

IsStrongStorageUnitRef

Function Prototype

```
XMPBoolean IsStrongStorageUnitRef(
        XMPStorageUnitRef ref);
```

Protocol

Persistent Reference

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns kXMPTrue if ref is a strong Storage Unit Ref. Otherwise, it returns kXMPFalse.

Inputs

ref	XMPStorageUnitRef to be tested.

Outputs

<return>	Boolean showing whether the ref is a Strong Storage Unit Ref.

Exceptions Signalled

None.

Pre conditions

this XMPStorageUnit is focused to a Value.

Post conditions

None.

IsWeakStorageUnitRef

Function Prototype

```
XMPBoolean IsWeakStorageUnitRef(
        XMPStorageUnitRef ref);
```

Protocol

Persistent Reference

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns kXMPTrue if ref is a weak Storage Unit Ref. Otherwise, it returns kXMPFalse.

Inputs

ref	XMPStorageUnitRef to be tested.

Outputs

<return>	Boolean showing whether the ref is a Strong Storage Unit Ref.

Exceptions Signalled

None.

Pre conditions

this XMPStorageUnit is focused to a Value.

Post conditions

None.

Lock

Function Prototype

```
XMPStorageUnitKey    Lock(
                        XMPStorageUnitKey key);
```

Protocol

Shared access

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Sets the status of this XMPStorageUnit to locked. Note that this is used for implementing a thread-safety mechanism. In order for the mechanism to work, every thread will have to acquire the lock before any operation on this XMPStorageUnit.

Inputs

key	Previous XMPStorageUnitKey acquired.

Outputs

<return>	XMPStorageUnitKey

Exceptions Signalled

kXMPErrStorageUnitLocked	XMPStorageUnit is already locked.
kXMPErrInvalidStorageUnitKey	Invalid XMPStorageUnitKey.

Pre conditions

None.

Post conditions

Either an exception is raised or the lock is granted.

Purge

This method is only of interest to Container Application developers.

Function Prototype

```
XMPSize   Purge(
             XMPSize size);
```

Protocol

Memory Management

Protection

Public. Private within Storage System. This method should only be called by XMPDraft::Purge.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Purge memory from ephemeral store until size bytes (not necessarily contiguous) have been freed up.

Inputs

size	Number of bytes to purge

Outputs

<return>	Number of bytes actually purged

Exceptions Signalled
none

Pre conditions
None.

Post conditions
Either size bytes (not necessarily contiguous) are free in the default heap or all Properties and Values in memory have been flushed out.

Release

Function Prototype
XMPDraft* Release();

Protocol
Reference counting

Protection
Public.

Override policy
Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation
Releases this XMPStorageUnit object. This is called when the reference to this XMPStorageUnit is no longer needed.

Inputs
none

Outputs
<return> fDraft

Exceptions Signalled
none

Pre conditions
None.

Post conditions
this is no longer a valid XMPStorageUnit object.

Remove

Function Prototype
XMPStorageUnit* Remove();

Protocol
Storage Unit Manipulation; Value Manipulation

Protection
Public.

Override policy
Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation
Removes a Value if focused to a Value. Otherwise, removes a property if focused to a property. Otherwise, removes all the properties.

Inputs
none

Outputs
<return> this

Exceptions Signalled
None.

Pre conditions
None.

Post conditions
The current focus is undefined.

RemoveStorageUnitRef

Function Prototype

```
XMPStorageUnit*    RemoveStorageUnitRef(
                       XMPStorageUnitRef ref);
```

Protocol

Persistent Reference

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Removes a XMPStorageUnitRef from a Value.

Inputs

None.

Outputs

None.

Exceptions Signalled

kXMPErrUnfocusedStorageUnit this XMPStorageUnit is not focused to a Value.

Pre conditions

this XMPStorageUnit object is focused to a value.
ref is a valid XMPStorageUnitRef (either strong or weak).

Post conditions

ref is no longer a valid XMPStorageUnitRef in that Value.

SetName

Function Prototype

```
void SetName(
         XMPName name);
```

Protocol

Setter

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Set the name of the Storage Unit.

Inputs

name the name of the focus

Outputs

none

Exceptions Signalled

none

Pre conditions

None.

Post conditions

the name of the Storage Unit is name.

SetOffset

Function Prototype

```
void   SetOffset(
           XMPULong offset);
```

Protocol

Setter

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Sets the current offset of the currently focused value to offset.

Inputs

offset	the offset to use witht eh currently focused value

Outputs

none

Exceptions Signalled

kXMPErrUnfocusedStorageUnit	this XMPStorageUnit is not focused to a Value.

Pre conditions

this XMPStorageUnit is focused to a Value.
0≤offset<size of the focused Value.

Post conditions

The current offset for value operations on that value is offset.

SetPromiseValue

Function Prototype

```
void  SetPromiseValue(
        XMPValueType valueType,
        XMPULong offset,
        XMPULong length,
        XMPValue value
        XMPPart *sourcePart);
```

Protocol

Promise

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Puts the given data into the focused value and make the focused value a promise.

Inputs

valueType	Type of Value the Promise is going to be created in.
offset	Offset when the Promise data should be stored.
length	Size of Promise data to be stored.
value	Buffer containing Promise data.
sourcePart	The Part which puts out the Promise.

Outputs

none

Exceptions Signalled

kXMPErrUnfocusedStorageUnit	this XMPStorageUnit is not focused to a Property.
kXMPErrInvalidType	valueType is invalid (kXMPNULL).
kXMPErrCannotAddType	Invalid Type.

Pre conditions

value must be of length bytes long or less.
this XMPStorageUnit is focused to a Property.

Post conditions

length bytes of value have been written into offset in the currently focused value. Also, the StorageUnit is updated to reflect that the value is a promise and the needed information is recorded.

SetType

Function Prototype

```
void   SetType(
           XMPValueType valueType);
```

Protocol

Setter

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Sets the type of the currently focused value.

Inputs

valueType	New type of the focused particular value

Outputs

none

Exceptions Signalled

kXMPErrUnfocusedStorageUnit	this XMPStorageUnit is not focused to a Value.
kXMPErrInvalidType	valueType is invalid (i.e., kXMPNULL).

Pre conditions

this XMPStorageUnit object is focused to a Value.

Post conditions

The type of the focused value is valueType.

SetValue

Function Prototype

```
void   SetValue(
           XMPULong length,
           XMPValue value);
```

Protocol

Value Manipulation

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Writes length bytes to the currently focused value at the current offset from the buffer value.

If length + offset is greater than the current size of the Value, the Value should be grown to accomodate the extra bytes.

Inputs

length	Number of bytes to write into the focussed value
value	Buffer of size>= length

Outputs

none

Exceptions Signalled

kXMPErrUnfocusedStorageUnit this XMPStorageUnit is not focused to a Value.

Pre conditions

this XMPStorageUnit object is focused to a Value.
value is a buffer of size length or more.

Post conditions

length bytes of value have been written into offset in the currently focused value

XMPStorageUnitCursor

Basic Class Documentation

XMPStorageUnitCursor has no base class.
Part developers or Platform developers can subclass this class.

Theory of Operation

A StorageUnitCursor contains a focus context for a storage unit. It can be used to make thread safe calls to all storage unit methods which would otherwise use the current focus of the storage unit.

Invariants Maintained by Class

The focus context (i.e., Property Name and Value Type OR Property Name and Value Index) cannot be changed once the object is initialized.

Other Persistent Properties

Member Functions

XMPStorageUnitCursor

Function Prototype

XMPStorageUnitCursor();

Protocol

Constructor

Protection

Public.

Override policy

Derived class **can** override.

Derived class **must** call base class behavior, before derived class behavior.

Basic operation

This is the constructor of the class.

Inputs

None.

Outputs

None

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

~XMPStorageUnitCursor

Function Prototype

~XMPStorageUnitCursor();

Protocol

Destructor

Protection

Public.

Override policy

Derived class **can** override.

Derived class **must** call base class behavior, after derived class behavior.

Basic operation

This is the destructor for the class.

Inputs

None

Outputs

None

Exceptions Signalled

None

Pre conditions

None.

Post conditions

None.

GetCursor

Function Prototype

```
void GetCursor(
        XMPPropertyName* propertyName,
        XMPValueType* valueType,
        XMPValueIndex* valueIndex);
```

Protocol

Getter

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the focus context of this cursor.

Inputs

propertyName	Property Name of the focus context.
valueType	Value Type for the focus context.
valueIndex	Pointer to buffer for Value index for the focus context.

Outputs

propertyName	Property Name for the focus context.
valueType	ValueType for the focus context.
valueIndex	Value index for the focus context.

Exceptions Signalled

valueIndex	Value index for the focus context.
valueIndex	Value index for the focus context.

Pre conditions

None.

Post conditions

None.

InitStorageUnitCursor

Function Prototype

```
void InitStorageUnitCursor(
        XMPPropertyName propertyName,
        XMPValueType valueType,
        XMPValueIndex valueIndex);
```

Protocol

Constructor

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, before derived class behavior.

Basic operation

Initializes this object with the focus context. propertyName must be supplied.

If valueType is kXMPNULL and valueIndex is 0, the focus context is to the specified Property.
If valueType is not kXMPNULL, the focus context is the value with the specified type.
If valueType is kXMPNULL and valueIndex is not 0, the focus context is the indexed value.

Inputs

propertyName	Property Name for the focus context.
valueType	Value Type for the focus context.
valueIndex	Value index for the focus context.

Outputs

None

Exceptions Signalled

kXMPErrInvalidProperty	Property Name is kXMPNULL.

Pre conditions

None.

Post conditions

this object is a fully functional XMPStorageUnitCursor (i.e., it can be used for XMPStorageUnit::Focus).

XMPStorageUnitRefIterator

Basic Class Documentation

XMPStorageUnitRefIterator is a companion of XMPStorageUnit , and is used to iterate over the references in a Value.
XMPStorageUnitRefIterator has no base class.
The class documented here (XMPStorageUnitRefIterator) is an abstract base class. Container Suite Implementors should subclass XMPStorageUnitRefIterator to provide the functionality of a OpenDoc StorageUnit Reference Iterator for their Container Suite.

Parts can instantiate this object when they need to iterate over all the persistent references of a Value.

Theory of Operation

Given the state of the focused XMPStorageUnit object, XMPStorageUnitRefIterator allows the developer to iterate over all the references in a Value, using the iterator's First(), Next() and IsNotComplete() methods.

Invariants Maintained by Class

XMPStorageUnitRef cannot be created or deleted during iteration.
XMPStorageUnit must remain focused to the Value on which XMPStorageUnitRefIterator is iterating.
The client of this class must check to see whether the reference returned by First or Next is valid or not by calling IsNotComplete.

Other Persistent Properties

Member Functions

XMPStorageUnitRefIterator

Function Prototype

XMPStorageUnitRefIterator();

Protocol

Constructor

Protection

Public.

Override policy

Derived class **must** override.

Derived class **must** call base class behavior, before derived class behavior.

Basic operation

Constructs the iterator.

Inputs

None.

Outputs

None

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

A properly constructed iterator.

~XMPStorageUnitRefIterator

Function Prototype

~XMPStorageUnitRefIterator();

Protocol

Destructor

Protection

Public.

Override policy

Derived class **must** override.

Derived class **must** call base class behavior, after derived class behavior.

Basic operation

This is the destructor of the class.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

this is a valid XMPStorageUnitRefIterator object.

Post conditions

this is no longer a valid XMPStorageUnitRefIterator object.

First

Function Prototype

```
XMPStorageUnitRef* First();
```

Protocol

Iteration

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the first reference in the Value if such a reference exists. If no such reference exists, the return value is undefined.

Inputs

None

Outputs

<return>	The first XMPStorageUnitRef in the focused Value of the XMPStorageUnit.

Exceptions Signalled

None

Pre conditions

None.

Post conditions

None.

InitStorageUnitRefIterator

Function Prototype

```
void InitStorageUnitRefIterator(
        XMPStorageUnit* storageUnit);
```

Protocol

Constructor

Protection

Public.

Override policy

Derived class **must** override.

Derived class **must** call base class behavior, before derived class behavior.

Basic operation

Initializes this XMPStorageUnitRefIterator object.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

IsNotComplete

Function Prototype

```
XMPBoolean IsNotComplete();
```

Protocol

Iteration

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns kXMPTrue if the iteration is not complete, kXMPFalse otherwise.

Inputs

None

Outputs

<return>	Boolean showing whether the iteration is incomplete.

Exceptions Signalled

None

Pre conditions

A constructed object of this class.

Post conditions

Result contains kXMPTrue if there are more references, kXMPFalse otherwise.

Next

Function Prototype

```
XMPStorageUnitRef* Next();
```

Protocol

Iteration

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the next reference in the Value if such a reference exists. If no such reference exists, the return value is undefined.

Inputs

None

Outputs

<return>	The next reference in the Value.

Exceptions Signalled

None

Pre conditions

First() has been called.

Post conditions

Result contains the next reference.

XMPStorageUnitView

Basic Class Documentation

This class is derived from XMPObject.
This class is implemented by Platform Implementors. It should be able to work with XMPStorageUnit and XMPStorageUnitCursor of any Container Suite.
Most methods of this class is not intended to be overriden.

Theory of Operation

A XMPStorageUnitView provides an easy and thread-safe way to access a specific Property or a Value of a Storage Unit. Many of the XMPStorageUnit methods are duplicated in XMPStorageUnitView. A XMPStorageUnitView method in general obtains the semaphore to ensure thread-safety, focuses the associated XMPStorageUnit and calls the corresponding XMPStorageUnit method and releases the semaphore. Due to this relations between XMPStorageUnitView and XMPStorageUnit, XMPStorageUnitView methods behave exactly the same as their corresponding XMPStorageUnit methods. That also means they return the same exceptions. (For this reason, the documentation for exceptions is omitted in this class. Please refer to documentation for XMPStorageUnit to get the full list of exceptions.)
XMPStorageUnitView objects can be constructed directly (i.e., using the constructor and Init method), or they can be obtained by a request to a XMPStorageUnit object. They may be deleted direcly by clients.

Parts can instantiate XMPStorageUnitView.

Invariants Maintained by Class

The XMPStorageUnit and XMPStorageUnitCursor of a StorageUnitView are specified in the InitStorageUnitView method, and these values never change.

Other Persistent Properties

Member Functions

XMPStorageUnitView

Function Prototype

XMPStorageUnitView();

Protocol

Initialization

Protection

Public.

Override policy

Derived class **can** override.

Derived class **must** call base class behavior, before derived class behavior.

Basic operation

This function is the constructor of the XMPStorageUnitView object.

Inputs

None.

Outputs

<return> this

Exceptions Signalled

none

Pre conditions

None.

