**CI Lab**

# OpenDoc Class Reference

From Component Integration Laboratories, Inc.

**Seed Note**

**Preliminary**

6/20/94
Developer Press
© Apple Computer, Inc. 1994

Apple Computer, Inc.
© 1994 Apple Computer, Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Computer, Inc.

The Apple logo is a trademark of Apple Computer, Inc.
Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this book. Apple retains all intellectual property rights associated with the technology described in this book. This book is intended to assist application developers to develop applications only for Apple Macintosh computers.

Every effort has been made to ensure that the information in this manual is accurate. Apple is not responsible for typographical errors.

Apple Computer, Inc.
20525 Mariani Avenue
Cupertino, CA 95014
408-996-1010

Apple, the Apple logo, and Macintosh are trademarks of Apple Computer, Inc., registered in the United States and other countries.

OpenDoc, and QuickTime are trademarks of Apple Computer, Inc.

Adobe Illustrator and Adobe Photoshop are trademarks of Adobe Systems Incorporated, which may be registered in certain jurisdictions.

CORBA is a trademark of the Object Management Group, Inc.

FrameMaker is a registered trademark of Frame Technology Corporation.

Helvetica and Palatino are registered trademarks of Linotype Company.

IBM, OS/2, and System Object Model are registered trademarks of International Business Machines, Inc.

ITC Zapf Dingbats is a registered trademark of International Typeface Corporation.

MacWrite is a registered trademark of Claris Corporation.

Microsoft is a registered trademark of Microsoft Corporation.

Simultaneously published in the United States and Canada.

**Even though Apple has reviewed this manual, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS MANUAL, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS MANUAL IS SOLD "AS IS," AND YOU, THE PURCHASER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.**

**IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS MANUAL, even if advised of the possibility of such damages.**

**THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.**

**Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state..**

# Contents


Preface About This Book xxv

What to Read xxv
Chapter Organization xxvi
Conventions Used in This Book xxvii
About Component Integration Laboratories xxvii

Chapter 1 The Class XMPPart 1

XMPPart 4
`XMPPart` 10
`~XMPPart` 11
`AbortRelinquishFocus` 11
`AddDisplayFrame` 12
`AdjustBorderShape` 13
`AdjustMenus` 14
`AttachSourceFrame` 14
`BeginRelinquishFocus` 15
`CanvasChanged` 16
`CanvasUpdated` 17
`ChangeKind` 17
`CloneInto` 18
`CloseDisplayFrame` 19
`CommitRelinquishFocus` 19
`ContainingPartPropertiesChanged` 20
`CreateEmbeddedFramesIterator` 21
`CreateLink` 22
`DisposeActionState` 23
`DragEnter` 24
`DragLeave` 24
`DragWithin` 25
`Draw` 26
`Drop` 27
`DropCompleted` 27




Draft. Preliminary. © Apple Computer, Inc. 6/20/94

EmbeddedFrameChanged 28
EmbeddedFrameSpec 29
ExternalizeKinds 30
FacetAdded 30
FacetRemoved 31
FocusAcquired 31
FocusLost 32
FrameShapeChanged 33
FulfillPromise 33
GeometryChanged 34
GetContainingPartProperties 35
GetPrintResolution 36
HandleEvent 36
HighlightChanged 37
InitPart 38
InitPartFromStorage 38
LinkStatusChanged 38
LinkUpdated 39
MouseEnter 39
MouseLeave 40
MouseWithin 41
Open 41
PresentationChanged 42
ReadActionState 43
ReadPartInfo 43
RedoAction 44
RemoveDisplayFrame 45
RemoveEmbeddedFrame 45
RequestEmbeddedFrame 46
RequestFrameShape 47
RevealFrame 48
RevealLink 49
SequenceChanged 49
UndoAction 50
UsedShapeChanged 51
ViewTypeChanged 51
WriteActionState 52
WritePartInfo 53



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

XMPArbitrator 76
CreateOwnerIterator 79
GetFocusOwner 80
IsFocusExclusive 80
IsFocusRegistered 81
Purge 81
RegisterFocus 82
RelinquishFocus 83
RelinquishFocusSet 84
RequestFocus 85
RequestFocusSet 86
TransferFocus 87
TransferFocusSet 88
UnregisterFocus 89
XMPCanvas 90
XMPCanvas 92
~XMPCanvas 92
GetFacet 93
GetGraphicsSystem 93
GetOwner 94
GetPlatformCanvas 94
GetUpdateShape 95
InitCanvas 95
Invalidate 96
IsDynamic 96
IsOffscreen 97
Purge 97
ResetUpdateShape 98
SetFacet 98
SetOwner 99
Validate 99
XMPClipboard 100
Clear 101
ExportClipboard 102
GetChangeID 103
GetContentStorageUnit 103
Lock 104
SetPlatformClipboard 105



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Unlock 106
XMPContainer 108
XMPContainer 110
~XMPContainer 110
Close 111
Create 111
GetDocument 112
GetID 113
GetName 113
GetStorageSystem 114
IncrementRefCount 114
InitContainer 115
Open 115
Purge 116
Release 117
ReleaseDocument 117
SetName 118
XMPDispatcher 119
AddDispatchModule 120
AddMonitor 121
Dispatch 122
Purge 123
RemoveDispatchModule 123
RemoveMonitor 124
ShouldTerminateEventLoop 124
TerminateEventLoop 125
XMPDocument 126
XMPDocument 128
~XMPDocument 128
CollapseDrafts 129
CreateDraft 130
GetBaseDraft 131
GetContainer 132
GetDocumentProperties 132
GetDraft 133
GetID 135
GetName 136
IncrementRefCount 136
InitDocument 137
Purge 137



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Release 138
ReleaseDraft 138
SaveToAPrevDraft 139
SetBaseDraftFromForeignDraft 140
SetName 141
XMPDraft 142
XMPDraft 145
~XMPDraft 146
AbortClone 146
BeginClone 147
ChangedFromPrev 147
CreateFrame 148
CreateLinkSource 149
CreateLinkSpec 150
CreatePart 151
CreateStorageUnit 151
EndClone 152
Externalize 153
GetDocument 153
GetDraftProperties 154
GetFrame 154
GetID 155
GetLink 155
GetLinkSource 156
GetName 157
GetPart 157
GetPermissions 158
GetStorageUnit 159
IncrementRefCount 159
InitDraft 160
Purge 160
Release 161
ReleaseFrame 161
ReleaseLink 162
ReleasePart 163
ReleaseStorageUnit 163
RemoveChanges 164
RemoveFrame 165
RemoveFromDocument 166
RemoveLink 166



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

RemoveLinkSource 167
RemovePart 168
RemoveStorageUnit 169
SaveToAPrevious 170
SetChangedFromPrev 170
SetName 171
XMPDragAndDrop 172
XMPDragAndDrop 173
~XMPDragAndDrop 174
Clear 174
GetContentStorageUnit 175
InitDragAndDrop 175
Purge 176
StartDrag 176
XMPDragItemIterator 178
First 179
IsNotComplete 179
Next 180
XMPEmbeddedFramesIterator 181
First 182
IsNotComplete 182
Next 182
XMPFacet 184
Method Descriptions 189
XMPFacet 189
~XMPFacet 189
ActiveBorderContainsPoint 189
ChangeActiveShape 190
ChangeClipShape 190
ChangeExternalTransform 191
ChangeGeometry 191
ChangeHighlight 192
ContainsPoint 192
CreateEmbeddedFacet 192
CreateFacetIterator 194
Draw 194
DrawActiveBorder 195
DrawChildren 195
DrawChildrenAlways 195
DrawnIn 196



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

GetActiveShape 196
GetAggregateClipShape 197
GetCanvas 197
GetClipShape 197
GetContainingFacet 198
GetContentTransform 198
GetExternalTransform 199
GetFrame 199
GetFrameTransform 199
GetHighlight 200
GetPartInfo 200
GetWindow 200
GetWindowAggregateClipShape 201
GetWindowContentTransform 201
GetWindowFrameTransform 201
HasCanvas 202
InitFacet 202
Invalidate 203
InvalidateActiveBorder 203
IsSelected 204
MoveBefore 204
MoveBehind 205
RemoveFacet 205
SetCanvas 206
SetPartInfo 207
SetSelected 207
SetWindow 207
Update 208
Validate 208
XMPFacetIterator 209
XMPFacetIterator 210
~XMPFacetIterator 210
First 211
IsNotComplete 211
Next 212
SkipChildren 212
XMPFocusOwnerIterator 213
XMPFocusOwnerIterator 214
~XMPFocusOwnerIterator 214
First 215



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

InitFocusOwnerIterator 215
IsNotComplete 216
Next 216
XMPFocusSet 218
XMPFocusSet 219
~XMPFocusSet 219
Add 220
Contains 220
CreateIterator 221
InitFocusSet 221
Remove 222
XMPFocusSetIterator 223
~XMPFocusSetIterator 224
First 224
IsNotComplete 225
Next 225
XMPFrame 226
ChangeFrameShape 231
ChangeInternalTransform 232
ChangeLinkStatus 232
ChangePart 233
ChangePresentation 233
ChangeUsedShape 234
ChangeViewType 234
CloneTo 235
Close 235
ContentChanged 236
CreateFacetIterator 236
DoesPropagateEvents 237
DrawActiveBorder 237
Externalize 237
FacetAdded 238
FacetRemoved 238
GetContainingFrame 238
GetFrameGroup 239
GetFrameShape 239
GetInternalTransform 239
GetLinkStatus 240
GetPart 240
GetPartInfo 240



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

GetPresentation 241
GetSequenceNumber 241
GetUsedShape 241
GetViewType 242
Invalidate 242
InvalidateActiveBorder 242
IsDragging 243
IsDroppable 243
IsFrozen 243
IsOverlaid 244
IsRoot 244
IsSubframe 244
Purge 245
Release 245
Remove 245
RequestFrameShape 246
SetContainingFrame 246
SetDragging 247
SetDroppable 247
SetFrameGroup 247
SetFrozen 248
SetPartInfo 248
SetPresentation 248
SetPropagateEvents 249
SetSequenceNumber 249
SetSubframe 250
SetViewType 250
Validate 250
XMPFrameFacetIterator 251
XMPFrameFacetIterator 251
~XMPFrameFacetIterator 252
First 252
IsNotComplete 253
Next 253
XMPLink 254
CloneTo 255
Externalize 256
GetChangeID 257
GetChangeTime 257
GetContentStorageUnit 258



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

GetStatus 258
Lock 259
RegisterDependent 260
ShowSourceContent 260
Unlock 261
UnregisterDependent 262
XMPLinkManager 263
AnyLinkImported 265
DraftClosing 265
DraftOpened 266
DraftSaved 266
UnsavedExportedLinks 267
XMPLinkSource 268
Clear 270
CloneTo 271
ContentChanged 272
Externalize 273
GetChangeID 273
GetContentStorageUnit 274
GetLink 274
IsAutoUpdate 275
Lock 275
SetAutoUpdate 276
SetSourcePart 276
Unlock 277
XMPLinkSpec 279
XMPLinkSpec 280
~XMPLinkSpec 281
ReadLinkSpec 281
WriteLinkSpec 282
XMPMenuBar 284
XMPMenuBar 285
~XMPMenuBar 286
AddMenuBefore 286
AddMenuLast 287
Copy 288
Display 288
GetMenu 289
InitMenuBar 289
Purge 290