Post conditions

this is an uninitialized XMPStorageUnitView object.

~XMPStorageUnitView

Function Prototype

~XMPStorageUnitView();

Protocol

Cleanup

Protection

Public.

Override policy

Derived class **can** override.

Derived class **must** call base class behavior, after derived class behavior.

Basic operation

This is the destructor of the XMPStorageUnitView object.
This method releases the XMPStorageUnit with which this XMPStorageUnitView object is associated and deletes the cursor which provides the context.
Note that this method does not externalize the XMPStorageUnit.

Inputs

none

Outputs

none

Exceptions Signalled

none

Pre conditions

None.

Post conditions

this is no longer a valid XMPStorageUnitView object.

AddProperty

Function Prototype

XMPStorageUnitView* AddProperty(
XMPPropertyName propertyName);

Protocol

Storage Unit Manipulation

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Adds a Property to the Storage Unit this XMPStorageUnitView is viewing. The resulting XMPStorageUnit will have the new Property as the focus even though the XMPStorageUnitView will maintain the cursor.

Inputs

XMPPropertyName

Outputs

none

Exceptions Signalled

none

Pre conditions

Its XMPStorageUnitCursor corresponds to a particular Property of the XMPStorageUnit.

Post conditions

The viewed XMPStorageUnit has a focus on the new Property.

AddValue

Function Prototype

```
XMPStorageUnitView* AddValue(
        XMPValueType type);
```

Protocol

Storage Unit Manipulation

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Adds a Value to the XMPStorageUnit it is viewing. The resulting XMPStorageUnit will have a new Value with the focus on it. This XMPStorageUnitView will maintain its cursor.

Inputs

none

Outputs

none

Exceptions Signalled

none

Pre conditions

Its XMPStorageUnitCursor corresponds to a particular value of the XMPStorageUnit.

Post conditions

The viewed XMPStorageUnit has a veiw on the added Value.

CloneInto

Function Prototype

```
void CloneInto(
        XMPDraftKey key,
        XMPStorageUnit* destSU,
        XMPStorageUnit* initiatingFrameSU);
```

Protocol

Storage Unit Management

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Deep-copy all Properties and Values of the Storage Unit this XMPStorageUnitView is viewing to the specified destStorageUnit.
In order to avoid copying extra Storage Units during the deep-copy, an initiatingFrameSU may be specified. If initiatingFrameSU is not kXMPNULL, only those Storage Units reachable from initiatingFrameSU will be copied. If initiatingFrameSU is kXMPNULL, all the Storage Units reachable from this Storage Unit will be copied.
Note that the actual copying may not be completed until after EndClone is finished.

Inputs

key	XMPDraftKey identifying the Clone transaction.
destStorageUnit	Destination Storage Unit to where this Stoage Unit is copied.

initiatingFrameSU	All Storage Units cloned from this point on should be within the scope of this XMPFrame.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

CloneTo

Function Prototype

```
XMPStorageUnit* CloneTo(
        XMPDraftKey key,
        XMPDraft* toDraft,
        XMPStorageUnit* initiatingFrameSU);
```

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Deep copies properties and values from the XMPStorageUnit this XMPStorageUnitView is viewing to a newly created Storage Unit in the Draft referred to by toDraft. A XMPStorageUnit referring to the newly created Storage Unit is returned.

In order to avoid copying extra Storage Units during the deep-copy, an initiatingFrameSU may be specified. If initiatingFrameSU is not kXMPNULL, only those Storage Units reachable from initiatingFrameSU will be copied. If initiatingFrameSU is kXMPNULL, all the Storage Units reachable from this Storage Unit will be copied.

Note that the actual copying may not be completed until after EndClone is finished.

Inputs

key	XMPDraftKey identifying the Clone transaction.
toDraft	Destination Draft when a new Storage Unit will be created for the Clone operation.
initiatingFrameSU	All Storage Units cloned from this point on should be within the scope of this XMPFrame.

Outputs

none

Exceptions Signalled

none

Pre conditions

None.

Post conditions

The focus of the viewed XMPStorageUnit is undefined.

CopyTo

Function Prototype

```
void CopyTo(
        XMPStorageUnit* toSU);
```

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Shallow copies properties and values from the XMPStorageUnit this XMPStorageUnitView is viewing to toSU.

Inputs

toSU	the storage unit to which data is to be copied

Outputs

none

Exceptions Signalled

none

Pre conditions

None.

Post conditions

All properties and values of the XMPStorageUnit this XMPStorageUnitView object is viewing have the respective properties and values in toSU.
The focus of the viewed XMPStorageUnit is undefined.

DeleteValue

Function Prototype

```
void   DeleteValue(
           XMPULong length);
```

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Delete length bytes at the current offset (inclusive) in the focused Value.
If the focused Value is a Promise Value, the Promise is resolved first before insertion is done.

Inputs

offset	the offset into the focussed value
length	the length of bytes to delete from the focussed value

Outputs

none

Exceptions Signalled

none

Pre conditions

Its XMPStorageUnitCursor corresponds to a particular value of the XMPStorageUnit.

Post conditions

length bytes at the current offset in the currently focused value have been deleted.
The focus of the viewed XMPStorageUnit is on the context specified by the cursor of this XMPStorageUnitView object.

Externalize

Function Prototype

XMPStorageUnitView* Externalize();

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Makes all changes of the XMPStorageUnit this XMPStorageUnitView object is viewing persistent.

Inputs

none

Outputs

<return>	this

Exceptions Signalled

none

Pre conditions

None.

Post conditions

All properties and values in the associated XMPStorageUnit have persistently written out and are no longer dependent on their ephemeral storage representation.

The focus of the viewed XMPStorageUnit is undefined.

GetCursor

Function Prototype

XMPStorageUnitCursor* GetCursor();

Protocol

Object Storage

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the XMPStorageUnitCursor used to focus the XMPStorageUnit.

Inputs

none

Outputs

<return>	XMPStorageUnitCursor which provides the context for this XMPStorageUnitView object.

Exceptions Signalled

none

Pre conditions

None.

Post conditions

None.

GetGenerationNumber

Function Prototype

XMPULong GetGenerationNumber();

Protocol

Generation

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the generation number of the Value this XMPStorageUnitView object is viewing.

Inputs

none

Outputs

<return>	Generation number of the focused value

Exceptions Signalled

None.

Pre conditions

Its XMPStorageUnitCursor corresponds to a particular value of the XMPStorageUnit.

Post conditions

The focus of the viewed XMPStorageUnit is on the context specified by the cursor of this XMPStorageUnitView object.

GetID

Function Prototype

```
XMPID GetID();
```

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the persistent ID of the XMPStorageUnit object this XMPStorageUnitView object is viewing.

Inputs

none

Outputs

<return>	ID of the XMPStorageUnit object this object is viewing.

Exceptions Signalled

none

Pre conditions

None.

Post conditions

None.

GetIDFromStorageUnitRef

Function Prototype

```
XMPStorageUnitID GetIDFromStorageUnitRef(
        XMPStorageUnitRef ref);
```

Protocol

Storage Unit Reference

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Resolves the given ref to a XMPStorageUnitID and returns the XMPStorageUnitID. If the specified ref cannot be resolved, an exception is raised.

Inputs

ref	persistent reference to be resolved.

Outputs

XMPStorageUnitID	ID of the embedded StorageUnit.

Exceptions Signalled

kXMPNotValueFocussed

Pre conditions

Its XMPStorageUnitCursor corresponds to a particular value of the XMPStorageUnit.
ref must be generated by GetStrongStorageUnitRef or GetWeakStorageUnitRef.

Post conditions

The focus of the viewed XMPStorageUnit is on the context specified by the cursor of this XMPStorageUnitView object.

GetName

Function Prototype

```
XMPName* GetName();
```

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the name of the Storage Unit this XMPStorageUnitView is viewing.

Inputs

none

Outputs

<return>	the name of the focus

Exceptions Signalled

none

Pre conditions

None.

Post conditions

None.

GetOffset

Function Prototype

```
XMPULong  GetOffset();
```

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Return the offset of the Value this XMPStorageUnitVeiw is viewing.

Inputs

none

Outputs

<return>	Offset of the current Value.

Exceptions Signalled

kXMPNotValueFocussed

Pre conditions

Its XMPStorageUnitCursor corresponds to a particular Value of the Storage Unit.

Post conditions

None.

GetPromiseValue

Function Prototype

```
XMPULong    GetPromiseValue(
                XMPValueType valueType,
                XMPULong offset,
                XMPULong length,
                XMPValue value,
                XMPPart **sourcePart);
```

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Get the promise data in the Value this XMPStorageUnitView object is viewing. This does not resolve the promise.

Inputs

valueType	Type of the Value containing the desired promise data.
offset	Offset from where the desired promise data should be retrieved.
length	Length of desired promise data.
value	Buffer used to contain the desired promise data.

Outputs

<return>	Number of bytes read.
value	Buffer containing the promise data.

Exceptions Signalled

None.

Pre conditions

Its XMPStorageUnitCursor corresponds to a particular value of the XMPStorageUnit.

Post conditions

length bytes have been read from offset in the value of the given valueType into the value parameter.

The focus of the viewed XMPStorageUnit is on the context specified by the cursor of this XMPStorageUnitView object.

GetProperty

Function Prototype

```
XMPPropertyName    GetProperty();
```

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the Property Name in the cursor of this XMPStorageView object.

Inputs

None.

Outputs

<return>	Property name.

Exceptions Signalled

None.

Pre conditions

Its XMPStorageUnitCursor corresponds to a particular property or value of the XMPStorageUnit.

Post conditions

The focus of the viewed XMPStorageUnit is on the context specified by the cursor of this XMPStorageUnitView object.

GetSize

Function Prototype

```
XMPULong GetSize();
```

Protocol

Storage Unit Manipulation

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the size of the focused Property or Value this XMPStorageUnitView object is viewing.

Inputs

None.

Outputs

<return>	Size of the focused Property or Value this XMPStorageUnitView object is viewing.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetStorageUnit

Function Prototype

```
XMPStorageUnit* GetStorageUnit();
```

Protocol

Object Storage

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the XMPStorageUnit that this XMPStorageUnitView is viewing. The returned XMPStorageUnit should be focused using the XMPStorageUnitCursor stored in this XMPStorageUnitView object.

Inputs

none

Outputs

<return>	XMPStorageUnit this XMPStorageUnitView is viewing.

Exceptions Signalled

none

Pre conditions

None.

Post conditions

None.

GetStorageUnitRefIterator

Function Prototype

```
XMPStorageUnitRefIterator* GetStorageUnitRefIterator();
```

Protocol

Iteration

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns an iterator for iterating over all the persistent references created in the Value this XMPStorageUnitRefIterator is viewing.

Inputs

None.

Outputs

<return>	Iterator used to iterate over all persistent references in the focused Value.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetStrongStorageUnitRef

Function Prototype

```
XMPStorageUnitRef GetStrongStorageUnitRef(
        XMPStorageUnit* embeddedSU);
```

Protocol

Storage Unit Reference

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns a persistent reference to the Storage Unit refered to by embeddedSU. This persistent reference may be stored within the Value this XMPStorageUnitView is viewing and later used to retrieve the given Storage Unit. The returned XMPStorageUnitRef is strong in the sense that the Storage Unit it refers to will be copied in the CloneTo operation.

Inputs

embeddedSU	XMPStorageUnit referring to the Storage Unit whose persistent reference is desired.

Outputs

<return>	persistent reference to the Storage Unit referred to by embeddedSU.

Exceptions Signalled

none

Pre conditions

Its XMPStorageUnitCursor corresponds to a particular value of the XMPStorageUnit.

Post conditions

The focus of the viewed XMPStorageUnit is on the context specified by the cursor of this XMPStorageUnitView object.

GetType

Function Prototype

```
XMPValueType GetType();
```

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns the type of the Value in the cursor of this XMPStorageUnitView object.

Inputs

none

Outputs

<return>	the type of the focused particular value

Exceptions Signalled

none

Pre conditions

Its XMPStorageUnitCursor corresponds to a particular value of the XMPStorageUnit.

Post conditions

The focus of the viewed XMPStorageUnit is on the context specified by the cursor of this XMPStorageUnitView object.

GetValue

Function Prototype

```
XMPULong  GetValue(
        XMPULong length,
        XMPValue value);
```

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Reads length bytes from current offset of the Value this XMPStorageUnitView object is viewing into value.
If the Value contains a promise, the promise is resolved first before the reading is done.
If offset+length is bigger than the size of the Value, only the bytes from the offset (inclusive) to the end of the Value will be read rendering the actual number of bytes read smaller than length.

Inputs

offset	the offset into the focussed value
length	the length of bytes to read into value
value	a buffer of size >= length

Outputs

<return>	Number of bytes actually read.
value	Buffer containing the read data.

Exceptions Signalled

none

Pre conditions

Its XMPStorageUnitCursor corresponds to a particular value of the XMPStorageUnit.
value is a buffer of length length or more

Post conditions

value has read length bytes from offset in the cursor-focused value
The focus of the viewed XMPStorageUnit is on the context specified by the cursor of this XMPStorageUnitView object.

GetWeakStorageUnitRef

Function Prototype

```
XMPStorageUnitRef GetWeakStorageUnitRef(
        XMPStorageUnit* embeddedSU);
```

Protocol

Storage Unit Reference

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns a persistent reference to the Storage Unit refered to by embeddedSU. This persistent reference may be stored within the Value this XMPStorageUnitView object is viewing and later used to retrieve the given Storage Unit. This XMPStorageUnitRef is weak in the sense that the Storage Unit it refers to will only be copied over in the CloneTo operation only if there is another Strong Storage Unit Ref referring to it.

Inputs

embeddedSU	XMPStorageUnit object referring to the Storage Unit whose persistent reference is desired.

Outputs

<return>	Persistent reference to the Storage Unit referred to by embeddedSU.

Exceptions Signalled

none

Pre conditions

Its XMPStorageUnitCursor corresponds to a particular value of the XMPStorageUnit.

Post conditions

The focus of the viewed XMPStorageUnit is on the context specified by the cursor of this XMPStorageUnitView object.

IncrementGenerationNumber

Function Prototype

```
XMPULong IncrementGenerationNumber();
```

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Increment and returns the generation number of the Value this XMPStorageUnitView object is viewing.

Inputs

none

Outputs

<return>	Generation number of the Storage Unit

Exceptions Signalled

None.

Pre conditions

Its XMPStorageUnitCursor corresponds to a particular value of the XMPStorageUnit.

Post conditions

The focus of the viewed XMPStorageUnit is on the context specified by the cursor of this XMPStorageUnitView object.

InitStorageUnitView

Function Prototype

```
void InitStorageUnitView(
        XMPStorageUnit* storageUnit,
        XMPStorageUnitCursor* cursor);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **must** call base class behavior, before derived class behavior.

Basic operation

Initializes this XMPStorageUnitView by storing the parameters.

Inputs

storageUnit	XMPStorageUnit this XMPStorageUnitView is associated with.
cursor	Context (Property and/or Value) in which the XMPStorageUnitView methods should work.

Outputs

none

Exceptions Signalled

none

Pre conditions

None.

Post conditions

this is an initialized XMPStorageUnitView object.

InsertValue

Function Prototype

```
void   InsertValue(
           XMPULong length,
           XMPValue value);
```

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Inserts length bytes at the current offset in the value this XMPStorageUnitView object is viewing.

If the focused Value contains a Promise, the Promise is resolved first before insertion is done.

Inputs

length	the length of bytes to write into the focussed value
value	a buffer of length>= length

Outputs

none

Exceptions Signalled

none

Pre conditions

Its XMPStorageUnitCursor corresponds to a particular value of the XMPStorageUnit.

value is a buffer of length length or more

Post conditions

length bytes of value have been inserted at offset in the currently focused value

The focus of the viewed XMPStorageUnit is on the context specified by the cursor of this XMPStorageUnitView object.

Internalize

This method is only of interest to Container Application developers.

Function Prototype

```
XMPStorageUnitView* Internalize();
```

Protocol

ObjectStorage

Protection

Public. Private within Storage System.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Internalizes the XMPStorageUnit object this XMPStorageUnitView object is viewing.

Inputs

none

Outputs

<return>	this

Exceptions Signalled

none

Pre conditions

None.

Post conditions

all properties and values in the focus have been brought into ephemeral storage from persistent storage

IsPromiseValue

Function Prototype

```
XMPBoolean IsPromiseValue();
```

Protocol

Promise

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns kXMPTrue if the Value this XMPStorageUnitView object is viewing contains a Promise. Otherwise, returns kXMPFalse.

If the focused Value is a Promise, the Promise will not be resolved by this call.

Inputs

none

Outputs

<return>	XMPBoolean showing whether the focused Value contains a Promise.

Exceptions Signalled

None.

Pre conditions

Its XMPStorageUnitCursor corresponds to a particular value of the XMPStorageUnit.

Post conditions

The focus of the viewed XMPStorageUnit is on the context specified by the cursor of this XMPStorageUnitView object.

IsStrongStorageUnitRef

Function Prototype

```
XMPBoolean IsStrongStorageUnitRef(
        XMPStorageUnitRef ref);
```

Protocol

Storage Unit Reference

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns kXMPTrue if ref is a strong XMPStorageUnitRef. Otherwise, returns kXMPFalse.

Inputs

ref	XMPStorageUnitRef to be tested.

Outputs

<return>	Boolean showing whether ref is a strong XMPStorageUnitRef.

Exceptions Signalled

None.

Pre conditions

Its XMPStorageUnitCursor corresponds to a particular value of the XMPStorageUnit.

Post conditions

The focus of the viewed XMPStorageUnit is on the context specified by the cursor of this XMPStorageUnitView object.

IsWeakStorageUnitRef

Function Prototype

```
XMPBoolean IsWeakStorageUnitRef(
        XMPStorageUnitRef ref);
```

Protocol

Storage Unit Reference

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Returns kXMPTrue if ref is a weak XMPStorageUnitRef. Otherwise, returns kXMPFalse.

Inputs

ref	XMPStorageUnitRef to be tested.

Outputs

<return>	Boolean showing whether the ref is a weak XMPStorageUnitRef.

Exceptions Signalled

None.

Pre conditions

Its XMPStorageUnitCursor corresponds to a particular value of the XMPStorageUnit.

Post conditions

The focus of the viewed XMPStorageUnit is on the context specified by the cursor of this XMPStorageUnitView object.

Purge

Function Prototype

```
XMPSize   Purge(
        XMPSize size);
```

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Purge memory from ephemeral store until size bytes (not necessarily contiguous) have been freed up.

Inputs

size	the number of bytes to purge

Outputs

<return>	the number of bytes actually purged

Exceptions Signalled

none

Pre conditions

None.

Post conditions

Either size bytes (not necessarily contiguous) are free in the default heap or all properties and values in the associated XMPStorageUnit have been flushed out.

Remove

Function Prototype

```
XMPStorageUnitView* Remove();
```

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Removes the value or property this XMPStorageUnitView object is viewing. This is a dangerous operation in the sense that this method makes the view of this XMPStorageUnitView invalid. This method should be used with caution.

Inputs

none

Outputs

<return>	this

Exceptions Signalled

none

Pre conditions

Its XMPStorageUnitCursor corresponds to a particular property or value of the XMPStorageUnit.

Post conditions

the property or value at the cursor focus is removed
The focus of the viewed XMPStorageUnit is undefined.

RemoveStorageUnitRef

Function Prototype

```
XMPStorageUnitView* RemoveStorageUnitRef(
        XMPStorageUnitRef ref);
```

Protocol

Storage Unit Reference

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Removes a XMPStorageUnitRef from a Value.

Inputs

ref	persistent reference to a Storage Unit.

Outputs

<return>	this XMPStorageUnitView object.

Exceptions Signalled

None.

Pre conditions

Its XMPStorageUnitCursor corresponds to a particular value of the XMPStorageUnit.
ref is a valid XMPStorageUnitRef (either strong or weak).

Post conditions

ref is no longer a valid XMPStorageUnitRef in that Value.

SetName

Function Prototype

```
void SetName(
        XMPName name);
```

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Set the name of the Storage Unit this XMPStorageUnitView is viewing.

Inputs

name	the name of the focus

Outputs

none

Exceptions Signalled

none

Pre conditions

None.

Post conditions

the name of the focus = name

SetOffset

Function Prototype

```
void   SetOffset(
           XMPULong offset);
```

Protocol

Value Manipulation

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Sets the offset of the value this XMPStorageUnitView object is viewing.

Inputs

offset	Offset to use with the focused value

Outputs

none

Exceptions Signalled

none

Pre conditions

Its XMPStorageUnitCursor corresponds to a particular value of the Storage Unit.

Post conditions

the current offset for value operations is offset.

SetPromiseValue

Function Prototype

```
void  SetPromiseValue(
        XMPValueType valueType,
        XMPULong offset,
        XMPULong length,
        XMPValue value);
```

Protocol

Promise

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Put the given data into the value this XMPStorageUnitView is viewing and make the viewed value a promise.

Inputs

valueType	Type of Value to be containing the Promise data.
offset	Offset from which the data should be written.
length	Length of data to be written.
value	Buffer containing the data to be written.

Outputs

none

Exceptions Signalled

None.

Pre conditions

Its XMPStorageUnitCursor corresponds to a particular value of the XMPStorageUnit.

Post conditions

length bytes of value have been written into offset in the focused value. Also, the StorageUnit is updated to reflect that the value is a promise.
The focus of the viewed XMPStorageUnit is on the context specified by the cursor of this XMPStorageUnitView object.

SetType

Function Prototype

```
void  SetType(
        XMPValueType valueType);
```

Protocol

Value manipulation

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Sets the type of the Value of the Storage Unit this XMPStorageUnitView is viewing.

Inputs

valueType	Type of the focused Value.

Outputs

Exceptions	Signalled

Exceptions Signalled

none

Pre conditions

Its XMPStorageUnitCursor corresponds to a particular value of the XMPStorageUnit.

Post conditions

the type of that value = valueType

The focus of the viewed XMPStorageUnit is on the context specified by the cursor of this XMPStorageUnitView object.

SetValue

Function Prototype

```
void   SetValue(
           XMPULong length,
           XMPValue value);
```

Protocol

ObjectStorage

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Write length bytes to the current offset of the Value this XMPStorageUnitView is viewing.

If length + offset is greater than the current size of the Value, the Value should be grown to accomodate the extra bytes.

Inputs

length	the length of bytes to write into the focussed value
value	a buffer of length>= length

Outputs

None

Exceptions Signalled

none

Pre conditions

Its XMPStorageUnitCursor corresponds to a particular value of the XMPStorageUnit.

value is a buffer of length length or more

Post conditions

length bytes of value have been written into offset in the focused Value.

The focus of the viewed XMPStorageUnit is on the context specified by the cursor of this XMPStorageUnitView object.

XMPTransform

Basic Class Documentation

Base class: XMPObject.
Class XMPTransform has no friends.
XMPTransform objects are used to define transformations between coordinate spaces in different XMPFrames. More background information can be found in the XMPFrame class documentation.

Theory of Operation

There are different flavors of Transforms for different graphics models. F'rinstance, regular QuickDraw transforms are different than GX transforms in that they only allow offsets, while GX allows the full panoply of scale/rotate/skew/perspective.
In general, a transform uses a 3x3 matrix to perform 2-d transformations of XMPPoints. Transforms can be concatenated by multiplying their matrics, resulting in an "aggregate transform". See any standard computer graphics text for gory details.
(In classic QuickDraw, however, the data is merely an XMPPoint representing the offset.)
The data representing the transform are kept in a platform-specific (or graphics-system-specific) data structure and can be updated by the caller of this class directly or rather by calling the appropiate transformation method. The type of this data structure is cached in a storage member.
Conversion of a parameter's transform data type to a receiving object's transform data type supports multiple graphics systems in one document.

Invariants Maintained by Class

The fPlatformTransform data member holds a reference (pointer, handle, etc.) to a platform-specific transformation data structure.
The type of the graphics system in which fPlatformTransform is represented is stored in fGraphicsSystem.

Other Persistent Properties

Member Functions

XMPTranslation

Basic Class Documentation

XMPTranslation is not derived from any class. However, it depends on the system service which provides data translation. The system service should also allow users to query what kind of translation is available.
The class documented here is an abstract base class. Platform implementors must subclass this class to provide the OpenDoc Translation functionalities on their platforms.

This object is instantiated by XMPSession at session startup time only. Parts should not instantiate this object even though Parts are the main client of this object.

Theory of Operation

XMPTranslation provides data translation capability to OpenDoc documents and their parts. At the process' startup time, a XMPTranslation object is instantiated and stored with the XMPSession. Whenever a part needs the translation service, it can acquire the XMPTranslation object through the XMPSession object.
The translation service can be triggered when a part does not know how to handle data of an unfamiliar type (e.g., when importing data from XMPClipboard or XMPDragAndDrop). The part can ask XMPTranslation to find out to what types the data can be translated. Alternatively, the part may ask XMPTranslation whether the data can be translated to one or more types that the part can handle. If the desired types are available, the part can then have XMPTranslation done the actual conversion of data.

Invariants Maintained by Class

The translation methods provided by XMPTranslation should include all those provided by the system translation service.

Other Persistent Properties

Member Functions

XMPTranslation

This method is only of interest to Container Application developers.

Function Prototype

```
XMPTranslation();
```

Protocol

Constructor

Protection

Public. Private between XMPSession and XMPTranslation. This method should only be called by XMPSession::InitSession.

Override policy

Derived class **must** override.

Derived class **must** call base class behavior, before derived class behavior.

Basic operation

This function is the constructor of the XMPTranslation object. It is called once at the process' startup time.

Inputs

None

Outputs

None

Exceptions Signalled

None

Pre conditions

None.

Post conditions

this is an uninitialized XMPTranslation object.

~XMPTranslation

This method is only of interest to Container Application developers.

Function Prototype

```
~XMPTranslation();
```

Protocol

Destructor

Protection

Public. Private between XMPSession and XMPTranslation. This method should only be called by XMPSession::~XMPSession or the equivalent shutdown method.

Override policy

Derived class **must** override.

Derived class **must** call base class behavior, after derived class behavior.

Basic operation

This function is the destructor of the XMPTranslation object. It is called once when the process quits. It closes the system translation service.

Inputs

None

Outputs

None

Exceptions Signalled

None

Pre conditions

None.

Post conditions

None.

GetISOTypeFromPlatformType

Function Prototype

```
XMPType GetISOTypeFromPlatformType(
        XMPPlatformType platformType,
        XMPPlatformTypeSpace typeSpace);
```

Protocol

Translation

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the ISO type corresponding to the argument platform type in the context of the argument type space.

Inputs

platformType	The platform specific type
typeSpace	The context in which to interprete the type

Outputs

<result>	The ISO type

Exceptions Signalled

None.

Pre conditions
None.
Post conditions
None.

GetPlatformTypeFromISOType

Function Prototype

```
XMPPlatformType GetPlatformTypeFromISOType(
        XMPType type);
```

Protocol
Translation
Protection
Public.
Override policy
Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.
Basic operation
This function returns the platform type corresponding to the argument ISO type.
Inputs

type	The ISO type.

Outputs

<result>	The platform type.

Exceptions Signalled
None.
Pre conditions
None.
Post conditions
None.

GetTranslateMethod

Function Prototype

```
XMPBoolean GetTranslateMethod(
        XMPTypeSet* givenTypes,
        XMPType nativeType,
        XMPTranslateResult* result,
        XMPTranslateMethod* howToTranslate);
```

Protocol
Translation
Protection
Public.
Override policy
Derived class **must** override.

Derived class **cannot** call base class behavior.
Basic operation
Queries the system translation service to see whether a given type (or a set of types) can be translated to another type (i.e., nativeType.)
If the translation can be done, this function will return kCanTranslate and some private data in howToTranslate. Typically, howToTranslate contains information for XMPTranslation::Translate to do the actual translation. If the caller does not want to continue the translation process, it can dispose of howToTranslate using the OpenDoc memory management calls.
If the given types cannot be translated into the native type, kCannotTranslate is returned.
Both givenTypes and nativeType are unchanged by this function.
Inputs

givenTypes	Types of the source data.

	nativeType	Type that the part can handle.
Outputs		
	<return>	true if translation to nativeType is possible
	result	kCanTranslate kCannotTranslate ???
	howToTranslate	Hints used in XMPTranslation::Translate for the actual translation. Callers should use this as a blackbox.
Exceptions Signalled		
	kXMPErrEmptySet	givenTypes is empty.

Pre conditions

givenTypes is a non-empty set.

Post conditions

None.

GetTranslationOf

Function Prototype

```
XMPTypeSet* GetTranslationOf(
        XMPType fromType);
```

Protocol

Translation

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

This function returns a XMPTypeSet that contains all the types to which fromType can be translated. If translation cannot be done on fromType, an empty set is returned.

Inputs		
	fromType	Type of the source data.
Outputs		
	<return>	A set of types that can be translated to from the source data.
Exceptions Signalled		
	kXMPErrOutOfMemory	Cannot create a XMPTypeSet because the system is running out of memory.

Pre conditions

None.

Post conditions

fromType is unchanged.

InitTranslation

This method is only of interest to Container Application developers.

Function Prototype