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Release 290
RemoveMenu 291
XMPMessageInterface 292
CreateEvent 293
CreatePartAddrDesc 294
CreatePartObjSpec 295
ProcessSemanticEvent 296
Purge 296
Send 297
XMPNameResolver 299
CallObjectAccessor 300
CreateSwapToken 301
CreateToken 302
DisposeToken 302
GetContainingFrame 303
GetContextInfo 304
Purge 305
Resolve 305
XMPNameSpace 307
Exists 308
GetName 308
GetValue 309
Purge 309
ReadFromFile 310
ReadFromStorage 310
Register 311
Unregister 312
WriteToFile 313
WriteToStorage 313
XMPNameSpaceManager 315
CreateNameSpace 316
DeleteNameSpace 317
HasNameSpace 317
Purge 318
XMPObject 319
XMPObject 320
~XMPObject 320
GetExtension 321
HasExtension 321
InitObject 322



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Purge 323
ReleaseExtension 323
XMPPersistentObject 325
XMPPersistentObject 326
~XMPPersistentObject 327
Externalize 327
GetID 328
GetStorageUnit 328
InitPersistentObject 329
InitPersistentObjectFromStorage 329
XMPPlatformTypeList 331
XMPPlatformTypeList 332
~XMPPlatformTypeList 333
Add 333
AddLast 334
Contains 335
Count 335
CreatePlatformTypeListIterator 336
Difference 336
InitPlatformTypeList 337
Remove 338
Union 338
XMPPlatformTypeListIterator 340
~XMPPlatformTypeListIterator 341
First 341
IsNotComplete 342
Next 342
XMPRefCntObject 344
XMPRefCntObject 345
~XMPRefCntObject 345
IncrementRefCount 346
InitRefCntObject 346
Release 347
XMPSemanticInterface 348
XMPSemanticInterface 350
~XMPSemanticInterface 351
GetCoercionHandler 351
GetEventHandler 352
GetObjectAccessor 353
GetOSLSupportFlags 354



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

GetSpecialHandler 355
InitSemanticInterface 355
InstallAdjustMarksProc 356
InstallCoercionHandler 357
InstallCompareProc 358
InstallCountProc 358
InstallDisposeTokenProc 359
InstallEventHandler 359
InstallGetErrDescProc 360
InstallGetMarkTokenProc 361
InstallMarkProc 361
InstallObjectAccessor 362
InstallSpecialHandler 363
RemoveCoercionHandler 364
RemoveEventHandler 365
RemoveObjectAccessor 366
RemoveSpecialHandler 367
SetOSLSupportFlags 367
XMPSession 369
Close 370
Exists 371
GetArbitrator 371
GetClipboard 372
GetDispatcher 372
GetDragAndDrop 373
GetLinkManager 373
GetMessageInterface 373
GetNameResolver 374
GetNameSpaceManager 374
GetSemanticInterface 375
GetShellSemtInterface 375
GetStorageSystem 376
GetTranslation 376
GetType 376
GetUndo 377
GetWindowState 377
Purge 378
RegisterObject 378
RemoveEntry 379
Tokenize 380



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

UniqueChangeID 380
UnregisterObject 381
XMPShape 382
XMPShape 384
~XMPShape 384
ContainsPoint 385
Copy 385
CopyFrom 386
CopyPolygon 386
GetBoundingBox 387
GetGeometryMode 388
GetPlatformShape 388
HasGeometry 389
Intersect 390
InverseTransform 391
IsEmpty 391
IsRectangular 392
IsSameAs 392
ReadShape 393
SetGeometryMode 394
SetPlatformShape 395
SetPolygon 395
SetRectangle 396
Subtract 397
Transform 397
Union 398
WriteShape 399
XMPStorageSystem 400
CreateContainer 401
GetContainer 402
GetSession 403
NeedSpace 404
ReleaseContainer 404
XMPStorageUnit 406
XMPStorageUnit 409
~XMPStorageUnit 410
AddProperty 410
AddValue 411
ClearAllPromises 412
CloneInto 412



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

CloneTo 414
CopyTo 415
CountProperties 415
CountValues 416
CreateCursor 416
CreateView 417
DeleteValue 418
Exists 419
Exists 420
Externalize 420
Focus 421
Focus 422
GetDraft 423
GetGenerationNumber 424
GetID 424
GetIDFromStorageUnitRef 425
GetName 425
GetOffset 426
GetPromiseValue 427
GetProperty 428
GetSession 429
GetSize 429
GetStorageUnitRefIterator 430
GetStrongStorageUnitRef 430
GetType 431
GetValue 432
GetWeakStorageUnitRef 433
IncrementGenerationNumber 433
IncrementRefCount 434
InitStorageUnit 434
InsertValue 435
Internalize 436
IsPromiseValue 436
IsStrongStorageUnitRef 437
IsWeakStorageUnitRef 437
Lock 438
Purge 439
Release 439
Remove 440
RemoveStorageUnitRef 440



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

SetName 441
SetOffset 441
SetPromiseValue 442
SetType 443
SetValue 444
XMPStorageUnitCursor 445
XMPStorageUnitCursor 446
~XMPStorageUnitCursor 446
GetCursor 446
InitStorageUnitCursor 447
XMPStorageUnitRefIterator 449
XMPStorageUnitRefIterator 450
~XMPStorageUnitRefIterator 450
First 451
InitStorageUnitRefIterator 451
IsNotComplete 452
Next 452
XMPStorageUnitView 453
XMPStorageUnitView 455
~XMPStorageUnitView 456
AddProperty 456
AddValue 457
CloneInto 457
CloneTo 458
CopyTo 459
DeleteValue 460
Externalize 460
GetCursor 461
GetGenerationNumber 461
GetID 462
GetIDFromStorageUnitRef 462
GetName 463
GetOffset 463
GetPromiseValue 464
GetProperty 465
GetSize 465
GetStorageUnit 466
GetStorageUnitRefIterator 466
GetStrongStorageUnitRef 467
GetType 468



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

GetValue 468
GetWeakStorageUnitRef 469
IncrementGenerationNumber 470
InitStorageUnitView 470
InsertValue 471
Internalize 472
IsPromiseValue 472
IsStrongStorageUnitRef 473
IsWeakStorageUnitRef 473
Purge 474
Remove 475
RemoveStorageUnitRef 475
SetName 476
SetOffset 476
SetPromiseValue 477
SetType 478
SetValue 478
XMPTransform 480
XMPTransform 482
~XMPTransform 482
ConvertTransform 483
Copy 484
CopyFrom 484
GetGraphicsSystem 485
GetMatrix 485
GetOffset 486
GetPlatformTransform 486
GetPreScaleOffset 487
GetScale 488
GetType 488
HasMatrix 488
Invert 489
InvertPoint 490
InvertShape 491
IsSameAs 491
MoveBy 492
PostCompose 492
PreCompose 493
ReadFrom 493
Reset 494



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

ScaleBy 494
ScaleDownBy 495
SetMatrix 495
SetOffset 496
SetPlatformTransform 496
TransformPoint 497
TransformShape 498
WriteTo 498
XMPTranslation 500
XMPTranslation 501
~XMPTranslation 502
GetISOTypeFromPlatformType 502
GetPlatformTypeFromISOType 503
GetTranslateMethod 503
GetTranslationOf 504
InitTranslation 505
Purge 505
Translate 506
Translate 507
XMPTypeList 509
XMPTypeList 510
~XMPTypeList 510
Add 511
AddLast 511
Contains 512
Count 513
CreateTypeListIterator 513
InitTypeList 514
InitTypeList 515
Remove 515
XMPTypeListIterator 517
~XMPTypeListIterator 518
First 518
IsNotComplete 519
Next 519
XMPUndo 521
AddActionToHistory 523
ClearActionHistory 524
ClearRedoHistory 524
MarkActionHistory 525



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

PeekRedoHistory 526
PeekUndoHistory 527
Purge 527
Redo 528
Undo 529
XMPValueIterator 530
XMPValueIterator 531
~XMPValueIterator 531
First 531
IsNotComplete 532
Next 533
XMPWindow 534
Close 536
CloseAndRemove 536
GetFacetUnderPoint 537
GetID 538
GetPlatformWindow 538
GetRootFrame 539
GetSourceFrame 539
Hide 540
IsActive 540
IsResizable 541
IsRootWindow 541
IsShown 542
Open 542
Purge 543
Resized 543
SetShouldSave 544
SetShouldShowLinks 544
ShouldSave 545
ShouldShowLinks 545
Show 546
XMPWindowIterator 547
~XMPWindowIterator 548
First 548
IsNotComplete 549
Last 549
Next 550
Previous 550
XMPWindowState 552



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

AdjustPartMenus 554
CloseWindows 554
CopyBaseMenuBar 555
CreateWindowIterator 555
Externalize 556
GetActiveWindow 556
GetRootWindowCount 557
GetTotalRootWindowCount 557
GetWindow 558
GetWindowCount 559
GetXMPWindow 559
Internalize 560
IsXMPWindow 561
OpenWindows 561
Purge 562
SetBaseMenuBar 563
SetDefaultWindowTitles 563

Chapter 3 OpenDoc Enhancement Classes 565

XMPExtension 568
XMPExtension 569
~XMPExtension 569
GetBase 570
InitExtension 570
Release 571
XMPDispatchModule 572
XMPDispatchModule 573
~XMPDispatchModule 573
Dispatch 574
InitDispatchModule 574
XMPFocusModule 576
XMPFocusModule 577
~XMPFocusModule 577
CreateOwnerIterator 578
GetFocusOwner 578
InitFocusModule 579
IsFocusExclusive 579
SetFocusOwnership 580



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`UnsetFocusOwnership` 581

Chapter 4 Global Functions, Types, and Constants 583

Global Functions 586
`OpenXMPSession` 586
`ShowLinkDestinationInfo` 586
`ShowLinkSourceInfo` 587
`ShowPartFrameInfo` 588
`ShowPasteAsDialog` 589
Types 591
Data Interchange 591
Storage System 593
General 593
Imaging 594
Linking 598
UI Subsystem 598
Messaging 599
Constants 601
Name Spaces 601
General 601
Category Constants 602
Parts 602
Frame Groups 602
UI Subsystem 603
Foci 603
Menu IDs 603
OpenDoc Events 604
Command IDs 605
Persistent Properties 607
Persistent Value Types 608
Undo 608
Messaging subsystem 609
General 609
Properties for Part Info Dialog 609
Data Interchange 610
General 610



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Storage System 611
Persistent Properties 612
Persistent Value Types 613



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# About This Book

This book, the *OpenDoc Class Reference*, provides reference documentation to the platform-independent classes, methods, global functions, types, and constants defined by OpenDoc. If you are developing a component application (part handler) for any platform on which OpenDoc is implemented, you need the information in this book.

Before you read this book, you should already be familiar with the basic concepts of OpenDoc, as described in the first chapter of the *OpenDoc Programmer's Guide* [under preparation]. As you read the information in this book, you may also want to refer to subsequent chapters of the *Programmer's Guide*; those chapters describe how the classes and methods described here work together in a functioning part handler.

For detailed tutorial instructions and code samples from actual part handlers, you can use one of the platform-specific versions of the *OpenDoc Cookbook* [also under preparation]. In addition, the *OpenDoc Cookbook* has the reference documentation for all platform-specific methods, types, and constants that are part of OpenDoc on each platform. For example, the Macintosh version of the *OpenDoc Cookbook* documents all Macintosh-specific methods of the class `XMPMenuBar`. [This information is currently in the document *OpenDoc Class Reference: Macintosh Supplement*.]

## What to Read

You can read the chapters of this book in any order.

- Chapter 1, "The Class XMPPart," describes the principal abstract class that you must subclass to create a component application. Once you have overridden the methods of XMPPart, you have done most of the work necessary to create an OpenDoc part handler.
- Chapter 2, "OpenDoc Capability Classes," describes the bulk of the OpenDoc class hierarchy. These are the classes whose methods your part handler uses to perform its tasks. Classes in this chapter are arranged



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

alphabetically for rapid access; methods within each class are arranged alphabetically also.

- Chapter 3, "OpenDoc Enhancement Classes," describes those classes that you can subclass in order to extend the capabilities of OpenDoc. (Scripting support is one such extension that is already provided for you in OpenDoc.) This chapter describes the classes `XMPExtension`, `XMPFocusModule`, and `XMPDispatchModule`, which you can subclass to provide other kinds of enhancements.
- Chapter 4, "Global Functions, Types, and Constants," describes the globally available functions, types, and constants that OpenDoc provides for all platforms.

## Chapter Organization

For the chapters that describe one or more classes, the classes are presented in individual sections arranged in alphabetical order. Each class description gives all information pertaining to that class, including descriptions of its methods.

Each class description includes the following elements:

- The name of the class, its position in the OpenDoc class hierarchy, and a summary statement its purpose
- A description section, which gives additional information about the class, including pertinent instantiation or subclassing information, and information on invariants, if appropriate
- A methods section, which includes an annotated list of methods, followed by detailed descriptions of each of the methods, in alphabetical order

Each method description includes the following elements:

- The name of the method and a summary statement its purpose
- The method prototype, followed descriptions of each of the method's parameters, plus its return value, if any
- A description section, which includes information the operation of the method, calling instructions, preconditions and postconditions, and overriding instructions.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

- A list of exceptions that the method can generate.

Chapter 4 is organized differently. It presents reference descriptions for the global functions provided by OpenDoc, followed by tabular descriptions of the available data types and constants, organized by topic.

# Conventions Used in This Book

This book uses various conventions to present certain types of information.

## Special Fonts

All code listings, reserved words, and the names of data structures, constants, fields, parameters, and functions are shown in Courier (`this is Courier`).

## Interface Language

Function prototypes in this book are currently in C++. In future versions of this document, they will be in IDL (Interface Description Language), the language used by the compiler for SOM (System Object Model), the platform-independent object-handling technology that underlies OpenDoc. IDL syntax is similar, although not identical, to that of C and C++.

# About Component Integration Laboratories

***summary information and address to be provided***



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# PREFACE



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

CHAPTER 1

# The Class XMPPart

## Contents


XMPPart 4
- XMPPart 10
- ~XMPPart 11
- AbortRelinquishFocus 11
- AddDisplayFrame 12
- AdjustBorderShape 13
- AdjustMenus 14
- AttachSourceFrame 14
- BeginRelinquishFocus 15
- CanvasChanged 16
- CanvasUpdated 17
- ChangeKind 17
- CloneInto 18
- CloseDisplayFrame 19
- CommitRelinquishFocus 19
- ContainingPartPropertiesChanged 20
- CreateEmbeddedFramesIterator 21
- CreateLink 22
- DisposeActionState 23
- DragEnter 24
- DragLeave 24
- DragWithin 25
- Draw 26
- Drop 27
- DropCompleted 27
- EmbeddedFrameChanged 28





Draft. Preliminary. © Apple Computer, Inc. 6/20/94

EmbeddedFrameSpec 29
ExternalizeKinds 30
FacetAdded 30
FacetRemoved 31
FocusAcquired 31
FocusLost 32
FrameShapeChanged 33
FulfillPromise 33
GeometryChanged 34
GetContainingPartProperties 35
GetPrintResolution 36
HandleEvent 36
HighlightChanged 37
InitPart 38
InitPartFromStorage 38
LinkStatusChanged 38
LinkUpdated 39
MouseEnter 39
MouseLeave 40
MouseWithin 41
Open 41
PresentationChanged 42
ReadActionState 43
ReadPartInfo 43
RedoAction 44
RemoveDisplayFrame 45
RemoveEmbeddedFrame 45
RequestEmbeddedFrame 46
RequestFrameShape 47
RevealFrame 48
RevealLink 49
SequenceChanged 49
UndoAction 50
UsedShapeChanged 51
ViewTypeChanged 51
WriteActionState 52
WritePartInfo 53



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# The Class XMPPart

This chapter describes the class `XMPPart`, which you must subclass to create a part handler.

**Note**
One recommended method for developing a part handler involves use of a part-handler framework, a set of classes that provide an intermediate step between your part handler's code and the OpenDoc class library. Such a framework is not documented here, nor are any classes that your part handler might implement on its own, independent of or as wrappers to OpenDoc classes. This chapter, like the rest of this book, describes only the classes defined by OpenDoc itself. For information on using a framework to develop a part handler, see [documentation to come]. ◆



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPPart

*Superclasses* `XMPPersistentObject` → `XMPRefCntObject` → `XMPObject`

*Subclasses* none

`XMPPart` is the abstract superclass from which you instantiate subclasses for implementation as part handlers. There can be no instances of `XMPPart` itself, but there will always be instantiations of its subclasses.

`XMPPart` provides a template for the structure and behavior of parts. Each part in an OpenDoc document is represented by one instance of `XMPPart`.

## Description

**Parts** are the building blocks of an OpenDoc document. Each part object is an instantiation of `XMPPart`, and consists of a unit of the document's content, accompanied by the code that manipulates it. **Part content** consists of data of a given structure or type. This data is represented by the instance variables for that instantiation of `XMPPart`. Instantiated part objects are separated by category into **part types**. These part types include various data formats, such as text, graphics, or video. To the end user, parts simply represent the discrete screen areas in a document where one kind of content ends and where another kind begins. Documents may contain the many different kinds of parts listed above.

The unit of code that manipulates the part content is referred to as the **part handler**. Each part handler controls the behavior of one part type. There are two kinds of part handlers: **part editors** and **part viewers**. A part editor can display and change the data contained in its part, whereas the part viewer can only display that data.

Each part type exhibits a unique behavior. This behavior is determined by the part handler. The behavior is unique because each part in an OpenDoc application represents an independent instantiation of `XMPPart`. The unique behavior of each instantiation of `XMPPart` is defined during the implementation of the methods of that instantiation. Because the methods of



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

XMPPart are pure virtual methods, they must be must be implemented by all part handlers during the instantiation of new part objects. One part handler supplies the implementation of behavior for all parts of a given part type.

Parts can embed other parts, and possess little or no information regarding the data contained in that embedded part. Every document possesses a single part that resides at its topmost level. This is referred to as the **root part**. All other parts in this document will be embedded in the root part. Other parts are embedded in parts that are themselves embedded in the root part. A part that contains other parts that are embedded within it is referred to as the **containing part** or **embedding part**. The root part, for instance, is the containing part for the entire document.

Parts participate in many of the OpenDoc protocols. Developers 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.

The fSU field that is inherited from XMPPersistentObject by XMPPart is initialized to point to the part's storage unit. This field can never be set to 0, 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 that 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 that support extensions, including a semantic interface, must manage the creation and deletion of their extension objects.

For more information onXMPPart, see the class descriptions for XMPFacet, XMPFrame, XMPShape, XMPCanvas, XMPStorageUnit, XMPTransform, and XMPLink.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Methods

The methods of `XMPPart` are used for the activation and general manipulation of part objects.

#### Constructor

`XMPPart` — The constructor for the `XMPPart` object.

#### Part Activation

`AbortRelinquishFocus` — Called by the arbitrator to cause the part to cancel changes initiated by the `BeginRelinquishFocus` call, and resume ownership of the focus.

`BeginRelinquishFocus` — Asks a part to give up ownership of a certain focus.

`CommitRelinquishFocus` — Completes the process of relinquishing the ownership of the specified focus that was started in `Part::BeginRelinquishFocus()`.

`FocusAcquired` — Notifies the part that one of its frames has acquired the specified focus.

`FocusLost` — Notifies the part that a focus has unexpectedly been lost.

#### Binding

`ChangeKind` — Asks a part to change into a new kind of content, i.e. ask an ASCII Text part to change into a Styled Text part.

`ExternalizeKinds` — Externalizes a representation for each 'kind' that it supports.

#### Data Interchange

`FulfillPromise` — Fulfills a previously made promise by providing the actual data the promise represents.

#### DragAndDrop

`DragEnter` — Activates the part used to track the drag.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `DragLeave` | Signals that the drag operation is complete, and so deactivates the part from drag tracking. |
| `DragWithin` | Tracks the drag operation, and provides graphical feedback regarding possible drop targets. |
| `Drop` | Transfers the dragged data into this part. |
| `DropCompleted` | Notifies the part that the drop is completed. |

**Embedding**

| | |
|---|---|
| `ContainingPartPropertiesChanged` | Used by `ContainingPart` to inform the contained part of changes to the content properties of that object within the containing part's content model that encapsulates the embedded frame. |
| `CreateEmbeddedFramesIterator` | Creates an object that will iterate over the frames that are embedded within a display frame of this part. |
| `EmbeddedFrameSpec` | Creates an object specifier for the embedded frame. |
| `GetContainingPartProperties` | Used by an embedded part to create a storage unit and write properties that the containing part associates with the embedded frame into it, before returning it. |
| `RevealFrame` | Asks a part to make an embedded frame visible. |

**Facet**

| | |
|---|---|
| `FacetAdded` | Notifies the part that a facet has been added to one of its display frames. |
| `FacetRemoved` | Notifies the part that a facet has been removed from one of its display frames. |
| `GeometryChanged` | Called by a facet of the part to inform it that the clipping shape and/or external transform of that facet has changed. |

**Frames**

| | |
|---|---|
| `AddDisplayFrame` | Tells a part to add "frame" as one of its display frames. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `AdjustBorderShape` | Adjusts the shape of the active frame border of an embedded frame. |
| `AttachSourceFrame` | Called by the object that requested creation of a frame, immediately after the creation of the frame. |
| `CloseDisplayFrame` | Informs a part that one of its display frames is closing. |
| `CreateEmbeddedFrame` | Asks the part to create a new frame and embed a part in it. |
| `FrameShapeChanged` | Notifies the part that the frame shape of one of its display frames has been changed by its containing part. |
| `Open` | Creates and opens a presentation of the part in a frame in a new window. |
| `PresentationChanged` | Notifies the part that the presentation kind of one of its frames has been changed. |
| `ReadPartInfo` | Internalizes the `partInfo` for a display frame of this part. |
| `RemoveDisplayFrame` | Removes the frame from the list of display frames. |
| `RemoveEmbeddedFrame` | Removes a frame used for viewing an embedded part. |
| `RequestEmbeddedFrame` | Asks the part to create a new frame and embed a part in it. |
| `RequestFrameShape` | Used by an embedded part to ask for a change in the shape of its frame. |
| `SequenceChanged` | Informs the part that the sequencing of a group of its display frames has been changed. |
| `UsedShapeChanged` | Informs a containing part that the used shape of one of its embedded frames has changed. |
| `ViewTypeChanged` | Notifies the part that the `viewType` of one of its frames has been changed. |
| `WritePartInfo` | Externalizes the frame's `partInfo` data onto the frame's storage unit. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

### Imaging

| | |
|---|---|
| `CanvasChanged` | Notifies the part that one of its facets has been moved to image on a different canvas. |
| `CanvasUpdated` | Notifies the part that a canvas that it owns has been updated and that its contents need to be copied to its parent canvas. |
| `Draw` | Draws the part in the facet, updating the portion of the facet in the `invalidShape`. |
| `GetPrintResolution` | Returns the minimum desired resolution in dots per inch that this part requires for printing the contents of the specified frame. |
| `HighlightChanged` | Notifies the part that the highlight state of one of its frames has been changed. |

### Linking

| | |
|---|---|
| `CreateLink` | Creates a new link object. |
| `EmbeddedFrameChanged` | Notifies the containing part that the content displayed in the argument "embedded frame" has changed. |
| `LinkStatusChanged` | Allows the part to set the link status of any embedded frames. |
| `LinkUpdated` | Retrieves the data from the link and incorporates it into this part at the link's destination, thereby replacing any previous content of the link. |
| `RevealLink` | Reveals data that was previously linked in a window, making it available for viewing in a display frame. |

### Storage

| | |
|---|---|
| `CloneInto` | Externalizes this `XMPPart` into `storageUnit`. |

### UI Events

| | |
|---|---|
| `AdjustMenus` | Called by the `windowState` object just prior to `MenuSelect` or `MenuKey` to allow the part to adjust the presentation of its menus for display. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `HandleEvent` | Returns a Boolean specifying whether this part handled the event. |
| `MouseEnter` | Called when the mouse is moved over a display frame of this part (mouse button up). |
| `MouseLeave` | Called when the mouse is moved out of a display frame of this part (mouse button up). |
| `MouseWithin` | Called when the mouse is moved within a display frame of this part (mouse button up). |

**Undo**

| | |
|---|---|
| `DisposeActionState` | Tells the part to dispose of the action data. |
| `InitPart` | Called by the draft object. |
| `InitPartFromStorage` | Called by the draft object. |
| `ReadActionState` | Internalizes the `undoAction` data from the storage unit. |
| `RedoAction` | Tells the part to Redo the action described by `actionState`. |
| `UndoAction` | Tells the part to undo the action that is described by `actionState`. |
| `WriteActionState` | Writes out the `actionState` data onto a `StorageUnitView`, and externalizes the `undoAction` data to the storage unit. |

**Destructor**

| | |
|---|---|
| `~XMPPart` | The destructor for the `XMPPart` class. |

Detailed descriptions of these methods follow, in alphabetical order.