```
void InitTranslation();
```

Protocol

Constructor

Protection

Public. Private between XMPSession and XMPTranslation. This method should only be called by XMPSession::InitSession.

Override policy

Derived class **must** override.

Derived class **must** call base class behavior, before derived class behavior.

Basic operation

Initializes the system translation service. It is called once at the process' startup time.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

this is an initialized XMPTranslation object.

Purge

This method is only of interest to Container Application developers.

Function Prototype

```
XMPSize Purge(
        XMPSize size);
```

Protocol

Memory Management

Protection

Public. Private between XMPSession and XMPTranslation. This should only be called by XMPSession.

Override policy

Derived class **must** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Purges memory from ephemeral store until size bytes (not necessarily contiguous) have been freed up.

Inputs

size	Number of bytes to purge.

Outputs

<return>	Number of bytes actually purged.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

Translate

Function Prototype

```
void Translate(
        XMPStorageUnitView* fromView,
        XMPTranslateMethod howToTranslate,
        XMPStorageUnitView* toView);
```

Protocol

Translation

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Translates the Value viewed by fromView and stores the translated data in the Value viewed by toView using howToTranslation as translation hints.
The parameter howToTranslate is the returned result from XMPTranslation::CanTranslate . It contains the translation information needed for the translation.
This function, not the caller, is responsible for disposing howToTranslate.

Inputs

fromView	Value which contains the data to be translated.
howToTranslate	Hints generated from XMPTranslation::CanTranslate.

Outputs

<return>	Buffer containing translated data.
toView	Value where the translated data should be stored.

Exceptions Signalled

kXMPErrBadTranslationMethod	Corrupted XMPTranslationMethod.
kXMPErrOutOfMemory	Cannot create a buffer to contain the translated data because the system is running out of memory.
kXMPErrCannotTranslate	Data cannot be translated using the XMPTranslationMethod.
kXMPErrCannotDisposeTranslationMethod	TranslationMethod cannot be disposed.

Pre conditions

howToTranslate is the return result from XMPTranslation::CanTranslate and it contains information for this translation.

Post conditions

The content of Value viewed by fromView is unchanged.
howToTranslate is disposed of.

Translate

Function Prototype

```
XMPVMethod void Translate(
        XMPPtr fromData,
        XMPULong fromDataSize,
        XMPType givenType,
        XMPTranslateMethod howToTranslate,
        XMPPtr* toData,
        XMPULong* toDataSize);
```

Protocol

Translation

Protection

Public.

Override policy

Derived class **must** override.

Derived class **cannot** call base class behavior.

Basic operation

Translates the data in the buffer pointed to by fromData (of length fromDataSize) and stores the translated data in the buffer pointed by toData using howToTranslate.
If the buffer pointed to by toData is not large enough to store the translated data, an exception is raised.
The parameter howToTranslate is the returned result from XMPTranslation::CanTranslate . It contains the translation information needed for the translation.

This method, not the caller, is responsible for disposing howToTranslate.

Inputs

fromData	Source data to be translated.
givenType	Type of source data.
howToTranslate	Hints generated from XMPTranslation::CanTranslate.

Outputs

<return>	Buffer containing translated data.

Exceptions Signalled

kXMPErrOutOfMemory	Buffer pointed to by toData is not large enough to contain the translated data.
kXMPErrCannotTranslate	Data cannot be translated using the XMPTranslationMethod.
kXMPErrCannotDisposeTranslationMethod	TranslationMethod cannot be disposed.

Pre conditions

howToTranslate is the return result from XMPTranslation::CanTranslate and it contains information for this translation.

Post conditions

The content referred to by fromData is unchanged.

XMPTypeSet

Basic Class Documentation

XMPTypeSet is a collection class for XMPType objects with set semantics. XMPTypeSet has no base class.

Theory of Operation

XMPTypeSet is a collection of XMPTypes. Users of this class can add and remove types , query the existence of a type, and get the number of types in the class.

Parts may create instances of this class. Subclassing of XMPTypeSet is not anticipated.

The main clients of this class are XMPTranslation and parts. XMPTranslation class uses XMPTypeSet as parameters to its functions. Parts can use XMPTypeSet to store a set of Types for repeated uses.

XMPTypeSet uses OpenDoc's private collection class.

Invariants Maintained by Class

Every item in a XMPTypeSet object is unique. Therefore, a XMPTypeSet object contains a set of different XMPTypes.
A XMPTypeSet object can contain zero or more XMPTypes. In other words, the type set can either be empty or non-empty.
There is no particular ordering of XMPTypes in a XMPTypeSet object.

Other Persistent Properties

Member Functions

XMPTypeSet

Function Prototype

XMPTypeSet();

Protocol

Creation and Deletion

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **must** call base class behavior, before derived class behavior.

Basic operation

This function is the constructor of the class.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

~XMPTypeSet

Function Prototype

~XMPTypeSet();

Protocol

Creation and Deletion

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

This method is the destructor of the class.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

Add

Function Prototype

```
void Add(
        XMPType type);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Adds the argument type to the set. If the type is already present in the set, no action is taken.

Inputs

type	The type to add to the set.

Outputs

None.

Exceptions Signalled

kXMPErrOutOfMemory	Could not add the argument type to the set.
kXMPErrInvalidType	Argument type is not a valid XMPType.

Pre conditions

None.

Post conditions

The set contains the argument type.

Contains

Function Prototype

```
XMPBoolean Contains(
        XMPType type);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Test this set for the presence of the argument type.

Inputs

type	The type to test for inclusion in the set.

Outputs

<return>	kXMPTrue if the argument type is in the set and kXMPFalse otherwise.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

Count

Function Prototype

```
XMPULong Count();
```

Protocol

None.

Protection

Public. None.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, before derived class behavior.

Basic operation

This function returns the number of types in the set. If the set is empty, zero is returned.

Inputs

None.

Outputs

<return>	Number of types in the set.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

CreateTypeSetIterator

Function Prototype

```
XMPTypeSetIterator* CreateTypeSetIterator();
```

Protocol

Iteration

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Return an iterator for this set. The client should dispose of the iterator using the delete operator.

Inputs

None.

Outputs

<return>	An iterator for this set.

Exceptions Signalled

kXMPErrOutOfMemory	Could not create the iterator.

Pre conditions

None.

Post conditions

None.

Remove

Function Prototype

```
void Remove(
        XMPType type);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, before derived class behavior.

Basic operation

Removes the argument type from the set. If the type is not in the set, no action is taken.

Inputs

type	Type to be removed.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

The argument type is not in the set.

XMPTypeSetIterator

Basic Class Documentation

XMPTypeSetIterator is an iterator for the class XMPTypeSet. XMPTypeSetIterator has no base class.

Theory of Operation

This class is used to iterate over the elements of a XMPTypeSet instance. Iteration is forward only. The iteration can be restarted at any time by calling the First method. If the XMPTypeSet instance is modified during the iteration, First or Next will throw an exception. When an exception is thrown, the iterator cannot be used further, and should be destroyed.

Parts should create an XMPTypeSetIterator by calling the CreateTypeSetIterator method of the XMPTypeSet instance to be iterated over. The iterator should be disposed using the delete operator.

Subclassing of XMPTypeSetIterator is not anticipated.

The implementation of XMPTypeSetIterator uses OpenDoc's private collection class.

Invariants Maintained by Class

Other Persistent Properties

Member Functions

~XMPTypeSetIterator

Function Prototype

~XMPTypeSetIterator();

Protocol

Creation and Deletion

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Class instance destructor.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

First

Function Prototype

XMPType First();

Protocol

Iteration

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Initiates iteration and returns an arbitrary first element from the set. After creating an iterator using XMPTypeSet::CreateTypeSetIterator, call this method once, then call Next repeatedly to iterate through all elements of the set. First can be called at any time to restart the iteration.

Inputs

None.

Outputs

<return>	An arbitrary XMPType object in the set. If the set is empty kXMPNULL is returned, although the client should call IsNotComplete to test for completion.

Exceptions Signalled

kXMPErrIteratorOutOfSync	The XMPTypeSet instance was changed during the iteration.

Pre conditions

None.

Post conditions

None.

IsNotComplete

Function Prototype

```
XMPBoolean IsNotComplete();
```

Protocol

Iteration

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Test if iteration is complete.

Inputs

None.

Outputs

<return>	Returns kXMPTrue if the last call to First or Next returned a valid XMPType element, and kXMPFalse otherwise.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

Next

Function Prototype

```
XMPType Next();
```

Protocol

Iteration

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns an arbitrary next element from the set. First should be called prior to this routine to start the iteration. Next can be called repeatedly until IsNotComplete returns kXMPTrue.

Inputs

None.

Outputs

<return>	<return>

Exceptions Signalled

kXMPErrIteratorOutOfSync	The XMPTypeSet instance was changed during the iteration.

Pre conditions

First has been called to start the iteration.

Post conditions

None.

XMPUndo

Basic Class Documentation

XMPUndo is used to store command history.
XMPUndo is a derived class of XMPObject.
XMPUndo is largely platform-independent.
If per-document undo is supported, XMPUndo is closely tied to XMPDraft. Each draft should contain at most one XMPUndo object.

Theory of Operation

XMPUndo is a session-wide object used to store command history.
The OpenDoc session object creates an XMPUndo object. Parts and the OpenDoc shell access XMPUndo through XMPSession. If thread safety is a concern, the part should acquire the undo focus fromXMPArbitrator first. As soon as the part is done with the XMPUndo object, it should reliquish the undo focus at once.
Conceptually, XMPUndo object contains two stacks -- the Undo Stack and the Redo Stack. When an action is done, the action data provided by the part is put on the Undo Stack. When an action needs to be "undone", the action data is popped from the Undo stack onto the Redo stack. At the same time, the part is also notified so that it can undo its previous action using the stored action data. When an undone action needs to be redone, the action data is popped from the Redo Stack back to the Undo Stack. The part is also notified so that it can redo the undone action.
The Undo Stack and Redo Stack can be cleared upon request. When clearing is to be done, XMPUndo asks each part to dispose of its action data stored in the Stacks. The order of disposal is done from older actions to newer actions.
There are times when a subhistory is required. For example, one may need a new action context when entering a modal state e.g, a modal dialog. XMPUndo allows "marks" to be placed on the Stacks. If a mark is placed on a Stack, the Stack may only be cleared to the mark. In our example, when the modal dialog exits, any actions done within the context of the dialog are disposed of. However, all the actions done before the modal dialog are preserved in the Stacks.
XMPUndo object also allows callers to peek at the top of the Undo Stack and the Redo Stack. The application shell can then peek to find out the action labels and use the labels for menu items.

Invariants Maintained by Class

XMPUndo must contain two stacks (an undo stack and a redo stack) at all times.

Other Persistent Properties

Member Functions

AddActionToHistory

Function Prototype

```
void AddActionToHistory(
        XMPPart* whichPart,
        XMPActionData actionData,
        XMPActionType actionType,
        XMPName* actionLabel);
```

Protocol

Undo

Protection

Public. Parts can call, but shell apps must not call.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

This function pushes actionData together with its associated part onto the Undo Stack.

Inputs

whichPart	the part that performed the action
actionData	Data provided the part with which it can restore itself to the pre-action state
actionType	kSingleAction means it is a single action. kBeginAction means the first of a two-part action.kEndAction means the second of a two-part action.
undoActionLabel	A user-visible label for the undo command including the word "Undo".
redoActionLabel	A user-visible label for the redo command including the word "Redo".

Outputs

None

Exceptions Signalled

kXMPErrCannotAddAction	Currently already doing an undo or redo; trying to add a nested begin action.
kXMPErrOutOfMemory	Initialization of XMPUndo object failed; Couldn't allocate memory for action info.

Pre conditions

Undo Stack has been created.

Post conditions

whichPart and actionData are on the top of the Undo Stack.

ClearActionHistory

Function Prototype

```
void ClearActionHistory(
        XMPRespectMarksChoices respectMarks);
```

Protocol

Undo

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

This function clears the Stacks down to the last marks if respectMarks is kXMPRespectMarks. Otherwise, it clears the Stacks. If kXMPRespectMarks is true, the previous marks are cleared.

Inputs

respectMarks	respect or don't respect marks.

Outputs

None

Exceptions Signalled

None

Pre conditions

Undo Stack and Redo Stack have been created.

Post conditions

Undo Stack and Redo Stack should be empty or cleared to the marks depending on respectMarks.

ClearRedoHistory

Function Prototype

```
void ClearRedoHistory();
```

Protocol

Undo

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

This function clears the Redo history. Marks are always respected.

Inputs

None

Outputs

None

Exceptions Signalled

None

Pre conditions

Redo Stack has been created.

Post conditions

Redo Stack should be empty.

MarkActionHistory

Function Prototype

```
void MarkActionHistory();
```

Protocol

Undo

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

This function places a mark on top of the Undo Stack and the Redo Stack. The marks are used to indicate the beginning of a subhistory. When XMPUndo::ClearActionHistory is done, the caller has the choice to clear the whole Stacks or to clear only down to these marks. This can be used when entering a modal state like a modal dialog, to give the effect of creating a new context.

Inputs

None

Outputs

None

Exceptions Signalled

kXMPErrCannotMarkAction XMPUndo was never initialized properly.

Pre conditions

Both Undo Stack and Redo Stack are created.

Post conditions

Marks are on top of both Undo Stack and Redo Stack.

PeekRedoHistory

Function Prototype

```
XMPBoolean PeekRedoHistory(
        XMPPart** part,
```

XMPActionData* actionData,
XMPActionType* actionType,
XMPName** actionLabel);

Protocol

Undo

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

This function returns information about the redoable action on the top of the Redo Stack. If the function returns kXMPTrue, there was an action to return, else no action was available and the information in the parameters is invalid. This function repects marks.

Inputs

None

Outputs

part	Part which performed the redoable action
actionData	Action data provided by part
actionType	kSingleAction, kBeginAction, kEndAction
actionLabel	User visible redo string. A pointer to internal storage is passed back so care should be taken to make a copy.
Return	Returns whether there was anything on the stack.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

PeekUndoHistory

Function Prototype

XMPBoolean PeekUndoHistory(
XMPPart** part,
XMPActionData* actionData,
XMPActionType* actionType,
XMPName** actionLabel);

Protocol

Undo

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

This function returns information about the undoable action on the top of the Undo Stack. If the function returns kXMPTrue, there was an action to return, else no action was available and the information in the parameters is invalid. This function repects marks.