## XMPPart

The `XMPPart` method is the constructor for the `XMPPart` object.

```
XMPPart();
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Called only by the draft object. You must override this method.

This constructor is never called directly. It can only be called by invoking a constructor for a derived class.

## ~XMPPart

The `~XMPPart` method is the destructor for the `XMPPart` class.

```
~XMPPart();
```

**DESCRIPTION**

Called only by the draft object. You must override this method.

This destructor is never called directly. It can only be called by invoking a destructor for a derived class.

## AbortRelinquishFocus

The `AbortRelinquishFocus` method is called by the arbitrator to cause the part to cancel changes initiated by the `BeginRelinquishFocus` call, and resume ownership of the focus.

```
void AbortRelinquishFocus  (XMPTypeToken focus,
                            XMPFrame* ownerFrame,
                            XMPFrame* proposedFrame);
```

`focus` A tokenized string representing the focus type that is being relinquished.

`ownerFrame` A reference to the frame that currently possesses the focus.

`proposedFrame`
A reference to the frame of the part that has requested the focus.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Called by the arbitrator object. You must override this method.

The part must return itself to the state where it previously possessed the focus, before continuing use of that focus. Before calling this method, you must ensure that the arbitrator has previously called `XMPPart::BeginRelinquishFocus()`.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrInvalidFocus` | Part doesn't own this focus. |
| `kXMPErrNotPrepared` | Part hasn't prepared to relinquish focus. |
| `kXMPErrInvalidFrame` | the `ownerFrame` is not a display frame of this part. |

## AddDisplayFrame

The `AddDisplayFrame` method tells a part to add "frame" as one of its display frames.

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

`frame` A reference to the new frame to add.

**DESCRIPTION**

This method is only called by frame objects during their initialization. You must override this method.

The part must record "frame" in whatever internal structures it uses to remember its display frames. The new frame will carry a suggested `viewType` and presentation information. The part should look at those settings, and decide if it can support them; if not, it should update those settings in the frame to reflect a default presentation that it can support. Note that the part must support the required set of standard `viewTypes` (frame, icon, small icon, thumbnail, etc.). The part will then create the appropriate `partInfo` data and store it in the frame. This data allows the part to distinguish the frame from its



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

other display frames, and is also a handy place to store other view-related information.

## AdjustBorderShape

The `AdjustBorderShape` method adjusts the shape of the active frame border of an embedded frame.

```
XMPShape* AdjustBorderShape   (XMPFrame* embeddedFrame,
                               XMPShape* shape);
```

`embeddedFrame`
: A reference to an embedded frame that is contained in this part, for which the border is being created.

`shape`
: A reference to the starting shape for the active border of the embedded frame.

*return value*
: The shape to use for the active border, clipped by this part's content.

### DESCRIPTION

Called by the arbitrator object. You must override this method.

The “shape” parameter specifies the requested shape. The return value specifies that shape as it is obscured by the other contents of this part.

### EXCEPTIONS

| | |
|---|---|
| `kXMPErrInvalidFrame` | The embeddedFrame is not an `embeddedFrame` of this part. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## AdjustMenus

The `AdjustMenus` method is called by the `windowState` object just prior to `MenuSelect` or `MenuKey` to allow the part to adjust the presentation of its menus for display.

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

`frame` A reference to the display frame for which the menus need to be adjusted.

**DESCRIPTION**

Called by the windowState object. You must override this method.

When this method is called the part should enable and disable menus as appropriate to its current state.

**EXCEPTIONS**

`kXMPErrInvalidFrame` The specified frame is not a display frame of this part.

## AttachSourceFrame

The `AttachSourceFrame` method is called by the object that requested creation of a frame, immediately after the creation of the frame.

```
void AttachSourceFrame  (XMPFrame* frame,
                         XMPFrame* sourceFrame);
```

`frame` A reference to a display frame for this part.

`sourceFrame`
A reference to another display frame for this part.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

You must override this method.

Calling this method 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 receive this call just after a display frame has been added. Attaching a source frame should cause the display frame to look identical to it. If presentations differ, it causes the display frame to be equivalent to it. This causes duplication of the embedded frames, and ensures that the view in one frame is updated when content in the other is changed. Before calling this method, you must ensure that both `frame` and `sourceFrame` are display frames of the part.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrInvalidFrame` | Either `frame` or `sourceFrame` is not a display frame of this part. |

## BeginRelinquishFocus

The `BeginRelinquishFocus` method asks a part to give up ownership of a certain focus.

```
XMPBoolean BeginRelinquishFocus  (XMPTypeToken focus,
                                  XMPFrame* ownerFrame,
                                  XMPFrame* proposedFrame);
```

| | |
|---|---|
| `focus` | The focus type being relinquished. |
| `ownerFrame` | A reference to the part's frame that possesses the focus. |
| `proposedFrame` | A reference to the frame of belonging to the other part that has requested the focus. |
| *return value* | Returns a Boolean specifying whether the part can give up the requested focus. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Called by the arbitrator object. You must override this method.

Upon receiving this call, a part decides if it can give up the requested focus. If it determines the answer to be yes, it prepares to relinquish the focus. It also ensures that, during the process of relinquishing the specified focus, and before `CommitRelinquishFocus` is called, it does not enter a state where the focus can't be given up. Before calling this method, you must ensure that part owns the requested focus.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrInvalidFocus` | Part doesn't own this focus. |
| `kXMPErrInvalidFrame` | The specified `ownerFrame` is not a display frame of this part. |

## CanvasChanged

The `CanvasChanged` method notifies the part that one of its facets has been moved to image on a different canvas.

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

`facet` A reference to the facet that has a new canvas.

**DESCRIPTION**

Called by the facet object. You must override this method.

The part must update any internal state necessary for compliance with this method.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## CanvasUpdated

The `CanvasUpdated` method notifies the part that a canvas that it owns has been updated and that its contents need to be copied to its parent canvas.

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

`canvas` A reference to the canvas that has been updated.

**DESCRIPTION**

Called by the facet object. You must override this method.

The data must be copied from an off-screen canvas, transformed, and then placed in the containing canvas.

## ChangeKind

The `ChangeKind` method asks a part to change into a new kind of content, i.e. ask an ASCII Text part to change into a Styled Text part.

```
void ChangeKind (XMPType kind);
```

`kind` The new kind for the part.

**DESCRIPTION**

You must override this method.

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 content property of the part's storage unit should become the given kind. Before calling this method, you must ensure that the part supports the specified `kind`. After this method is executed successfully, the part's primary kind is the one that was specified in the parameter, and the first type of the first value in the content property of the part's storage unit is of the specified type. When it reads in data, the part uses this representation.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrInvalidKind` | Part does not support that kind of content. |

## CloneInto

The `CloneInto` method externalizes this `XMPPart` into `storageUnit`.

```
void CloneInto (XMPDraftKey key,
                XMPStorageUnit* storageUnit,
                XMPStorageUnit* initiatingFrame);
```

`key` The `XMPDraftKey` identifying the Clone transaction

`storageUnit`
A reference to the `XMPStorageUnit` referring to the destination storage unit.

`initiatingFrame`
A reference to the `initiatingFrame`.

**DESCRIPTION**

This method is generally called by `XMPFrame::CloneTo`. You must override this method.

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 that are accessible from `initiatingFrameSU` will be copied. If `initiatingFrameSU` is `kXMPNull`, all the storage units that are accessible from this storage unit will be copied. Note that the actual copying may not be completed until after `EndClone` is finished.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## CloseDisplayFrame

The `CloseDisplayFrame` method informs a part that one of its display frames is closing.

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

`frame` A reference to the frame that is to be closed.

**DESCRIPTION**

Called by the frame object. You must override this method.

The part should remove "frame" from its list of display frames, call `Frame::Close()` on any frames that are embedded within that frame, and then release the frame.

**EXCEPTIONS**

`kXMPErrInvalidFrame` The specified frame is not a display frame of this part.

## CommitRelinquishFocus

The `CommitRelinquishFocus` method completes the process of relinquishing the ownership of the specified focus that was started in `Part::BeginRelinquishFocus()`.

```
void CommitRelinquishFocus (XMPTypeToken focus,
                            XMPFrame* ownerFrame,
                            XMPFrame* proposedFrame);
```

`focus` The focus type that is being relinquished.

`ownerFrame` A reference to the part's frame that possesses the focus.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`proposedFrame`
A reference to the frame belonging to the other part that has requested the focus.

**DESCRIPTION**

Called by the arbitrator object. You must override this method.

In the process of relinquishing ownership of the specified focus, Part should release whatever external resources are associated with that focus.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrInvalidFocus` | Part doesn't own this focus. |
| `kXMPErrNotPrepared` | Part hasn't prepared to relinquish focus. |
| `kXMPErrInvalidFrame` | The specified `ownerFrame` is not a display frame of this part. |

## ContainingPartPropertiesChanged

The `ContainingPartPropertiesChanged` method is used by `ContainingPart` to inform the contained part of changes to the content properties of that object within the containing part's content model that encapsulates the embedded frame.

```
void ContainingPartPropertiesChanged
                              (XMPFrame* frame,
                               XMPStorageUnit* propertyUnit);
```

`frame` A reference to the specified frame.

`propertyUnit`
A reference to the specified `propertyUnit`

**DESCRIPTION**

You must override this method.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

This method inspects “propertyUnit” for properties that this part can understand. Where applicable, it incorporate those properties into part's content data. It ignores inapplicable properties, without signalling an error.

**DESCRIPTION**

You must override this method.

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` parameter 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()`.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrCannotEmbed` | This part does not support embedding. |
| `kXMPErrOutOfMemory` | There is not enough memory to embed a part. |

## CreateEmbeddedFramesIterator

The `CreateEmbeddedFramesIterator` method creates an object that will iterate over the frames that are embedded within a display frame of this part.

```
XMPEmbeddedFramesIterator* CreateEmbeddedFramesIterator
                                        (XMPFrame* frame);
```

| | |
|---|---|
| `frame` | A reference to the display frame of this part. |
| *return value* | The iterator. Caller must delete the iterator when finished using it. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

You must override this method.

To indicate all the display frames, use `kXMPNull` for the frame parameter.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrOutOfMemory` | Not enough memory to allocate iterator. |
| `kXMPErrCannotEmbed` | This part does not support embedding. |

## CreateLink

The `CreateLink` method creates a new link object.

```
XMPLink* CreateLink  (XMPPtr data,
                      XMPULong size);
```

| | |
|---|---|
| `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. |
| *return value* | The link object. |

**DESCRIPTION**

You must override this method.

If a link already exists to the content identified by the data and size arguments, this method returns the link object; otherwise, it creates a new link object, puts in the initial data and returns it to the caller.

This method identifies the content to be linked (by resolving the object specifier saved in the data parameter, or by some other means). It then creates a link object to represent the content data. The part must maintain information about what portion of its contents have been linked to it, so that it may notify link clients when that data has been changed. The link created is returned to the caller. Before calling this method, you must ensure that the data identifies some



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

portion of this part's contents. After calling this method successfully, this part maintains a link to the identified content.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrCantLink` | Part does not support linking protocol. |
| `kXMPErrInvalidLinkData` | Could not identify content for link. |
| `kXMPErrOutOfMemory` | There is not enough memory to allocate link object. |

## DisposeActionState

The `DisposeActionState` method tells the part to dispose of the action data.

```
void DisposeActionState (XMPActionData actionState,
                         XMPDoneState doneState);
```

`actionState`
Information previously logged by the part to allow it to undo the action.

`doneState` State of the undo action. It is `kDone` or `kReDone` if the action was on the document's Undo stack, and `kUndone` if it was on the Redo stack.

**DESCRIPTION**

You must override this method.

Before calling this method, you must ensure that "actionState" is an action data block previously logged by this part. After calling this method successfully, memory for "actionState" has been reclaimed. It is no longer usable to perform undo operations.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrInvalidUndo` | Not an undo action of this part. |

## DragEnter

The `DragEnter` method activates the part used to track the drag.

```
void DragEnter (XMPDragItemIterator* dragInfo,
                XMPFacet* facet,
                XMPPoint where);
```

| | |
|---|---|
| `dragInfo` | A reference to the drag item iterator describing the contents, as well as the types and values of the dragged data. |
| `facet` | A reference to the facet of the part that the drag occurs in. |
| `where` | The mouse coordinates of the drag, expressed in frame coordinates. |

**DESCRIPTION**

Called by the facet object. You must override this method.

The part should display a drag target border within the facet. During drag tracking (`DragEnter`, `DragWithin`) the part should never attempt to read data from any of the storage units supplied by the iterator. The part should only inspect the type of the dragged data. Before calling this method, you must ensure that the part is ready to receive `DrawWithin` messages.

## DragLeave

The `DragLeave` method signals that the drag operation is complete, and so deactivates the part from drag tracking.

```
void DragLeave (XMPFacet* facet,
                XMPPoint where);
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `facet` | A reference to the facet of the part that the drag is in. |
| `where` | The mouse coordinates of the drag, expressed in frame coordinates. |

**DESCRIPTION**

You must override this method.

This method un-highlights any content previously highlighted during drag tracking. It also removes the drag target border.Before calling this method you must ensure that the part was previously activated via `DragEnter()`.After executing tis method successfully, the part is no longer active, and cannot receive further `DragWithin()` or`Drop()` messages.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrNotDragging` | Part has not received a DragEnter call. |

## DragWithin

The `DragWithin` method tracks the drag operation, and provides graphical feedback regarding possible drop targets.

```
void DragWithin   (XMPDragItemIterator* dragInfo,
                   XMPFacet* facet,
                   XMPPoint where);
```

| | |
|---|---|
| `dragInfo` | A reference to the drag item iterator describing the contents, as well as the types and values of the dragged data. |
| `facet` | A reference to the facet of the part that the drag is in. |
| `where` | The mouse coordinates of the drag, expressed in frame coordinates. |

**DESCRIPTION**

You must override this method.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

This method highlights the appropriate content to provide feedback about potential drop targets, based on type of dragged data. It gets the type information out of the `dragInfo`. During drag tracking (`DragEnter`, `DragWithin`) the part should never attempt to read data from any of the storage units supplied by the iterator. The part should only inspect the type of the dragged data. Before calling tis method, you must ensure that the part was previously activated via `DragEnter`().

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrNotDragging` | Part has not received a DragEnter call. |

## Draw

The `Draw` method draws the part in the facet, updating the portion of the facet in the `invalidShape`.

```
void Draw   (XMPFacet* facet,
             XMPShape* invalidShape);
```

facet — A reference to the facet in which to draw the part.

invalidShape — A reference to the portion of the facet to update, expressed in frame coordinates.

**DESCRIPTION**

Called by the facet object. You must override this method.

The part should draw itself on the facet's canvas. The part must examine its canvas' `isDynamic` flag to determine if it will be drawing on the screen or to a printer, and then draw itself appropriately.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrInvalidFrame` | “frame” isn't one of the part's display frames. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Drop

The `Drop` method transfers the dragged data into this part.

```
XMPDropResult Drop   (XMPDragItemIterator* dropInfo,
                      XMPFacet* facet,
                      XMPPoint where);
```

| | |
|---|---|
| `dropInfo` | A reference to the drag item iterator describing the contents, and types and values of the dragged data. |
| `facet` | A reference to the facet of the part that the drag is in. |
| `where` | The mouse coordinates of the drag, expressed in frame coordinates. |

*return value*

**DESCRIPTION**

Called by the facet object.You must override this method.

Respond to the Drop operation. This might involve moving or copying data in to the part, or some other response entirely. However, the response should use the data returned from the dropInfo iterator to perform the operation. Before calling this method, you must ensure that the part was previously activated via `DragEnter()`.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrNotDragging` | Part hasn't received a `DragEnter` message. |

## DropCompleted

The `DropCompleted` method notifies the part that the drop is completed.

```
void DropCompleted   (XMPPart* destPart,
                      XMPDropResult dropResult);
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`destPart` A reference to the destination part of the drag.

`dropResult` The result code of the drop operation.

**DESCRIPTION**

Called by the dragAndDrop object. You must override this method.

`DropCompleted` is a peculiar method. If `DragAndDrop::StartDrag` is synchronous, the return code to `DragAndDrop::StartDrag` is the drop result. `Part::DropCompleted` does not need to be called. However, if it is required that `DragAndDrop::StartDrag` returns immediately, there needs to be a way to notify the result of the drop. Before calling this method, you must ensure that the part previously started an asynchronous drag.

## EmbeddedFrameChanged

The `EmbeddedFrameChanged` method notifies the containing part that the content displayed in the argument "embedded frame" has changed.

```
void EmbeddedFrameChanged  (XMPFrame* frame,
                            XMPChangeID change);
```

`frame` A reference to the embedded frame whose contents have changed.

`change` A unique change ID.

**DESCRIPTION**

Called by an frame object belonging to an embedded part when the frame object's `ContentChanged` method is called.You must override this method.

The part should call the `ContentChanged` method of any of its frames that would be interested in this information. This method should only be called by frame objects. The part is not responsible for passing this notification on to its containing part. The part may ignore this notification if it is uninterested in changes to embedded content. A part should wait a certain length of time (a second perhaps) before updating its display so that subsequent calls to



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

EmbeddedFrameChanged with the same XMPChangeID don't result in multiple updates for the same change.After executing tis method successfully, the ContentChanged method of any of the part's frames may have been called and the part may have taken action to update its display.

## EmbeddedFrameSpec

The EmbeddedFrameSpec method creates an object specifier for the embedded frame.

```
void EmbeddedFrameSpec  (XMPFrame* embeddedFrame,
                         XMPObjectSpec* spec);
```

`embeddedFrame`
A reference to the frame you want to create a specifier for.

`spec` A reference to the frame's specifier.

### DESCRIPTION

You must override this method.

If this part is itself embedded, this method asks its containing part for the specifier for the part's display frame, then concatenates the specifier for that embedded frame.

### EXCEPTIONS

| | |
|---|---|
| `kXMPErrInvalidFrame` | `embeddedFrame` is not an embedded frame of this part. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## ExternalizeKinds

The `ExternalizeKinds` method externalizes a representation for each 'kind' that it supports.

```
void ExternalizeKinds (XMPTypeList* kindlist);
```

`kindlist` A reference to the set of kinds the part should attempt to externalize.

**DESCRIPTION**

You must override this method.

The part should iterate through all the kinds in 'kindList', and externalize a representation for each that it supports. This call does not specify anything about the ordering of those kinds in the contents property. Before calling this method, you must ensure that the part supports a subset of the kinds in 'kindList'. After executing this method successfully, the contents property of the part has a value and representation for each kind in `kindListsupported`.

## FacetAdded

The `FacetAdded` method notifies the part that a facet has been added to one of its display frames.

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

`facet` A reference to the facet that has been added.

**DESCRIPTION**

Called by a display frame of the part. You must override this method.

If the part does not display asynchronously and it isn't a containing part, it doesn't need to do anything in response to this call. If the part displays



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

asynchronously, it must make sure to draw in the new facet as well. If the part has embedded frames, it must create facets for them.

## FacetRemoved

The `FacetRemoved` method notifies the part that a facet has been removed from one of its display frames.

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

`facet` A reference to the facet that is being removed.

**DESCRIPTION**

Called by a display frame of this part. You must override this method.

If the part does not display asynchronously and isn't a containing part, it doesn't need to do anything in response to this call. If the part displays asynchronously, it must stop drawing in the facet. If the part has embedded frames, it must remove their facets as well.

**EXCEPTIONS**

`kXMPErrInvalidFacet` The specified facet is not a facet of this part.

## FocusAcquired

The `FocusAcquired` method notifies the part that one of its frames has acquired the specified focus.

```
void FocusAcquired   (XMPTypeToken focus,
                      XMPFrame* ownerFrame);
```

`focus` A tokenized string representing the focus type to acquire.

`ownerFrame` A reference to the frame that gets the focus.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

You must override this method.

This method may be 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.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrInvalidFocus` | Part cannot own this focus. |
| `kXMPErrInvalidFrame` | The specified `ownerFrame` is not a display frame of this part. |

## FocusLost

The `FocusLost` method notifies the part that a focus has unexpectedly been lost.

```
void FocusLost (XMPTypeToken focus,
                XMPFrame* ownerFrame);
```

| | |
|---|---|
| `focus` | The focus that was lost. |
| `ownerFrame` | A reference to the frame from which the focus was lost. |

**DESCRIPTION**

You must override this method.

This method takes whatever actions necessary to recover from having lost the focus, i.e. close connections, de-highlight selection, etc.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrInvalidFocus` | Part doesn't own this focus. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `kXMPErrInvalidFrame` | The specified `ownerFrame` is not a display frame of this part. |

## FrameShapeChanged

The `FrameShapeChanged` method notifies the part that the frame shape of one of its display frames has been changed by its containing part.

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

| | |
|---|---|
| `frame` | A reference to the frame that is being reshaped. |

**DESCRIPTION**

Called by `Frame::ChangeFrameShape`. You must override this method.

The part should take whatever actions are necessary to respond to the new shape. It may need to re-layout its content, change its used shape, resize its embedded frames, or something else. It also has the option of turning around and asking the frame for a different frame shape via `RequestFrameShape`, though it must be able to handle the shape it is given. If the size of the frame is not sufficient, the part may ask the containing part for a continuation frame via `CreatEmbeddedFrame`.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrInvalidFrame` | The specified frame is not a display frame of this part. |

## FulfillPromise

The `FulfillPromise` method fulfills a previously made promise by providing the actual data the promise represents.

```
void FulfillPromise (XMPStorageUnitView *promiseSUView);
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`promiseSUView`
A reference to a view into the storage unit that contains the promise value.

**DESCRIPTION**

You must override this method.

The `promiseSUView` is a view into the same storage unit value that is created by `SetPromiseValue`. The information about the promise is encoded in the `promiseSUView` in the Part Handler's own internal format. The promise data is written into the `promiseSUView` replacing the data stored in the part handler's format.

When the promise was made, a record of it was kept by this part. The `promiseSUView` contains the part's information for that promise. The part uses the data in the `promiseSUView` to determine which data to move/copy. The part then writes the data into the `promiseSUView`.