Inputs

None

Outputs

part	Part which performed the undoable action
actionData	Action data provided by part
actionType	kSingleAction, kBeginAction, kEndAction
actionLabel	User visible undo string. A pointer to internal storage is returned so care should be taken to make a copy.
Return	Returns whether there was anything on the stack.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

Purge

Function Prototype

```
XMPSize Purge(
        XMPSize size);
```

Protocol

None.

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Frees up some memory, if possible.

Inputs

size	The amount of memory requested

Outputs

<return>	The amount of memory actually freed.

Exceptions Signalled

none

Pre conditions

None.

Post conditions

Some memory may be freed.

Redo

This method is only of interest to Container Application developers.

Function Prototype

```
void Redo();
```

Protocol

Undo

Protection

Public. Usually the container is responsible for calling this.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

This function redoes the last undone action in the Redo history.

Inputs

None

Outputs

None

Exceptions Signalled

Others…	AppleEvent errors and any error returned from the part.
kXMPErrEmptyStack	Redo stack is empty; Undo object was never initialized.

Pre conditions

Both Undo Stack and Redo Stack are created.
Redo Stack is not empty.

Post conditions

What was on top of the Redo Stack is now on top of the Undo Stack.

Undo

This method is only of interest to Container Application developers.

Function Prototype

```
void Undo();
```

Protocol

Undo

Protection

Public. Usually only called by the container app or document shell.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

This function undoes the last action in the Undo history.

Inputs

None

Outputs

None

Exceptions Signalled

Other…	AppleEvent errors or any errors returned by the part.
kXMPErrEmptyStack	Undo stack is empty; Undo object was never initialized.

Pre conditions

Both Undo Stack and Redo Stack are created.
Undo Stack is not empty.

Post conditions

What was on top of the Undo Stack is now on top of the Redo Stack.

XMPValueIterator

Basic Class Documentation

This class is an iterator for use with the XMPNameSpace class. It has no superclass. The platform vendor will implement.

Theory of Operation

This iterator follows the standard semantics of OpenDoc iterators except that two values are returned in pointer parameters: the key and the value. This class is a friend of XMPNameSpace. It's implementation depends heavily on the implementation of the XMPNameSpace class.

Invariants Maintained by Class

It is only meant to be used in the context of a for statement. Any other use is not guaranteed to be correct.

Other Persistent Properties

Member Functions

XMPValueIterator

Function Prototype

```
XMPValueIterator(
        XMPNameSpace* nameSpace);
```

Protocol

Name Binding

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Initialize the private member variables.

Inputs

nameSpace	The XMPNameSpace over which to iterate.

Outputs

None

Exceptions Signalled

None

Pre conditions

A valid XMPNameSpace must exist.

Post conditions

None.

~XMPValueIterator

Function Prototype

```
~XMPValueIterator();
```

Protocol

Name Binding

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Deletes private storage.

Inputs

None.

Outputs

None

Exceptions Signalled

None

Pre conditions

None.

Post conditions

None.

First

Function Prototype

```
void First(
        XMPISOStr* key,
        XMPPtr* value);
```

Protocol

Name Binding

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Return the first key/value pair. If there are no entries in the table, the values returned are garbage and the next call to IsNotComplete will return false. Note that the XMPISOStr returned is a copy. The client must dispose of it.

Inputs

key	a pointer to the key to be returned.
value	a pointer to the value to be returned.

Outputs

key	the key placed in the storage pointed to by key.
value	the value placed in the storage pointed to by value .

Exceptions Signalled

kXMPErrWrongType	this is thrown if the type of the key does not match the type of the XMPNameSpace with which the iterator was initialized.
kXMPErrOutOfMemory	Not enough memory to begin the iteration.

Pre conditions

None.

Post conditions

None.

IsNotComplete

Function Prototype

```
XMPBoolean IsNotComplete();
```

Protocol

Name Binding

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Return whether we have tried to go beyond the end of the XMPNameSpace entries.

Inputs

None

Outputs

<return>	kXMPFalse if we have tried to get an entry that was beyond the end of the table, kXMPTrue otherwise.

Exceptions Signalled

None

Pre conditions

None.

Post conditions

None.

Next

Function Prototype

```
void Next(
        XMPISOStr* key,
        XMPPtr* value);
```

Protocol

Name Binding

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Return the next key/value pair. If there are no more entries in the table, the values returned are garbage and the next call to IsNotComplete will return false. Note that the XMPISOStr returned is a copy. The client must dispose of it.

Inputs

key	a pointer to the key to be returned.
value	a pointer to the value to be returned.

Outputs

key	the key placed in the storage pointed to by key.
value	the value placed in the storage pointed to by value .

Exceptions Signalled

kXMPErrWrongType	this is thrown if the type of the key does not match the type of the XMPNameSpace with which the iterator was initialized.

Pre conditions

First must be have been called!

Post conditions

None.

XMPWindow

Basic Class Documentation

XMPWindow is a subclass of XMPRefCntObject.
Related classes are XMPWindowState and XMPWindowIterator , XMPFrame and XMPFacet.
XMPWindow is platform-specific.

Theory of Operation

Every window must be associated with an OpenDoc XMPWindow object so that the part belonging to the root frame of the window, and embedded parts, can receive events from the OpenDoc event dispatcher.
Each XMPWindow contains a pointer to a platform-specific window structure. The facilites of the platform are used to actually create new windows. These are added to the window state using XMPWindowState::CreateWindow() which creates and returns an XMPWindow instance. Part editors can send messages to these XMPWindow instances, and obtain the platform window to make platform-specific calls. Each window has a root Frame and a root Facet.

Invariants Maintained by Class

An XMPWindow contains a reference to a platform-specific window.
An XMPWindow always has a root frame and facet.
Various other properties, such as the window type (whether it has a close box etc.) are invariant.

Other Persistent Properties

kXMPPropWindowRect
kXMPPropWindowTitle
kXMPPropWindowProcID
kXMPPropWindowIsVisible
kXMPPropWindowHasCloseBox
kXMPPropWindowHasZoomBox
kXMPPropWindowIsResizable
kXMPPropWindowIsRootWindow
kXMPPropWindowIsFloating
kXMPPropWindowHasMailer
kXMPPropWindowIsMailerAware
kXMPPropWindowRefCon
kXMPPropRootFrame
kXMPPropSourceFrame
kXMPPropShouldShowLinks

Member Functions

Close

This method is only of interest to Container Application developers.

Function Prototype

```
void Close();
```

Protocol

Display

Protection

Public. Called by the shell and window state. Parts typically call CloseAndRemove.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Closes this window. The window is hidden, removed from the window state, and the root frame is closed. The facet hierarchy is deleted.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

The window is closed, and no longer usable.

CloseAndRemove

Function Prototype

```
void CloseAndRemove();
```

Protocol

Display

Protection

Public. Called by parts to remove auxiliary windows like palettes, which are not stored persistently.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Closes this window. The window is hidden, removed from the window state, and the root frame is removed from the draft. The facet hierarchy is deleted.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

The window is closed, and no longer usable.

GetFacetUnderPoint

Function Prototype

```
XMPFrame* GetFacetUnderPoint(
        XMPPoint aPoint);
```

Protocol

Hit Detection

Protection

Public. Part Editors typically won't call this method.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the facet under the given point. If several nested facets are under the point, the innermost one is returned. If multiple overlapping frames are under the point, the unobscured(i.e. frontmost) one is returned. The "frozen" and "selected" properties of frames and facets are respected.

Inputs

aPoint	A point in window coordinates.

Outputs

<return>	The facet under the point.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

No effect on this instance.

GetID

Function Prototype

```
XMPID GetID();
```

Protocol

Invariant Properties

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the ID of the window. The WindowState creates IDs for windows. A window pointer can be obtained by calling XMPWindowState::getWindow. In this way, parts can detect that a window has been destroyed.

Inputs

None.

Outputs

<return>	The ID of this window.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

No effect on this instance.

GetPlatformWindow

Function Prototype

```
XMPPlatformWindow GetPlatformWindow();
```

Protocol

Invariant Properties

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the platform-specific window data structure

Inputs

None.

Outputs

<return>	The platform-specific window stored in fPlatformWindow.

Exceptions Signalled

None.

Pre conditions
A valid initialized instance.
Post conditions
No effect on this instance.

GetRootFrame

Function Prototype
XMPFrame* GetRootFrame();
Protocol
Invariant Properties
Protection
Public.
Override policy
Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.
Basic operation
Returns the root frame of the window.
Inputs
None.
Outputs
<return> The root frame of the window.
Exceptions Signalled
None.
Pre conditions
A valid initialized instance.
Post conditions
No effect on this instance.

GetSourceFrame

Function Prototype
XMPFrame* GetSourceFrame();
Protocol
Invariant Properties
Protection
Public.
Override policy
Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.
Basic operation
Returns the frame from which this window was opened. May be kXMPNULL.
Inputs
None.
Outputs
<return> The frame from which this window was opened, or kXMPNULL
Exceptions Signalled
None.
Pre conditions
A valid initialized instance.
Post conditions
No effect on this instance.

Hide

Function Prototype
void Hide();

Protocol

Display

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Makes the window invisible.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

The window is hidden.

IsActive

Function Prototype

XMPBoolean IsActive();

Protocol

Layers

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns kXMPTrue if this window is a floating window, or the frontmost non-floating window.

Inputs

None.

Outputs

<return> kXMPTrue, if this is an active window.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

No effect on this instance.

IsResizable

Function Prototype

XMPBoolean IsResizable();

Protocol

Invariant Properties

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns kXMPTrue if the window is resizable by users, kXMPFalse otherwise.

Inputs

None.

Outputs

<return>	kXMPTrue if the window is resizable, kXMPFalse otherwise

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

No effect on this instance.

IsRootWindow

Function Prototype

```
XMPBoolean IsRootWindow();
```

Protocol

Invariant Properties

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns kXMPTrue if the window is a root window, kXMPFalse otherwise.

Inputs

None.

Outputs

<return>	kXMPTrue if the window is a root window, kXMPFalse otherwise

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

No effect on this instance.

IsShown

Function Prototype

```
XMPBoolean IsShown();
```

Protocol

Display

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns kXMPTrue if the window is currently shown, kXMPFalse otherwise.

Inputs

None.

Outputs

<return>	kXMPTrue if the window is currently shown, kXMPFalse otherwise

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

No effect on this instance.

Open

Function Prototype

```
void Open();
```

Protocol

Display

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Creates a facet hierarchy in this window. Does not make the window visible or change window ordering. Show() and Select() must be called for those operations.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

The window and facet structure is build, and the window can be shown using Show().

Purge

This method is only of interest to Container Application developers.

Function Prototype

```
XMPSize Purge(
        XMPSize size);
```

Protocol

Low Memory

Protection

Public. Called by OpenDoc.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Frees up some memory, by deleting cached values which can be recomputed or restored from persistent storage.

Inputs

size	The amount of memory requested.

Outputs

<return>	The amount of memory freed.

Exceptions Signalled

None.

Pre conditions
A valid initialized instance.
Post conditions
Some memory has been freed up.

Resized

Function Prototype
```
void Resized();
```
Protocol
Display
Protection
Public.
Override policy
Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.
Basic operation
Reshapes the root frame to match the window size. Should be called by parts if they resize a window programmatically
Inputs
None.
Outputs
None.
Exceptions Signalled
None.
Pre conditions
A valid initialized instance.
Post conditions
The root frame matches the window shape.

Select

This method is Macintosh specific.
Function Prototype
```
void Select();
```
Protocol
Layers
Protection
Public.
Override policy
Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.
Basic operation
Brings the window to the front, making it an active window.
Inputs
None.
Outputs
None.
Exceptions Signalled
None.
Pre conditions
A valid initialized instance.
Post conditions
The window is frontmost (except for floating windows), and active.

SetShouldSave

Function Prototype

```
void SetShouldSave(
        XMPBoolean shouldSave);
```

Protocol

'Ternalization

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Changes the value of the "shouldSave" property

Inputs

shouldSave	kXMPTrue if the window should be saved in the draft.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

fShouldSave is updated.

SetShouldShowLinks

Function Prototype

```
void SetShouldShowLinks(
        XMPBoolean shouldShowLinks);
```

Protocol

Display

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Changes the value of the "shouldShowLinks" property

Inputs

shouldShowLinks	kXMPTrue if links should be shown, kXMPFalse otherwise.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

fShouldShowLinks is updated.

ShouldSave

Function Prototype

```
XMPBoolean ShouldSave();
```

Protocol

'Ternalization

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns kXMPTrue if the window is saved in the draft, kXMPFalse otherwise.

Inputs

None.

Outputs

<return>	kXMPTrue if the window is saved in the draft, kXMPFalse otherwise

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

No effect on this instance.

ShouldShowLinks

Function Prototype

```
XMPBoolean ShouldShowLinks();
```

Protocol

Display

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns kXMPTrue if the user has requested that links be highlighted in the window.

Inputs

None.

Outputs

<return>	kXMPTrue if links should be shown, kXMPFalse otherwise

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

No effect on this instance.

Show

Function Prototype

```
void Show();
```

Protocol

Display

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Makes the window visible. Does not change the ordering of windows.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

The window is visible.

XMPWindowIterator

Basic Class Documentation

XMPWindowIterator is a companion class of XMPWindowState , used to iterate over the windows in the window state.
XMPWindowIterator has no base class.
Related classes are XMPWindowState and XMPWindow.
XMPWindowIterator is platform-independent.

Theory of Operation

XMPWindowIterator is a companion class of XMPWindowState , used to iterate over the windows in the window state.
Given the window state, XMPWindowIterator allows the developer to iterate over all the windows in a loop, using the iterator's First(), Next() and IsNotComplete() methods.
Note that the window list in the window state is ordered by creation time, not by screen ordering.

Invariants Maintained by Class

XMPWindowIterator maintains a reference to the window state, and to the current window (or to a lower-level iterator).

Other Persistent Properties

Member Functions

~XMPWindowIterator

Function Prototype

~XMPWindowIterator();

Protocol

Creation and Deletion

Protection

Public.

Override policy

Derived class **cannot** override.

Derived class **cannot** call base class behavior.

Basic operation

Frees the memory allocated by this class.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

A constructed instance.

Post conditions

The memory used by this instance is freed.

First

Function Prototype

XMPWindow* First();

Protocol

Begin Iteration

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the first window in the window state. Advance iteration with Next(). Not related to front-to-back ordering of windows.

Inputs

None.

Outputs

<return>	The first window in the window state

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

Current window is advanced.

IsNotComplete

Function Prototype

```
XMPBoolean IsNotComplete();
```

Protocol

Completion Test

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns kXMPTrue if there are more windows to iterate over, kXMPFalse otherwise.

Inputs

None.

Outputs

<return>	kXMPTrue, if there are more windows to iterate over.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance. First() or Last() has been called.

Post conditions

Result contains kXMPTrue if there are more windows to iterate over, kXMPFalse otherwise.

Last

Function Prototype

```
XMPWindow* Last();
```

Protocol

Begin Iteration

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the last window in the window state. Advance iteration with Previous().

Inputs

None.

Outputs

<return> The last window in the window state

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

Current window is advanced.

Next

Function Prototype

```
XMPWindow* Next();
```

Protocol

Advance Iteration

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the next window in the window state. Begin iteration with First().

Inputs

None.

Outputs

<return> The next window in the window state

Exceptions Signalled

None.

Pre conditions

First() has been called.

Post conditions

Current window is advanced.

Previous

Function Prototype

```
XMPWindow* Previous();
```

Protocol

Advance Iteration

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the previous window in the window state. Begin iteration with Last().

Inputs

None.

Outputs

<return> The previous window in the window state

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

Current window is advanced.

XMPWindowState

Basic Class Documentation

XMPWindowState maintains a list of windows, and some menu bar information.

Each OpenDoc process has access to a single XMPWindowState object, obtained from the XMPSession object.

XMPWindowState is a derived class of XMPObject.

XMPWindowState is implemented by platform vendors. Much of this class is platform-independent, but it contains platform-specific menu information, in addition to the window list.

Related classes are XMPWindow and XMPWindowIterator.

XMPWindowState participates in the Part Activation and UI Events protocols.

Theory of Operation

XMPWindow State contains a list of windows for all open document drafts in an OpenDoc session. The standard OpenDoc shell application will have one open document, but multiple drafts of that document may be open. Traditional applications which have been modified to support embedding may have multiple documents open.

The primary purpose of the window state is to assist in event distribution.

The order of the windows in the list is insignificant.

The elements in the list of windows are instances of class XMPWindow which is also part of the OpenDoc API, so parts can call methods of XMPWindow directly.

Invariants Maintained by Class

XMPWindowState contains a list of XMPWindow objects.

XMPWindowState also contains the base menu bar object.

Other Persistent Properties

Although XMPWindow is not an XMPPersistentObject, the WindowState stored windows persistently in a special draft property.

Member Functions

ActivateFrontWindows

This method is Macintosh specific.

Function Prototype

```
void       ActivateFrontWindows();
```

Protocol

Layers

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Activates all the floating windows and the first Document window. This should be called after a modal dialog has been dismissed. DeactivateFrontWindows should be called before the modal dialog is displayed.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

DeactivateFrontWindows has been called to deactivate all the floating windows and the first Document window.

Post conditions

All the floating windows and the firstDocument window should be active.

AdjustPartMenus

Function Prototype

```
void AdjustPartMenus();
```

Protocol

Menu Bar

Protection

Public. Called when the user clicks in the menu bar.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Calls Part::AdjustMenus for the part with the menu focus, so that it can enable/disable menu items as necessary.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

Menus are set up by part with menu focus.

CloseWindows

This method is only of interest to Container Application developers.

Function Prototype

```
void CloseWindows(
        XMPDraft* draft);
```

Protocol

'Ternalization

Protection

Public. Called by the OpenDoc shell when closing a draft.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Closes all windows belonging to the specified draft.

Inputs

draft	An open OpenDoc draft

Outputs

None.

Exceptions Signalled

kXMPErrInvalidDraft	The specified draft is not valid.

Pre conditions

A valid open draft object.

Post conditions

All windows of the specified draft are closed.

CopyBaseMenuBar

Function Prototype

```
XMPMenuBar* CopyBaseMenuBar();
```

Protocol

Menu Bar

Protection

Public. Called by Part Editors.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns a copy of the base menu bar. Called by part editors to create a menu bar to which they add their own menus.

Inputs

None.

Outputs

<return>	Copy of base menu bar

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

No effect on this instance.

CreateWindow

This method is Macintosh specific.

Function Prototype

```
XMPWindow* CreateWindow(
        XMPPlatformWindow platformWindow,
        XMPBoolean isRootWindow,
        XMPBoolean isResizable,
        XMPBoolean isFloating,
        XMPBoolean shouldSave,
        XMPPart* rootPart,
        XMPTypeToken viewType,
        XMPTypeToken presentation,
        XMPFrame* sourceFrame);
```

Protocol

Creation and Deletion

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Creates and initializes a window object, and adds it to the WindowState. The supplied platform window should be created invisble. Calls to CreateWindow are usually followed by Open(), Show() and Select() calls.

Inputs

platformWindow	A pointer to a platform-specific window structure
isRootWindow	Does this window "keep the document open". kXMPFalse for dialog and other auxiliary windows.
isResizable	kXMPTrue, if the window is resizable by the user.

isFloating	kXMPTrue if the window shouldalways float above others.
shouldSave	kXMPTrue if the window state should save this window persistently in the document.
rootPart	The part associated with the root frame of the window.
viewType	The view type for the root frame of the window. See XMPFrame.
presentation	The presentation type for the root frame. See XMPFrame.
sourceFrame	The frame from which this window was opened. Can be kXMPNull.

Outputs

<return>	The new window

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

The newly-created window is in the window state.

CreateWindowIterator

Function Prototype

```
XMPWindowIterator* CreateWindowIterator();
```

Protocol

Window Access

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns an iterator for all the windows (of all drafts) in the window state.

Inputs

None.

Outputs

<return>	The iterator

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

No effect on this instance.

DeactivateFrontWindows

This method is Macintosh specific.

Function Prototype

```
void      DeactivateFrontWindows();
```

Protocol

Layers

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Deactivates all the floating windows and the first Document window. This should be called before a modal dialog is brought up.

Inputs

None.

Outputs

None.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

All the floating windows and the firstDocument window should be deactivated.

Externalize

This method is only of interest to Container Application developers.

Function Prototype

```
void Externalize(
        XMPDraft* draft);
```

Protocol

'Ternalization

Protection

Public. Called by the OpenDoc shell when saving a draft.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Saves all windows belonging to the specified draft in the draft.

Inputs

draft	An open OpenDoc draft

Outputs

None.

Exceptions Signalled

kXMPErrInvalidDraft	The specified draft is not valid.

Pre conditions

A valid open draft object.

Post conditions

The draft contains the IDs of the windows that were saved in the draft.

GetActiveWindow

Function Prototype

```
XMPWindow* GetActiveWindow();
```

Protocol

Window Access

Protection

Public. Used by the dispatcher.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the front-most window which is not a "floating" window.

Inputs

None.

Outputs

<return>	The active window

Exceptions Signalled

None.

Pre conditions

Avalid initialized instance.

Post conditions

No effect on this instance.

GetFrontFloatingWindow

This method is Macintosh specific.

Function Prototype

```
XMPWindow*    GetFrontFloatingWindow();
```

Protocol

Layers

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the front floating window. If there is no such window, kXMPNULL is returned.

Inputs

None.

Outputs

<return>	Front floating window.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

No effect on this instance.

GetFrontRootWindow

This method is Macintosh specific.

Function Prototype

```
XMPWindow*    GetFrontRootWindow();
```

Protocol

Layers

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the front non-floating root document window. If there is no such window, kXMPNULL is returned.

Inputs

None.

Outputs

<return>	Front non-floating root document window

Exceptions Signalled

None.

Pre conditions

A valid initialized instance

Post conditions

No effect on this instance.

GetFrontWindow

This method is Macintosh specific.

Function Prototype

```
XMPWindow*    GetFrontWindow();
```

Protocol

Layers

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the front non-floating document window. If there is no such window, kXMPNULL is returned.

Inputs

None.

Outputs

<return>	First document window.

Exceptions Signalled

None.

Pre conditions

None.

Post conditions

None.

GetRootWindowCount

Function Prototype

```
XMPUShort GetRootWindowCount(
        XMPDraft* draft);
```

Protocol

Counting Windows

Protection

Public. Used by the document shell when closing a window, to determine when to close the draft.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the number of root windows belonging to the specified draft.

Inputs

None.

Outputs

<return>	The number of root windows belonging to the specified draft.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

No effect on this instance.

GetTotalRootWindowCount

Function Prototype

XMPUShort GetTotalRootWindowCount();

Protocol

Counting Windows

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the number of root windows of all drafts.

Inputs

None.

Outputs

<return>	The number of root windows of all drafts

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

No effect on this instance.

GetWindow

Function Prototype

XMPWindow* GetWindow(
XMPID id);

Protocol

Window Access

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the window object with the given ID, or kXMPNULL if the window has been destroyed or never existed. Note that these are not persistent IDs, but are valid for a session.

Inputs

id	A window ID, obtained from a window using XMPWindow::GetID.

Outputs

<return>	The window matching the specified ID, or kXMPNULL.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

No effect on this instance.

GetWindowCount

Function Prototype

XMPULong GetWindowCount();

Protocol

Counting Windows

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the number of windows in the window state (regardless of draft)

Inputs

None.

Outputs

<return>	The number of OpenDoc windows in the WindowState.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

No effect on this instance.

GetXMPWindow

Function Prototype

```
XMPWindow* GetXMPWindow(
        XMPPlatformWindow window);
```

Protocol

Window Access

Protection

Public. Called mostly by the dispatcher.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns the XMPWindow object corresponding to the specified platform-specific window object, or kXMPNULL.

Inputs

window	A platform-specific window object

Outputs

<return>	The corresponding OpenDoc window, or kXMPNULL if the specified window is not an OpenDoc window

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

No effect on this instance.

Internalize

This method is only of interest to Container Application developers.

Function Prototype

```
void Internalize(
        XMPDraft* draft);
```

Protocol

'Ternalization

Protection

Public. Called by the OpenDoc shell when a draft is opened.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Adds to the window state all windows persistently stored in the specified draft.

Inputs

draft	An open OpenDoc draft.

Outputs

None.

Exceptions Signalled

kXMPErrInvalidDraft	The specified draft is not valid.
kXMPErrOutOfMemory	Out of memory

Pre conditions

A valid open draft object.

Post conditions

The window state contains all the windows that were saved in the draft.

IsXMPWindow

Function Prototype

```
XMPBoolean IsXMPWindow(
        XMPPlatformWindow window);
```

Protocol

Window Access

Protection

Public. Called mostly by the dispatcher.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Returns true if the specified platform-specific window is an OpenDoc window.

Inputs

window	A platform-specific window object

Outputs

<return>	kXMPTrue, if the window is an OpenDoc window

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

No effect on this instance.

OpenWindows

This method is only of interest to Container Application developers.

Function Prototype

```
void OpenWindows(
        XMPDraft* draft);
```

Protocol

'Ternalization

Protection

Public. Called by the OpenDoc shell.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Opens all windows stored in the specified draft, and shows those that were visible when saved.

Inputs

draft	An open OpenDoc draft.

Outputs

None.

Exceptions Signalled

kXMPErrInvalidDraft	The specified draft is not valid.
kXMPErrOutOfMemory	Out of memory

Pre conditions

A valid open draft object. The WindowState has already Internalized the windows of the specified draft.

Post conditions

The windows of the specified draft are open.

Purge

This method is only of interest to Container Application developers.

Function Prototype

```
XMPSize Purge(
        XMPSize size);
```

Protocol

Low Memory

Protection

Public. Called by OpenDoc.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Frees up some memory, by deleting cached values which can be recomputed or restored from persistent storage.

Inputs

size	The amount of memory requested.

Outputs

<return>	The amount of memory freed.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance.

Post conditions

Some memory has been freed up.

SetBaseMenuBar

This method is only of interest to Container Application developers.

Function Prototype

```
void SetBaseMenuBar(
        XMPMenuBar* menuBar);
```

Protocol

Menu Bar

Protection

Public. Called by the OpenDoc shell application

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Stores the base menu bar created by the shell. Parts call CopyBaseMenuBar to create a menu bar to which they add their own menus.

Inputs

menuBar	Reference to base menu bar object

Outputs

None.

Exceptions Signalled

None.

Pre conditions

A valid initialized instance

Post conditions

Field is set.

SetDefaultWindowTitles

This method is only of interest to Container Application developers.

Function Prototype

```
void SetDefaultWindowTitles(
        XMPDraft* draft);
```

Protocol

'Ternalization

Protection

Public.

Override policy

Derived class **can** override.

Derived class **can** call base class behavior, during derived class behavior.

Basic operation

Synchronizes window titles with the file name.

Inputs

draft	An open OpenDoc draft

Outputs

None.

Exceptions Signalled

kXMPErrInvalidDraft	The specified draft is not valid.

Pre conditions

A valid open draft object.

Post conditions

The window titles are synchronized with the file name.

Global Functions

Basic Documentation

This section describes the global functions available in OpenDoc.

Functions

OpenXMPSession

This function is only called by container applications.

Function Prototype

XMPSession* OpenXMPSession ();

Protocol

System

Protection

Public.

Basic operation

Creates new XMPSession object; returns reference.

Inputs

None

Outputs

<return>	reference to the XMPSession object created by this call.

Exceptions Signalled

exception	OpenDoc could not be initialized.

Pre conditions

This should only be called once per process.

Post conditions

The OpenDoc environment will be initialized.

ShowPartFrameInfo

Function Prototype

void ShowPartFrameInfo(XMPFrame* frame);

Protocol

Part

Protection

Public.

Basic operation

Show the part/frame info dialog and permits users to change info on them.

Inputs

frame	the selected frame

Outputs

none

Exceptions Signalled

none

Pre conditions

'frame' is an XMPFrame object selected in the content of the caller.

Post conditions

Info settings have been updated to reflect what the user chose in the dialog.

CONSTANTS

XMPConstants

General

kXMPFalse	An XMPBoolean constant equal to 0 .
kXMPTrue	An XMPBoolean constant equal to 1.
kXMPNULL	The value NULL.
kXMPsmSystemScript	A Designator for the system script.
kXMPsmScriptLang	The current script language.
kXMPTinyIconSize	The size for tiny icons equal to 12x12 pixels.
kXMPSmallIconSize	The size for small icons equal to 16x16 pixels.
kXMPLargeIconSize	The size for large icons equal to 32x32 pixels.
kXMPThumbnailSize	The size for thumb nails equal to 64x64 pixels.

Name Spaces

kNameMappings	The resource type for NMAP resources.
kXMPIsMacOSTypeID	The Mac OSType identifier in NMAP resource.
kXMPIsINTLTextID	The International Text type identifier in NMAP resource.
kXMPIsAnISOStringID	The ISO String identifier in NMAP resource.
kXMPIsAnISOStringListID	The ISO String list identifier in NMAP resource.
kXMPNSTypeISOStr	An indicator of a name space with keys that are of type ISO String.
kXMPNSTypeOSType	An indicator of a name space with keys that are of type OSType.
kXMPNSTypeSLong	An indicator of a name space with keys that are of type signed long.