**EXCEPTIONS**

`kXMPErrInvalidPromise` The part did not make this promise.

## GeometryChanged

The `GeometryChanged` method is called by a facet of the part to inform it that the clipping shape and/or external transform of that facet has changed.

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

`facet` A reference to the facet where the clip shape has changed.

**DESCRIPTION**

Called by the facet object. You must override this method.

The part should use the new clip shape for display from now on. Parts that display only in response to update events don't need to do anything special,



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

provided they check the clip shape each time they draw. Parts that display asynchronously (like clocks, movies, etc.) must notice their new clipping and limit their display accordingly.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrInvalidFacet` | The specified facet is not a facet of this part. |

## GetContainingPartProperties

The `GetContainingPartProperties` method is used by an embedded part to create a storage unit and write properties that the containing part associates with the embedded frame into it, before returning it.

```
XMPStorageUnit* GetContainingPartProperties
                                        (XMPFrame* frame);
```

| | |
|---|---|
| `frame` | A reference to an embedded frame of this part. |
| *return value* | A storage unit containing the container properties for the specified frame. The caller has responsibility for deallocating storage. |

**DESCRIPTION**

Called by an embedded part. You must override this method.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrCannotEmbed` | This part does not support embedding. |
| `kXMPErrInvalidFrame` | The specified frame is not an embedded frame of this part. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetPrintResolution

The `GetPrintResolution` method returns the minimum desired resolution in dots per inch that this part requires for printing the contents of the specified frame.

```
XMPULong GetPrintResolution (XMPFrame* frame);
```

| | |
|---|---|
| `frame` | A reference to the frame for which the resolution is needed. |
| *return value* | The desired resolution in dots per inch. |

### DESCRIPTION

Called by part or the application and is used in performing printing. You must override this method.

### EXCEPTIONS

| | |
|---|---|
| `kXMPErrInvalidFrame` | The specified frame is not a display frame of this part. |

## HandleEvent

The `HandleEvent` method returns a Boolean specifying whether this part handled the event.

```
XMPBoolean HandleEvent  (XMPEventData event,
                         XMPFrame* frame,
                         XMPFacet* facet);
```

| | |
|---|---|
| `event` | A wrapper for platform-specific event data. |
| `frame` | A reference to the frame in which the event occurred. |
| `facet` | A reference to the facet in which the event occurred. `kXMPNull` is returned for events not based on geometry, such as keyboard events. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

*return value* Returns a Boolean specifying whether the part was able to handle the event.

**DESCRIPTION**

Called by the dispatcher object. You must override this method.

Part responds to the UI event if it can.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrInvalidFrame` | The specified frame is not a display frame of this part. |
| `kXMPErrInvalidFacet` | The specified facet is not a facet of this part. |

## HighlightChanged

The `HighlightChanged` method notifies the part that the highlight state of one of its facets has been changed.

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

`facet` A reference to a facet of the part.

**DESCRIPTION**

Called by the frame object. You must override this method.

It adjusts this part's presentation in the facet to its new highlight state. New state may be found by calling `facet->GetHighlight()`. It passes notification to embedded parts.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrInvalidFrame` | The specified facet is not a facet of this part. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## InitPart

The `InitPart` method is called by the draft object.

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

**DESCRIPTION**

You must override this method.

## InitPartFromStorage

The `InitPartFromStorage` method is called by the draft object.

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

**DESCRIPTION**

You must override this method.

## LinkStatusChanged

The `LinkStatusChanged` method allows the part to set the link status of any embedded frames.

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

`frame` A reference to a frame

**DESCRIPTION**

Called by a part's frame when that frame's `ChangeLinkStatus` method is called. You must override this method.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Frames notify their owner parts that their link status has changed whenever `ChangeLinkStatus` is called. After this method executes successfully, the part will call `ChangeLinkStatus` on any embedded frames that are involved in the link in question.

## LinkUpdated

The `LinkUpdated` method retrieves the data from the link and incorporates it into this part at the link's destination, thereby replacing any previous content of the link.

```
void LinkUpdated  (XMPLink* updatedLink,
                   XMPChangeID id);
```

`updatedLink`
A reference to the link that was changed.

`id` The change id that is associated with the link

**DESCRIPTION**

You must override this method.

**EXCEPTIONS**

`kXMPErrInvalidLink` This isn't one of the part's links.

## MouseEnter

The `MouseEnter` method is called when the mouse is moved over a display frame of this part (mouse button up).

```
void MouseEnter   (XMPFacet* facet,
                   XMPPoint where);
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `facet` | A reference to the facet into which the mouse has moved. |
| `where` | The mouse location in frame coordinates. |

**DESCRIPTION**

Called by the dispatcher object. You must override this method.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrInvalidFacet` | The specified facet is not a facet of this part. |

## MouseLeave

The `MouseLeave` method is called when the mouse is moved out of a display frame of this part (mouse button up).

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

| | |
|---|---|
| `facet` | A reference to the facet that the mouse has left. |

**DESCRIPTION**

Called by the dispatcher object. You must override this method.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrInvalidFacet` | The specified facet is not a facet of this part. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## MouseWithin

The `MouseWithin` method is called when the mouse is moved within a display frame of this part (mouse button up).

```
void MouseWithin  (XMPFacet* facet,
                   XMPPoint where);
```

`facet` A reference to the facet in which the mouse has moved.

`where` The mouse location in frame coordinates.

**DESCRIPTION**

Called by the dispatcher object. You must override this method.

**EXCEPTIONS**

`kXMPErrInvalidFacet` The specified facet is not a facet of this part.

## Open

The `Open` method creates and opens a presentation of the part in a frame in a new window.

```
XMPID Open (XMPFrame* frame);
```

`frame` A reference to the frame that should be used as a model for the new presentation. `kXMPNull` means that the part should create a new frame and open it in the window.

*return value* A unique id for the new window.

**DESCRIPTION**

You must override this method.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

This method adds this part as the root part of the window. It bases the presentation in the new frame on the presentation in the old frame, or on a default presentation if there is no old frame.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrInvalidFrame` | The specified frame isn't one of the part's display frames. |

## PresentationChanged

The `PresentationChanged` method notifies the part that the presentation kind of one of its frames has been changed.

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

| | |
|---|---|
| `frame` | A reference to a display frame of this part. |

**DESCRIPTION**

Called by `Frame::ChangePresentation()`. You must override this method.

The part should examine the new presentation kind via `frame->GetPresentation()`. If the part does not support that kind of presentation, it should correct the frame's presentation using `frame->SetPresentation()`. The part should then adjust its display in the display frame to be of the new presentation kind.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrInvalidFrame` | The specified frame isn't one of the part's display frames. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## ReadActionState

The `ReadActionState` method internalizes the `undoAction` data from the storage unit.

```
XMPPtr ReadActionState
                        (XMPStorageUnitView* storageUnitView);
```

`storageUnitView`
A reference to where to get the action data.

*return value* Information previously logged by the part to allow it to undo the action.

### DESCRIPTION

You must override this method.

This method reads the `actionState` data from a view on a `StorageUnit`. It then allocates the memory and passes ownership of the storage to the caller.

### EXCEPTIONS

| | |
|---|---|
| `kXMPErrInvalidUndo` | Not an undo action of this part. |
| `kXMPErrOutOfMemory` | Not enough memory to internalize data. |

## ReadPartInfo

The `ReadPartInfo` method internalizes the `partInfo` for a display frame of this part.

```
XMPPtr ReadPartInfo  (XMPFrame* frame,
                       XMPStorageUnitView* storageUnitView);
```

`frame` A reference to a display frame of this part.

`storageUnitView`
A reference to the frame's `partInfo` storage.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

*return value*

**DESCRIPTION**

Called by the frame object. You must override this method.

The data for the `partInfo` is stored in a value in the frame's storage unit, specified by the `suView` parameter. It gets the data from out of the value, and places it in a block of memory. It then returns the memory block to the frame for it to hold.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrInvalidFrame` | The specified frame is not a display frame of this part. |
| `kXMPErrOutOfMemory` | Not enough memory to allocate partInfo. |

## RedoAction

The `RedoAction` method tells the part to Redo the action described by `actionState`.

```
void RedoAction (XMPActionData actionState);
```

`actionState`
Information previously logged by the part to allow it to redo the action.

**DESCRIPTION**

Called by the undo object. You must override this method.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrInvalidUndo` | Not an undo action of this part. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## RemoveDisplayFrame

The `RemoveDisplayFrame` method removes the frame from the list of display frames.

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

`frame` A reference to the display frame of this part.

**DESCRIPTION**

Called by `Frame::Remove()`. You must override this method.

This method makes whatever other adjustments are necessary to deal with removing one of the presentations. This includes removing any frames embedded within this display frame.Before calling this method, you must ensure that the frame has no facets.

**EXCEPTIONS**

`kXMPErrInvalidFrame` The specified frame is not a display frame of this part.

## RemoveEmbeddedFrame

The `RemoveEmbeddedFrame` method removes a frame used for viewing an embedded part.

```
void RemoveEmbeddedFrame (XMPFrame* embeddedFrame);
```

`embeddedFrame`
A reference to an `embeddedFrame`

**DESCRIPTION**

Called by an embedded part. You must override this method.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

It is called by an embedded part to indicate it no longer needs the frame to display itself.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrCannotEmbed` | This part does not support embedding. |
| `kXMPErrInvalidFrame` | The specified frame is not an embedded frame of this part. |

## RequestEmbeddedFrame

The `RequestEmbeddedFrame` method asks the part to create a new frame and embed a part in it.

```
XMPFrame* RequestEmbeddedFrame
                                (XMPFrame* containingFrame,
                                 XMPFrame* baseFrame,
                                 XMPShape* frameShape,
                                 XMPPart* embedPart,
                                 XMPTypeToken viewType,
                                 XMPTypeToken presentation,
                                 XMPBoolean isOverlaid);
```

| | |
|---|---|
| `containingFrame` | The display frame of this part in which to embed the new frame. |
| `baseFrame` | A sibling frame of the frame to be created, also a display frame of the embedded Part. |
| `frameShape` | The requested shape of the new frame. In the frame's own coordinate space. |
| `embedPart` | The part to embed in the new frame. |
| `viewType` | View type for new frame. |
| `presentation` | Presentation for new frame. |
| `isOverlaid` | Whether the new frame should float above the part's content. |
| *return value* | The new frame. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

This is only requested by embedded parts which want additional frames in which to display in the same containing part.

If this part is a containing part, it should ask the draft to create a new frame, and embed the frame in its content. “baseFrame” is another display frame of the same part, already embedded in the same containing part. The frameShape parameter is relative to the frame coordinate system of the baseFrame. The containing part decides if it can fulfill the requests for frameShape. - if not, it can create the frame where it wants and with the shape it chooses. The containing part should assign the new frame to the same frameGroup as the baseFrame. 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().

**EXCEPTIONS**

`kXMPErrCannotEmbed`
: This part does not support embedding.

`kXMPErrOutOfMemory`
: Not enough memory to embed a part.

## RequestFrameShape

The `RequestFrameShape` method is used by an embedded part to ask for a change in the shape of its frame.

```
XMPShape* RequestFrameShape   (XMPFrame* embeddedFrame,
                               XMPShape* frameShape);
```

`embeddedFrame`
: A reference to an embedded frame of this part.

`frameShape` A reference to the requested shape, specified by the frame coordinates of the embedded frame.

*return value*



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Called by `Frame::RequestFrameShape()`. You must override this method.

This method decides what new shape to give the embedded frame, using the requested "frameShape" as a guideline. It then answers with what shape is actually granted as the return value. The requestor must abide by the returned shape, though it may make further requests for different shapes or additional frames.

## RevealFrame

The `RevealFrame` method asks a part to make an embedded frame visible.

```
void RevealFrame  (XMPFrame* embeddedFrame
                   XMPShape* revealShape);
```

`embeddedFrame`
A reference to an embedded frame of this part.

`revealShape`
A reference to a shape expressed in the coordinates of the `embeddedFrame`. The parameter indicates the portion of the frame that is be revealed.

**DESCRIPTION**

Called by an embedded part. You must override this method.

This method scrolls one of this part's display frame's to make the embedded frame visible therein. If this part has no visible frames, it asks 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, it opens a frame in a new window.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrInvalidFrame` | The specified frame is not an embedded frame of this part. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## RevealLink

The `RevealLink` method reveals data that was previously linked in a window, making it available for viewing in a display frame.

```
void RevealLink (XMPLinkSource* linkSource);
```

`linkSource` A reference to the linked content to reveal.

### DESCRIPTION

Called by `XMPLinkSource` objects. Should not be called by parts. You must override this method.

In some display frame for this part, this method selects the content linked by the `linkSource` argument, and scrolls it into view. That display frame is made the active frame. If no display frames for the part currently exist, or if this part's containing frame can't reveal the display frame, it opens a frame in a new window.

### EXCEPTIONS

`kXMPErrInvalidLink` This isn't one of the part's links.

## SequenceChanged

The `SequenceChanged` method informs the part that the sequencing of a group of its display frames has been changed.

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

`frame` A reference to the frame in the group whose sequence has been re-ordered.

### DESCRIPTION

Called by containing part.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

You must override this method.

The containing part of `frame` should call this when adding a new frame to the group or re-ordering the frames in the group. A single frame of the group is passed as an argument to indicate which group of which containing frame has been changed.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrInvalidFrame` | frame is not a display frame of this part. |

## UndoAction

The `UndoAction` method tells the part to undo the action that is described by `actionState`.

```
void UndoAction (XMPActionData actionState);
```

`actionState`
Information previously logged by the part that allows it to undo the action.

**DESCRIPTION**

Called by the undo object. You must override this method.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrInvalidUndo` | Not an undo action of this part. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## UsedShapeChanged

The `UsedShapeChanged` method informs a containing part that the used shape of one of its embedded frames has changed.

```
void UsedShapeChanged (XMPFrame* embeddedFrame);
```

`embeddedFrame`
A reference to an embedded frame of this part.

**DESCRIPTION**

Called by `Frame::ChangeUsedShape()`. You must override this method.

Containing parts that have arranged their content to conform to the used shape of an embedded frame (as in text wrapping) will need to adjust the layout of that content for the new used shape.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrInvalidFrame` | The specified embeddedFrame is not an `embeddedFrame` of this part. |

## ViewTypeChanged

The `ViewTypeChanged` method notifies the part that the `viewType` of one of its frames has been changed.

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

| | |
|---|---|
| `frame` | A reference to a display frame of this part. |

**DESCRIPTION**

Called by `Frame::ChangeViewType()`. You must override this method.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

The part should examine the new view type via `frame->GetViewType()`. If the part does not support that kind of view type, it should correct the frame's `viewType` using`frame->SetViewType()`. Note that parts MUST support the standard set of view types (se HI spec). The part should then adjust its display in the display frame to be of the new view type.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrInvalidFrame` | The specified frame is not a display frame of this part. |

## WriteActionState

The `WriteActionState` method writes out the `actionState` data onto a `StorageUnit` view, and externalizes the `undoAction` data to the storage unit.

```
void WriteActionState(XMPPtr actionState,
                      XMPStorageUnitView* storageUnitView);
```

`actionState`
Information previously logged by the part to allow it to redo the action.

`storageUnitView`
A reference to a `storageUnitView` location where the `actionState` data is to be externalized.

**DESCRIPTION**

Called by the undo object. You must override this method.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrInvalidUndo` | Not an undo action of this part. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## WritePartInfo

The `WritePartInfo` method externalizes the frame's `partInfo` data onto the frame's storage unit.