Category Constants

Editor categories.

kXMPCategoryText	="OpenDoc:Category:Text". The category of a Text part editor.
kXMPCategoryGraphics	= "OpenDoc:Category:Graphics". The category of a graphics part editor.

kXMPCategoryTime	="OpenDoc:Category:Time". The category of a clock part editor.
kXMPCategoryControl	="OpenDoc:Category:Control". The category of a control part.
kXMPCategoryVideo	="OpenDoc:Category:Video". The category of a movie part.
kXMPCategorySound	="OpenDoc:Category:Sound". The category of a sound part.
kXMPCategoryPartsBin	="OpenDoc:Category:PartsBin". The category of the PartsBin part.

Parts

kNewGroup	Used in the XMPPart::CreateEmbeddedFrame() call. Pass as the frameGroupID parameter to ask the containing part to put the created frame in a new frame group.

UI Subsystem - Foci

Constants for the standard foci managed by the Arbitrator. These constants are ISO strings which can be tokenized using XMPSession::Tokenize().

kXMPScrollingFocus	Of type XMPFocusType. PageUp etc. keys are sent to the owner of the scrolling focus, which may be different from the keystroke focus.
kXMPKeyFocus	Of type XMPFocusType. The frame which owns this focus is the recipient of keyboard events.
kXMPMenuFocus	Of type XMPFocusType. The frame which owns this focus receives menu events.
kXMPSelectionFocus	Of type XMPFocusType. The active border is drawn around all facets of the frame with the selection focus. Modified mouse clicks are sent to this frame.
kXMPModalFocus	Of type XMPFocusType. A frame in a modal state should own the modal focus.
kXMPScrollingFocus	Of type XMPFocusType. PageUp etc. keys are sent to the owner of the scrolling focus, which may be different from the keystroke focus.

UI Subsystem - Menu IDs

The IDs of the menus in the base menu bar installed by the shell.

kXMPAppleMenuID	The ID of the Apple Menu.

kXMPFileMenuID	The ID of the Document menu.
kXMPEditMenuID	The ID of the Edit Menu.
kXMPOCEFileMenuID	The ID of the Document menu on System 7 Pro [This should go away, and be the same as kXMPFileMenuID]

UI Subsystem - Standard Mac Events

These constants are simply redefinitions of the standard Macintosh event codes.

kXMPEvtNull	A null event. Represents idle time to parts.
kXMPEvtMouseDown	A mouse down event. See Inside Mac.
kXMPEvtMouseUp	A mouse up event. See Inside Mac.
kXMPEvtKeyDown	A key down event. See Inside Mac.
kXMPEvtKeyUp	A key up event. See Inside Mac.
kXMPEvtAutoKey	An autokey event. See Inside Mac.
kXMPEvtUpdate	An update event. See Inside Mac.
kXMPEvtDisk	A disk event. See Inside Mac.
kXMPEvtActivate	An activate event. See Inside Mac.
kXMPEvtOS	An OS Event (suspend/resume and mouseMoved). See Inside mac.

UI Subsystem - Additional OpenDoc Events

OpenDoc defines some additional events, over and above the standard Macintosh events.

kXMPEvtMenu	An adaptation of a mouse down in the menu bar, or the equivalent Command-Key combination. Message field of event record contains menu in high word and item in low word. Other fields of event record are as for mouse down or key down.
kXMPEvtMouseDownEmbedded	A mouse down in an embedded facet. Sent to containing facet. Message field of event record contains the embedded facet. Other fields of event record are as for mouse down.
kXMPEvtMouseUpEmbedded	A mouse up event in an embedded facet. Sent to containing facet. Message field of event record contains the embedded facet. Other fields of event record are as for mouse up.

kXMPEvtMouseDownBorder	Mouse down in the active border (around facets of the frame with the selection focus). Message field of event record contains the embedded facet. Other fields of event record are as for mouse down.
kXMPEvtMouseUpBorder	Mouse up in the active border (around facets of the frame with the selection focus). Message field of event record contains the embedded facet. Other fields of event record are as for mouse up.
kXMPEvtWindow	Adaptation of mouse event in the title bar of a window. Message field of event record contains part code returned by FindWindow(). Other fields of event record are as for mouse down.

UI Subsystem - Part Codes

These map exactly to the part codes as returned by FindWindow(). See Inside Mac.

kXMPMDInDesk

kXMPMDInMenuBar

kXMPMDInSysWindow

kXMPMDInContent

kXMPMDInDrag

kXMPMDInGrow

kXMPMDInGoAway

kXMPMDInZoomIn

kXMPMDInZoomOut

UI Subsystem - Command IDs

Position-independent command IDs for the menus in the base menu bar, as provided by the OpenDoc shell.

kXMPCommandAbout	The command ID for the About… menu item in the Apple menu.
kXMPCommandNew	The command ID for the New… item in the Document menu.
kXMPCommandOpen	The command ID for the Open item in the Document menu. Applies to a selected frame.

kXMPCommandOpenDocument The command ID for the Open Document… item in the Document menu, which brings up the Standard File dialog.

kXMPCommandInsert The command ID for the Insert… item in the Document menu.

kXMPCommandClose The command ID for the Close item in the Document menu.

kXMPCommandSave The command ID for the Save menu item in the Document menu.

kXMPCommandSaveACopy The command ID for the Save a Copy… item in the Document menu.

kXMPCommandRevert The command ID for the Revert… item in the Document menu.

kXMPCommandDraft The command ID for the Drafts… item in the Document menu.

kXMPCommandPageSetup The command ID for the Page Setup… item in the Document menu.

kXMPCommandPrint The command ID for the Print… item in the Document menu.

kXMPCommandShowPartsBin The command ID for the Show Parts Bin item in the Document menu.

kXMPCommandAddDeleteAOCEMailer The command ID for theAdd Mailer item in the Document menu, on a system running AOCE.

kXMPCommandExpandAOCEMailer The command ID for the Expand Mailer item in the Document menu, on a system running AOCE.

kXMPCommandContractAOCEMailer The command ID for the Contract Mailer item in the Document menu, on a system running AOCE.

kXMPCommandSendMail The command ID for Send… item in the Document menu, on a system running AOCE.

kXMPCommandForwardMail The command ID for Forward item in the Document menu, on a system running AOCE.

kXMPCommandReplyToMail The command ID for Reply item in the Document menu, on a system running AOCE.

kXMPCommandReplyToAllMail	The command ID for Reply to all item in the Document menu, on a system running AOCE.
kXMPCommandUndo	The command ID for Undo item in the Edit menu.
kXMPCommandRedo	The command ID for Redo item in the Edit menu.
kXMPCommandCut	The command ID for Cut item in the Edit menu.
kXMPCommandCopy	The command ID for Copy item in the Edit menu.
kXMPCommandPaste	The command ID for Paste item in the Edit menu.
kXMPCommandPasteAs	The command ID for the Paste As… item in the Edit menu.
kXMPCommandClear	The command ID for Clear item in the Edit menu.
kXMPCommandSelectAll	The command ID for the Select All item in the Edit menu.
kXMPCommandGetPartInfo	The command ID for the Get Part Info item in the Edit menu.
kXMPCommandPreferences	The command ID for the Preferences… item in the Edit menu.
kXMPCommandViewAsWin	The command ID for the View As Window item in the Edit menu.

UI Subsystem-Persistent Properties

Constants for the persistent properties of UI system object, namely XMPWindows.

kXMPPropWindowList	= "OpenDoc:Property:WindowList". A property of the draft. A list of storage unit IDs for windows.
kXMPPropWindow	= "OpenDoc:Property:Window". Not used?
kXMPPropWindowRect	= "OpenDoc:Property:WindowRect". Value type kXMPWindowRect. The bounding rectangle of a window.
kXMPPropWindowTitle	= "OpenDoc:Property:WindowTitle". Value type kXMPAppleTEXT. Stores the title of the window.

kXMPPropWindowProcID = "OpenDoc:Property:WindowProcID". Stores the ProcID of the window. Value type kXMPSShort.

kXMPPropWindowIsVisible = "OpenDoc:Property:WindowVisible". Stores whether the window is visible or not. Value type kXMPBoolean.

kXMPPropWindowHasCloseBox = "OpenDoc:Property:WindowHasCloseBox". Stores whether the window has a close box or not. Value type kXMPBoolean.

kXMPPropWindowHasZoomBox = "OpenDoc:Property:WindowHasZoomBox". Stores whether the window has a zoom box or not. Value type kXMPBoolean.

kXMPPropWindowIsResizable = "OpenDoc:Property:WindowIsResizable". Stores whether the window has a resize box or not. Value type kXMPBoolean.

kXMPPropWindowIsRootWindow = "OpenDoc:Property:WindowIsRootWindow". Stores whether the window is visible or not. Value type kXMPBoolean.

kXMPPropWindowIsFloating = "OpenDoc:Property:WindowIsFloating". Stores whether the window is floating or not. Value type kXMPBoolean.

kXMPPropWindowHasMailer = "OpenDoc:Property:WindowHasMailer". Stores whether the window has a mailer attached or not. Value type kXMPBoolean.

kXMPPropWindowIsMailerAware = "OpenDoc:Property:WindowIsMailerAware". Stores whether the window can have a mailer attached or not. Value type kXMPBoolean.

kXMPPropWindowRefCon = "OpenDoc:Property:WindowRefCon". Stores the value of the refCon field of the window record. Value type kXMPSLong.

kXMPPropRootFrame = "OpenDoc:Property:RootFrame". Storesthe ID of the root frame. Value type kXMPID.

kXMPPropSourceFrame = "OpenDoc:Property:SourceFrame". Storesthe ID of the source frame. Value type kXMPID.

kXMPPropShouldShowLinks = "OpenDoc:Property:ShouldShowLinks". Stores whether or not parts in this window should display link borders.

UI Subsystem - Persistent Value Types

The value types of the persistent properties of XMPWindow. (Those not listed here are standard ones like kXMPFrame).

kXMPWindowRect	A value type for the window bounding rectangle. Equal to kXMPRect.

UI subsytem - Undo

kXMPDone	Value passed to Part::DisposeActionState() if the action was on the Undo stack
kXMPUndone	Value passed to Part::DisposeActionState() if the action was on the Redo stack
kXMPRedone	Value passed to Part::DisposeActionState() if the action was on the Undo stack

Messaging subsystem

Constants of the Messaging subsystem

kXMPAppShell	Used to represent the shell application, when calling methods of the messaging subsystem which normally take an XMPPart* as an argument.
kOSLContextSize	Defined to be = sizeof(XMPOSLContext*);
cPart	= 'part'. An AppleEvent Object Model property.

Messaging - Properties for Part Info Dialog

Apple Event Object Model properties corresponding to the elements of the Part Info dialog.

kXMPPCategory	= 'cate'. The category of a selected part.
kXMPPSize	= 'size'. The size of a selected part.
kXMPPCreationDate	= 'crdt'. The creation date of a selected part.
kXMPPModDate	= 'modt'. The modification date of a selected part.
kXMPPAuthor	= 'auth'. The author of a selected part.
kXMPPIcon	= 'icon'. The icon of a selected part.
kXMPPName	= 'pnam'. The name of a selected part.
kXMPPKind	= 'pknd'. The kind of a selected part.
kXMPPEditor	= 'edtr'. The editor of a selected part.

kXMPPComments	= 'cmnt'. A comment about a selected part.
kXMPPView	= 'view'. The type of view of a selected part (Frame, Icon etc).
kXMPPIsFrozen	= 'frzn'. Whether a selected part is "frozen" or not.
kXMPPIsStationery	= pIsStationeryPad. Is the stationery flag of the selected part set?

Data Interchange - Constants

These constants are used during data interchange.

kXMPUnknownChange	An XMPChangeID value guaranteed to be different than any actual change identification. Can be used by parts when the change identifcation associated with shared content is unknown.
kXMPCloneCopy	An XMPCloneKind value specifying copy semantics into an intermediate draft such as the clipboard or a drag-and-drop container.
kXMPCloneCut	An XMPCloneKind value specifying cut semantics into an intermediate draft.
kXMPClonePaste	An XMPCloneKind value specifying paste semantics from an intermediate draft.
kXMPCloneDuplicate	An XMPCloneKind value specifying duplicate semantics directly from the source draft to into the destination draft.
kXMPInLinkDestination	The XMPLinkStatus value signifying a frame embedded in the destination of a link; the content of this frame is thus supplied by a link.
kXMPInLinkSource	The XMPLinkStatus value signifying a frame that is embedded in one or more sources of a link, but not in content that is the destination of a link.
kXMPNotInLink	The XMPLinkStatus value signifying a frame that is not embedded in any linked content, source or destination.
kXMPDropSucceed	An XMPDropResult value signifying a successful synchronous drop.
kXMPDropFail	An XMPDropResult value signifying an unsuccessful synchronous drop.

kXMPDropUnfinished	An XMPDropResult value signifying an asynchronous drag was started.

Storage System

These constants are used for the Storage System.

kDPNone	No access privilege on a Draft.
kDPTransient	No access privilege on a Draft.
kDPReadOnly	Read-only access privilege on a Draft.
kDPSharedWrite	Shared access on a Draft.
kDPExclusive	Exclusive read-write access on a Draft.
kInvisibleBlocks	Only Blocks with no references to them should be purged.
kAllBlocks	All blocks should be purged.
kVisibleBlocks	Only blocks with references to them should be purged.
kXMPDefaultDocumentID	Default Document ID.
kXMPPosUndefined	Undefined Property or Value context.
kXMPPosSame	Maintain the same Property or Value context.
kXMPPosAll	Focus to all Properties or all Values.
kXMPPosFirstSib	Set the context to the first Property of the Storage Unit or the first Value of the Storage Unit.
kXMPPosLastSib	Set the context to the last Property of the Storage Unit or the last Value of the Storage Unit.
kXMPPosNextSib	Set the context to the next Property of the Storage Unit or the first next of the Storage Unit with respect to the current context.
kXMPPosPrevSib	Set the context to the previous Property of the Storage Unit or the first previous of the Storage Unit with respect to the current context.
kXMPPosMWrap	Wraps iteration of Properties or Values.
kXMPIDAll	Set the context to all Values.
kXMPIndexAll	Set the context to all Values.

kXMPTypeAll	Set the context to all Values.

Data Interchange - Persistent Properties

These constants identify persistent properties of Data Interchange objects.

kXMPPropLink	A property containing the persistent identity of an XMPLink object.
kXMPPropLinkSource	A property containing the persistent identity of an XMPLinkSource object.
kXMPPropLinkSpec	A property containing the representation of an XMPLinkSpec object.
kXMPPropEditionAlias	A Macintosh-specific property containing the alias of an edition file used in implementing cross-document links.
kXMPPropLinkSection	A Macintosh-specific property containing the section record of a cross-document link.
kXMPPropLinkContentSU	A property containing the persistent identity of an XMPStorageUnit serving to store the contents of a link.
kXMPPropAutoExport	A property containing an XMPBoolean value for the auto export status of an XMPLinkSource object.
kXMPPropChangeID	A property containing the persistent form of an XMPChangeID value.
kXMPPropOriginalID	A property of XMPLink and XMPLinkSource objects in intermediate containers, such as the clipboard, containing the non-persistent object ID of the original object in the original draft.
kXMPPropOriginalDraft	A draft property of intermediate drafts, such as the clipboard draft, containing the non-persistent object ID of the original draft.
kXMPPropOriginalCloneKind	A draft property of intermediate drafts, such as the clipboard draft, containing the persistent representation of an XMPCloneKind value denoting the operation (kXMPCloneCut or kXMPCloneCopy) that deposited content in the draft.
kXMPPropEditionID	A document property containing the last edition file ID used by any draft of the document.

kXMPPropSectionID	A document property containing the last section ID used by any draft of the document.

Data Interchange - Persistent Value Types

These constants identify persistent value types used in Data Interchange.

kXMPLinkSpec	The XMPValueType for the persistent form of an XMPLinkSpec object.
kXMPChangeID	The XMPValueType for the persistent form of an XMPChangeID value.
kXMPLinkContent	The XMPType value used to identify a link content storage unit; used as the value of the kXMPPropStorageUnitType property.
kXMPLinkSource	An XMPType value. The value of the kXMPPropStorageUnitType property of storage units containing the persistent form of an XMPLinkSource object.
kXMPLink	An XMPType value. The value of the kXMPPropStorageUnitType property of storage units containing the persistent form of an XMPLink object.
kXMPAppleTEXT	The XMPValueType for values in Apple 'TEXT' scrap format.
kXMPApplestyl	The XMPValueType for values in Apple 'styl' scrap format.
kXMPApplePICT	The XMPValueType for values in Apple 'PICT' scrap format.
kXMPApplehfs	The XMPValueType for values in Apple 'hfs ' scrap format.
kXMPApplesnd	The XMPValueType for values in Apple 'snd ' scrap format.
kXMPApplealis	The XMPValueType for values in Apple 'alis' scrap format.
kXMPApplesect	The XMPValueType for values in Apple 'sect' scrap format.

TYPES

XMPTypes

General

These scalar types are used widely in OpenDoc.

XMPISOStr	A string type composed of 7 bit ascii terminated by a zero byte. Since the first zero byte terminates the string, NULLs may not be embedded.
XMPType	The type used generically within OpenDoc to represent storage unit types, focus types, and value types. An XMPType is an XMPISOStr.
XMPValueType	Identifies data value types. An XMPType.
XMPFixed	XMPFixed represents non-integer numbers in the range [-32768, 32768) as 32-bit values whose high 16 bits (including a sign bit) represent the integer part, and whose low 16 bits represent a fractional part. In effect, the “binary point” is in the middle of the number. An integer can be converted to XMPFixed by shifting it left 16 bits. An XMPFixed can be rounded to an integer by adding 0x8000 (0.5) and shifting it right 16 bits. You can convert between XMPFixed and floating-point by multiplying or dividing by 65536.0. XMPFixeds can be added or subtracted from each other as though they were regular integers. Multplication or division are trickier. XMPFixed is identical to the Macintosh "Fixed" type, and the Macintosh has several Toolbox routines, such as FixMul and FixDiv, for doing fixed-point arithmetic.
XMPFract	XMPFract is, like XMPFixed, a 32-bit fixed-point number used to represent non-integer values. However, it has only two bits to the left of the “binary point”, instead of 16. It can only represent values in the range [-2,2), but it has much higher precision than an XMPFixed. Use 30-bit shifts to convert between XMPFracts and integers, or a scaling factor of 1073741824.0 to convert between XMPFracts and floating-point numbers. (XMPFract is identical to the Macintosh "Fract" type, and the Toolbox has routines such as FracMul and FracDiv for doing arithmetic on Fract values.)
XMPBoolean	A boolean value.
XMPHeap	A memory heap pointer.

XMPSByte	A signed 8-bit value.
XMPUByte	An unsigned 8-bit value.
XMPUShort	An unsigned 16-bit value.
XMPSShort	A signed 16-bit value.
XMPULong	An unsigned 32-bit value.
XMPSLong	A signed 32-bit value.
XMPPtr	An XMP pointer. A void*.
XMPFileSpec	A struct specifying a file on disk.
XMPScriptCode	An international script code.
XMPFileRefNum	A file reference number. An XMPSShort .
XMPEditor	A type identifying a specific part editor.
XMPContainerSuite	A type idetifying a specific container suite.

Data Interchange

These scalar types are used by classes providing data interchange in OpenDoc.

XMPLinkKey	XMPLinkKey is used to provide thread-safe access to link content. It is required by every method of class XMPLink and class XMPLinkSource that return or modify the link's content storage unit. Values of this type are not inspectable. Values of this type are created by XMPLink::Lock and XMPLinkSource::Lock.
XMPChangeID	Values of this type are associated with clipboard content and with linked content. Two XMPChangeID values associated with different versions of the same content may be tested for equality, but any other use of these values is meaningless. For example, the change identification associated with the clipboard may be tested for equality with a previous clipboard change id to see if the contents have changed, but it is invalid to test if the current change id is greater than a previous value.
XMPC lipboardKey	XMPClipboardKey is used to provide thread-safe access to the clipboard. It is required by every method of class XMPClipboard that return or modify the clipboard's content storage unit. Values of this type are not inspectable. Values of this type are created by XMPClipboard::Lock.

XMPCloneKind	An enumerated type used to indicate the desired semantics of a clone operation. Values of this type are passed to XMPDraft::BeginClone. The values are: kXMPCloneCopy, specifying copy semantics into an intermediate draft such as the clipboard or a drag-and-drop container, kXMPCloneCut, specifying cut semantics into an intermediate draft, kXMPClonePaste, specifying paste semantics from an intermediate draft, and kXMPCloneDuplicate, specifying duplicate semantics directly from the source draft to into the destination draft.
XMPPlatformType	A cover for the platform-specific type used to identify data formats for data interchange.
XMPLinkStatus	An enumerated type specifying the link context of a frame. This type has three values. The value kXMPInLinkDestination describes a frame that is embedded in content that is the destination of a link. The value kXMPInLinkSource describes a frame that is embedded in content that is the source of a link (perhaps in several source links that overlap). The value kXMPNotInLink indicates a frame that is not embedded in linked content.
XMPDropResult	An enumerated type indicating the result of a drag. The three values are kXMPDropSucceed, meaning a successful synchronous drop; kXMPDropFail, meaning an unsuccessful synchronous drop, and kXMPDropUnfinished meaning that an asynchronous drag was started.

Storage System

XMPDocumentID	ID used to retrieve a Document.
XMPDocumentName	Name associated with a Document
XMPDraftID	Identifier for a Draft.
XMPDraftName	Name associated with a Draft.

Imaging

The majority of these data types are used to represent geometric structures in OpenDoc, such as distances, positions, bounding boxes and shapes. (Shapes at runtime are usually represented by XMPShape objects with an opaque data format, but they can be accessed and stored in one of the forms listed here.) Several of these types are used to describe non-geometric information about XMPFrames and XMPFacets.

XMPCoordinate	XMPCoordinate represents spatial coordinates in documents and windows. It's an XMPFixed, which usually implies 16 integer and 16 fractional bits, but internally you can partition the bits up however you please provided you shift the values appropriately when talking to the outside world. (If you are using a graphics system, such as QuickDraw GX, that handles arbitrary transformations, you can do this automatically by assigning a scaling factor to your internal transform.)
XMPPoint	An XMPPoint represents a spatial point in a window or document. In two-dimensional imaging models (all that exist so far for OpenDoc) it is a pair of XMPCoordinates. Macintosh developers accustomed to QuickDraw should take note that the x coordinate appears first. Also note that the coordinates are, of course, fixed-point numbers. On the other hand, an XMPPoint is identical to a QuickDraw GX gxPoint structure.
XMPRect	XMPRect represents a rectangle whose sides are aligned with the axes of the current coordinate system. It is represented as four XMPCoordinates giving the left, top, right and bottom extent of the rectangle (in that order.) By convention a rectangle does not include its bottom or right edge; this makes it easier to have adjacent yet non-overlapping rectangles. Macintosh developers accustomed to QuickDraw should note that the left coord comes before the top, and likewise the right comes before the bottom. On the other hand, an XMPRect is identical to a QuickDraw GX gxRect structure.
XMPPolygon	An XMPPolygon represents a two-dimensional shape made out of straight edges. It is the platform-independent interchange format for all shapes, including frame shapes. An XMPPolygon consists of one or more contours, each of which is a closed loop of XMPPoints connected by straight edges. Polygons with multiple contours may be composed of disjoint pieces, or may have interior holes. The convention is that clockwise contours [in a left-handed coordinate system as used on the Macintosh] fill positive space, while counter-clockwise contours represent holes. XMPPolygon structures are variable size and are always allocated on a heap. They start with a 32-bit count of the number of contours. The

	contours then follow: each starts with a 32-bit point count and then the points in order. The last point of a contour is of course connected to the first.
XMPMatrix	A 3x3 matrix of fixed-point (XMPFixed) numbers used to represent a transformation. This is the platform-independent representation of a coordinate transformation. Not all graphics systems support these kind of transformations; some (like QuickDraw) only support simple offsets. See the documentation for the XMPTransform class for more details.
XMPGraphicsSystem	An enumeration that lists possible graphics systems. Every graphics system supported by OpenDoc, on any platform, should have a unique XMPGraphicsSystem ID. These values are used in XMPTransforms and XMPShapes to tag graphics-system dependent transform or shape data. For instance, on the Macintosh you can feed a Region to an XMPShape provided you tag it with kXMPQuickDraw.
XMPPlatformShape	A wrapper for a pointer to graphics-system dependent shape data. The data format is unspecified, and it must be tagged with an XMPGraphicsSystem value to identify which graphics system it belongs to. A (XMPPlatformShape, XMPGraphicsSystem) pair is always sufficient to identify the exact type of the data.
XMPPlatformTransform	A wrapper for graphics-system dependent transformation data. The data format is unspecified, and it must be tagged with an XMPGraphicsSystem value to identify which graphics system it belongs to. A (XMPPlatformTransform, XMPGraphicsSystem) pair is always sufficient to identify the exact type of the data.
XMPGeometryMode	XMPGeometryMode is the geometry mode of an XMPShape, which tells whether the shape's geometric info (its polygonal representation) will be needed in the future. It has three possible values: • kXMPPreserveGeometry --the default-- means that the shape will preserve its geometric information as long as possible until it is lost by combination with a non-geometric shape. • kXMPLoseGeometry means that the geometry is not needed and can be discarded to optimize speed. A facet's clipShape will generally have this mode. •

	kXMPNeedsGeometry means that the geometry is required. Rather than discard geometry, the shape will throw a kXMPErrNoShapeGeometry exception if combined with a non-geometric shape. A facet's frameShape and usedShape will have this mode since they are stored persistently in polygonal form.
XMPHighlight	An enumeration describing the possible highlight states of an XMPFacet. Possible values are: kXMPNoHighlight, the facet is not highlited; kXMPFullHighlight, the facet is highlighted in foreground style; kXMPDimHighlight, the facet is highlighted in background style.
XMPFramePosition	An enumeration describing the position of an XMPFrame relative to a sibling frame. Possible values are: kXMPFrameBehind, kXMPFrameInFront.
XMPInfoType	The type of the partInfo data stored in an XMPFrame or XMPFacet. This is an opaque type. The part editor should cast the partInfo data to and from its own representation to use the data.

UI Subsystem

Types used in the interfaces of the UI subsystem.

XMPFocusType	An ISO string. Can be tokenized using XMPSession::Tokenize(). Arbitrator methods used tokenized form.
XMPPlatformWindow	A platform-specific structure representing a window. On the Macintosh, a WindowPtr.
XMPEventType	An unsigned short. Used in the "what" field of a Macintosh event record.
XMPEventData	A platform-specific structure representing an event. On the Macintosh, a pointer to a Macintosh event record.
XMPIdleFrequency	An unsigned long. The number of ticks (60ths of a second) between idles.
XMPPlatformMenu	A platform-specific structure representing a menu. On the Macintosh, a MenuHandle.
XMPMenuID	The ID of a menu. On the Macintosh, a signed short.

XMPMenuItemID	The ID of a menu item. On the Macintosh, a signed short.
XMPCommandID	A signed long. Used for Command IDs associated with menu/item combinations.
XMPPlatformMenuBar	A platform-specific structure representing a menu bar. A Handle, on the Macintosh.
XMPDoneState	Type of value passed to Part::DisposeActionState(). Possible values: kXMPDone or kXMPRedone if the action was on the Undo stack. kXMPUndone if the action was on the Redo stack.
XMPActionData	Action data for Undo/Redo. Defined to be an XMPPtr.

Messaging

Types used by clients of the Messaging subsystem. Most of the function pointer types defined in SemtIntf.h
correspond to AppleEvent function pointer types, with the addition of a Part* parameter.

XMPIdleProcPtr	See AppleEvent documentation.
XMPEventFilterProcPtr	See AppleEvent documentation.
XMPObjectSpec	Defined as an AppleEvent descriptor, or AEDesc.
XMPOSLContext	Equivalent to an AppleEvent Object Support Library OSLContext.
XMPEventHandlerProcPtr	See AppleEvent documentation.
XMPCoercionHandler	See AppleEvent documentation.
XMPDescCoercionHandler	See AppleEvent documentation.
XMPPtrCoercionHandler	See AppleEvent documentation.
XMPAccessorProcPtr	See AppleEvent documentation.
XMPCompareProcPtr	See AppleEvent documentation.
XMPCountProcPtr	See AppleEvent documentation.
XMPDisposeTokenProcPtr	See AppleEvent documentation.
XMPGetErrDescProcPtr	See AppleEvent documentation.
XMPGetMarkTokenProcPtr	See AppleEvent documentation.

XMPMarkProcPtr	See AppleEvent documentation.
XMPAdjustMarksProcPtr	See AppleEvent documentation.
XMPSpecialHandlerPtr	See AppleEvent documentation.
XMPTokenInquiryProcPtr	Pointer to a function that returns an XMPPart* if the given token refers to a part, or kXMPNULL otherwise.