```
void WritePartInfo   (XMPPtr partInfo,
                      XMPStorageUnitView* storageUnitView);
```

`partInfo` The partInfo to externalize.

`storageUnitView`
A view onto a value of the frame's storage unit.

### DESCRIPTION

Called by the frame object. You must override this method.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

CHAPTER 1

# The Class XMPPart


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

CHAPTER 2

# OpenDoc Capability Classes

## Contents


XMPArbitrator 76
- CreateOwnerIterator 79
- GetFocusOwner 80
- IsFocusExclusive 80
- IsFocusRegistered 81
- Purge 81
- RegisterFocus 82
- RelinquishFocus 83
- RelinquishFocusSet 84
- RequestFocus 85
- RequestFocusSet 86
- TransferFocus 87
- TransferFocusSet 88
- UnregisterFocus 89

XMPCanvas 90
- XMPCanvas 92
- ~XMPCanvas 92
- GetFacet 93
- GetGraphicsSystem 93
- GetOwner 94
- GetPlatformCanvas 94
- GetUpdateShape 95
- InitCanvas 95
- Invalidate 96
- IsDynamic 96
- IsOffscreen 97





Draft. Preliminary. © Apple Computer, Inc. 6/20/94
Tucson 1.0–Chapter TOC Seq

Purge 97
ResetUpdateShape 98
SetFacet 98
SetOwner 99
Validate 99
XMPClipboard 100
Clear 101
ExportClipboard 102
GetChangeID 103
GetContentStorageUnit 103
Lock 104
SetPlatformClipboard 105
Unlock 106
XMPContainer 108
XMPContainer 110
~XMPContainer 110
Close 111
Create 111
GetDocument 112
GetID 113
GetName 113
GetStorageSystem 114
IncrementRefCount 114
InitContainer 115
Open 115
Purge 116
Release 117
ReleaseDocument 117
SetName 118
XMPDispatcher 119
AddDispatchModule 120
AddMonitor 121
Dispatch 122
Purge 123
RemoveDispatchModule 123
RemoveMonitor 124
ShouldTerminateEventLoop 124



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

TerminateEventLoop 125
XMPDocument 126
XMPDocument 128
~XMPDocument 128
CollapseDrafts 129
CreateDraft 130
GetBaseDraft 131
GetContainer 132
GetDocumentProperties 132
GetDraft 133
GetID 135
GetName 136
IncrementRefCount 136
InitDocument 137
Purge 137
Release 138
ReleaseDraft 138
SaveToAPrevDraft 139
SetBaseDraftFromForeignDraft 140
SetName 141
XMPDraft 142
XMPDraft 145
~XMPDraft 146
AbortClone 146
BeginClone 147
ChangedFromPrev 147
CreateFrame 148
CreateLinkSource 149
CreateLinkSpec 150
CreatePart 151
CreateStorageUnit 151
EndClone 152
Externalize 153
GetDocument 153
GetDraftProperties 154
GetFrame 154
GetID 155



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

GetLink 155
GetLinkSource 156
GetName 157
GetPart 157
GetPermissions 158
GetStorageUnit 159
IncrementRefCount 159
InitDraft 160
Purge 160
Release 161
ReleaseFrame 161
ReleaseLink 162
ReleasePart 163
ReleaseStorageUnit 163
RemoveChanges 164
RemoveFrame 165
RemoveFromDocument 166
RemoveLink 166
RemoveLinkSource 167
RemovePart 168
RemoveStorageUnit 169
SaveToAPrevious 170
SetChangedFromPrev 170
SetName 171
XMPDragAndDrop 172
XMPDragAndDrop 173
~XMPDragAndDrop 174
Clear 174
GetContentStorageUnit 175
InitDragAndDrop 175
Purge 176
StartDrag 176
XMPDragItemIterator 178
First 179
IsNotComplete 179
Next 180
XMPEmbeddedFramesIterator 181



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

First 182
IsNotComplete 182
Next 182
XMPFacet 184
XMPFacet 189
~XMPFacet 189
ActiveBorderContainsPoint 189
ChangeActiveShape 190
ChangeClipShape 190
ChangeExternalTransform 191
ChangeGeometry 191
ChangeHighlight 192
ContainsPoint 192
CreateEmbeddedFacet 192
CreateFacetIterator 194
Draw 194
DrawActiveBorder 195
DrawChildren 195
DrawChildrenAlways 195
DrawnIn 196
GetActiveShape 196
GetAggregateClipShape 197
GetCanvas 197
GetClipShape 197
GetContainingFacet 198
GetContentTransform 198
GetExternalTransform 199
GetFrame 199
GetFrameTransform 199
GetHighlight 200
GetPartInfo 200
GetWindow 200
GetWindowAggregateClipShape 201
GetWindowContentTransform 201
GetWindowFrameTransform 201
HasCanvas 202
InitFacet 202



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Invalidate 203
InvalidateActiveBorder 203
IsSelected 204
MoveBefore 204
MoveBehind 205
RemoveFacet 205
SetCanvas 206
SetPartInfo 207
SetSelected 207
SetWindow 207
Update 208
Validate 208
XMPFacetIterator 209
XMPFacetIterator 210
~XMPFacetIterator 210
First 211
IsNotComplete 211
Next 212
SkipChildren 212
XMPFocusOwnerIterator 213
XMPFocusOwnerIterator 214
~XMPFocusOwnerIterator 214
First 215
InitFocusOwnerIterator 215
IsNotComplete 216
Next 216
XMPFocusSet 218
XMPFocusSet 219
~XMPFocusSet 219
Add 220
Contains 220
CreateIterator 221
InitFocusSet 221
Remove 222
XMPFocusSetIterator 223
~XMPFocusSetIterator 224
First 224



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

IsNotComplete 225
Next 225
XMPFrame 226
ChangeFrameShape 231
ChangeInternalTransform 232
ChangeLinkStatus 232
ChangePart 233
ChangePresentation 233
ChangeUsedShape 234
ChangeViewType 234
CloneTo 235
Close 235
ContentChanged 236
CreateFacetIterator 236
DoesPropagateEvents 237
DrawActiveBorder 237
Externalize 237
FacetAdded 238
FacetRemoved 238
GetContainingFrame 238
GetFrameGroup 239
GetFrameShape 239
GetInternalTransform 239
GetLinkStatus 240
GetPart 240
GetPartInfo 240
GetPresentation 241
GetSequenceNumber 241
GetUsedShape 241
GetViewType 242
Invalidate 242
InvalidateActiveBorder 242
IsDragging 243
IsDroppable 243
IsFrozen 243
IsOverlaid 244
IsRoot 244



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

IsSubframe 244
Purge 245
Release 245
Remove 245
RequestFrameShape 246
SetContainingFrame 246
SetDragging 247
SetDroppable 247
SetFrameGroup 247
SetFrozen 248
SetPartInfo 248
SetPresentation 248
SetPropagateEvents 249
SetSequenceNumber 249
SetSubframe 250
SetViewType 250
Validate 250
XMPFrameFacetIterator 251
XMPFrameFacetIterator 251
~XMPFrameFacetIterator 252
First 252
IsNotComplete 253
Next 253
XMPLink 254
CloneTo 255
Externalize 256
GetChangeID 257
GetChangeTime 257
GetContentStorageUnit 258
GetStatus 258
Lock 259
RegisterDependent 260
ShowSourceContent 260
Unlock 261
UnregisterDependent 262
XMPLinkManager 263
AnyLinkImported 265



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

DraftClosing 265
DraftOpened 266
DraftSaved 266
UnsavedExportedLinks 267
XMPLinkSource 268
Clear 270
CloneTo 271
ContentChanged 272
Externalize 273
GetChangeID 273
GetContentStorageUnit 274
GetLink 274
IsAutoUpdate 275
Lock 275
SetAutoUpdate 276
SetSourcePart 276
Unlock 277
XMPLinkSpec 279
XMPLinkSpec 280
~XMPLinkSpec 281
ReadLinkSpec 281
WriteLinkSpec 282
XMPMenuBar 284
XMPMenuBar 285
~XMPMenuBar 286
AddMenuBefore 286
AddMenuLast 287
Copy 288
Display 288
GetMenu 289
InitMenuBar 289
Purge 290
Release 290
RemoveMenu 291
XMPMessageInterface 292
CreateEvent 293
CreatePartAddrDesc 294



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

CreatePartObjSpec 295
ProcessSemanticEvent 296
Purge 296
Send 297
XMPNameResolver 299
CallObjectAccessor 300
CreateSwapToken 301
CreateToken 302
DisposeToken 302
GetContainingFrame 303
GetContextInfo 304
Purge 305
Resolve 305
XMPNameSpace 307
Exists 308
GetName 308
GetValue 309
Purge 309
ReadFromFile 310
ReadFromStorage 310
Register 311
Unregister 312
WriteToFile 313
WriteToStorage 313
XMPNameSpaceManager 315
CreateNameSpace 316
DeleteNameSpace 317
HasNameSpace 317
Purge 318
XMPObject 319
XMPObject 320
~XMPObject 320
GetExtension 321
HasExtension 321
InitObject 322
Purge 323
ReleaseExtension 323



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

XMPPersistentObject 325
XMPPersistentObject 326
~XMPPersistentObject 327
Externalize 327
GetID 328
GetStorageUnit 328
InitPersistentObject 329
InitPersistentObjectFromStorage 329
XMPPlatformTypeList 331
XMPPlatformTypeList 332
~XMPPlatformTypeList 333
Add 333
AddLast 334
Contains 335
Count 335
CreatePlatformTypeListIterator 336
Difference 336
InitPlatformTypeList 337
Remove 338
Union 338
XMPPlatformTypeListIterator 340
~XMPPlatformTypeListIterator 341
First 341
IsNotComplete 342
Next 342
XMPRefCntObject 344
XMPRefCntObject 345
~XMPRefCntObject 345
IncrementRefCount 346
InitRefCntObject 346
Release 347
XMPSemanticInterface 348
XMPSemanticInterface 350
~XMPSemanticInterface 351
GetCoercionHandler 351
GetEventHandler 352
GetObjectAccessor 353



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

GetOSLSupportFlags 354
GetSpecialHandler 355
InitSemanticInterface 355
InstallAdjustMarksProc 356
InstallCoercionHandler 357
InstallCompareProc 358
InstallCountProc 358
InstallDisposeTokenProc 359
InstallEventHandler 359
InstallGetErrDescProc 360
InstallGetMarkTokenProc 361
InstallMarkProc 361
InstallObjectAccessor 362
InstallSpecialHandler 363
RemoveCoercionHandler 364
RemoveEventHandler 365
RemoveObjectAccessor 366
RemoveSpecialHandler 367
SetOSLSupportFlags 367
XMPSession 369
Close 370
Exists 371
GetArbitrator 371
GetClipboard 372
GetDispatcher 372
GetDragAndDrop 373
GetLinkManager 373
GetMessageInterface 373
GetNameResolver 374
GetNameSpaceManager 374
GetSemanticInterface 375
GetShellSemtInterface 375
GetStorageSystem 376
GetTranslation 376
GetType 376
GetUndo 377
GetWindowState 377



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Purge 378
RegisterObject 378
RemoveEntry 379
Tokenize 380
UniqueChangeID 380
UnregisterObject 381
XMPShape 382
XMPShape 384
~XMPShape 384
ContainsPoint 385
Copy 385
CopyFrom 386
CopyPolygon 386
GetBoundingBox 387
GetGeometryMode 388
GetPlatformShape 388
HasGeometry 389
Intersect 390
InverseTransform 391
IsEmpty 391
IsRectangular 392
IsSameAs 392
ReadShape 393
SetGeometryMode 394
SetPlatformShape 395
SetPolygon 395
SetRectangle 396
Subtract 397
Transform 397
Union 398
WriteShape 399
XMPStorageSystem 400
CreateContainer 401
GetContainer 402
GetSession 403
NeedSpace 404
ReleaseContainer 404



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

XMPStorageUnit 406
XMPStorageUnit 409
~XMPStorageUnit 410
AddProperty 410
AddValue 411
ClearAllPromises 412
CloneInto 412
CloneTo 414
CopyTo 415
CountProperties 415
CountValues 416
CreateCursor 416
CreateView 417
DeleteValue 418
Exists 419
Exists 420
Externalize 420
Focus 421
Focus 422
GetDraft 423
GetGenerationNumber 424
GetID 424
GetIDFromStorageUnitRef 425
GetName 425
GetOffset 426
GetPromiseValue 427
GetProperty 428
GetSession 429
GetSize 429
GetStorageUnitRefIterator 430
GetStrongStorageUnitRef 430
GetType 431
GetValue 432
GetWeakStorageUnitRef 433
IncrementGenerationNumber 433
IncrementRefCount 434
InitStorageUnit 434



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

InsertValue 435
Internalize 436
IsPromiseValue 436
IsStrongStorageUnitRef 437
IsWeakStorageUnitRef 437
Lock 438
Purge 439
Release 439
Remove 440
RemoveStorageUnitRef 440
SetName 441
SetOffset 441
SetPromiseValue 442
SetType 443
SetValue 444
XMPStorageUnitCursor 445
XMPStorageUnitCursor 446
~XMPStorageUnitCursor 446
GetCursor 446
InitStorageUnitCursor 447
XMPStorageUnitRefIterator 449
XMPStorageUnitRefIterator 450
~XMPStorageUnitRefIterator 450
First 451
InitStorageUnitRefIterator 451
IsNotComplete 452
Next 452
XMPStorageUnitView 453
XMPStorageUnitView 455
~XMPStorageUnitView 456
AddProperty 456
AddValue 457
CloneInto 457
CloneTo 458
CopyTo 459
DeleteValue 460
Externalize 460



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

GetCursor 461
GetGenerationNumber 461
GetID 462
GetIDFromStorageUnitRef 462
GetName 463
GetOffset 463
GetPromiseValue 464
GetProperty 465
GetSize 465
GetStorageUnit 466
GetStorageUnitRefIterator 466
GetStrongStorageUnitRef 467
GetType 468
GetValue 468
GetWeakStorageUnitRef 469
IncrementGenerationNumber 470
InitStorageUnitView 470
InsertValue 471
Internalize 472
IsPromiseValue 472
IsStrongStorageUnitRef 473
IsWeakStorageUnitRef 473
Purge 474
Remove 475
RemoveStorageUnitRef 475
SetName 476
SetOffset 476
SetPromiseValue 477
SetType 478
SetValue 478
XMPTransform 480
XMPTransform 482
~XMPTransform 482
ConvertTransform 483
Copy 484
CopyFrom 484
GetGraphicsSystem 485



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

GetMatrix 485
GetOffset 486
GetPlatformTransform 486
GetPreScaleOffset 487
GetScale 488
GetType 488
HasMatrix 488
Invert 489
InvertPoint 490
InvertShape 491
IsSameAs 491
MoveBy 492
PostCompose 492
PreCompose 493
ReadFrom 493
Reset 494
ScaleBy 494
ScaleDownBy 495
SetMatrix 495
SetOffset 496
SetPlatformTransform 496
TransformPoint 497
TransformShape 498
WriteTo 498
XMPTranslation 500
XMPTranslation 501
~XMPTranslation 502
GetISOTypeFromPlatformType 502
GetPlatformTypeFromISOType 503
GetTranslateMethod 503
GetTranslationOf 504
InitTranslation 505
Purge 505
Translate 506
Translate 507
XMPTypeList 509
XMPTypeList 510



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

~XMPTypeList 510
Add 511
AddLast 511
Contains 512
Count 513
CreateTypeListIterator 513
InitTypeList 514
InitTypeList 515
Remove 515
XMPTypeListIterator 517
~XMPTypeListIterator 518
First 518
IsNotComplete 519
Next 519
XMPUndo 521
AddActionToHistory 523
ClearActionHistory 524
ClearRedoHistory 524
MarkActionHistory 525
PeekRedoHistory 526
PeekUndoHistory 527
Purge 527
Redo 528
Undo 529
XMPValueIterator 530
XMPValueIterator 531
~XMPValueIterator 531
First 531
IsNotComplete 532
Next 533
XMPWindow 534
Close 536
CloseAndRemove 536
GetFacetUnderPoint 537
GetID 538
GetPlatformWindow 538
GetRootFrame 539



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

GetSourceFrame 539
Hide 540
IsActive 540
IsResizable 541
IsRootWindow 541
IsShown 542
Open 542
Purge 543
Resized 543
SetShouldSave 544
SetShouldShowLinks 544
ShouldSave 545
ShouldShowLinks 545
Show 546
XMPWindowIterator 547
~XMPWindowIterator 548
First 548
IsNotComplete 549
Last 549
Next 550
Previous 550
XMPWindowState 552
AdjustPartMenus 554
CloseWindows 554
CopyBaseMenuBar 555
CreateWindowIterator 555
Externalize 556
GetActiveWindow 556
GetRootWindowCount 557
GetTotalRootWindowCount 557
GetWindow 558
GetWindowCount 559
GetXMPWindow 559
Internalize 560
IsXMPWindow 561
OpenWindows 561
Purge 562



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

SetBaseMenuBar 563
SetDefaultWindowTitles 563



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# OpenDoc Capability Classes

This chapter describes the classes that provide most of the capabilities of OpenDoc that your part handler uses. By using or instantiating the classes described here, your part can function within an OpenDoc document and can embed other parts.

Classes are presented in this chapter in alphabetical order; within each class description, methods are described in alphabetical order also. Methods used only by embedding applications or document shells, and therefore not of interest for writing a part handler, are marked as such.

**Note**
All classes and methods described in this chapter are platform-independent. For descriptions of any platform-specific methods that these classes may have, see the *OpenDoc Cookbook* for the platform in question. [For the Macintosh platform, this information is currently in a separate document entitled *OpenDoc Class Reference: Macintosh Supplement.* ] ◆



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPArbitrator

*Superclasses* `XMPObject`
*Subclasses* none

The `XMPArbitrator` class manages the competition between parts for shared global resources. Each of these resources is referred to as a **focus**.

## Description

When a frame or window is selected by an event such as a mouse click, the part owning that frame is activated. The part handler that manages the behavior of that frame then requests ownership of a set of shared global resources, called **foci**, which manage the behavior of a selected part, controlling the operation of events within its boundaries. These events typically include mouse clicks, keystrokes, menu activations and selections, and serial port operation.

Foci are indexed according to the type of resource that each represents, each **focus type** managing one kind of event. Example focus types include keystroke focus, menu bar focus, and selection focus`XMPArbitrator` directs keyboard events to the part that currently owns the keystroke focus, and menu events to the part that currently owns the menu bar focus. Many foci are related to event distribution within an OpenDoc document, such as menu bar events, keyboard events, and mouse clicks. Other foci can be associated with system-wide resources like ports.

Foci are grouped by focus type into a **focus set**. The ownership of the foci in the set is managed by an instance of `XMPArbitrator`. Foci are of the type `XMPFocusType` (ISO strings), but most arbitration methods use tokenized forms of these strings, of type `XMPTypeToken`. The `XMPFocusSet` class is used to store these sets of foci. Individual foci are grouped by their owner frame in **focus modules**These are instances of the class `XMPFocusModule`. `XMPArbitrator` maintains a table of these focus modules that is indexed by focus type. In addition, there is a standard focus module that arbitrates standard foci like keystroke focus and menu bar focus.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

When the part is later deactivated, it relinquishes ownership of this same focus set. During the time when a focus set is owned by a frame, the foci designate the part owning that frame as the target for input from the resources that they represent. During this time, they are said to be registered foci of that part. What may be considered the active part is actually the one that owns the selection focus. The part that owns the selection focus is the part whose frame has an active border drawn around it.

In most cases, the same part owns both menu bar and keystroke foci, but it is possible for one part to receive menu commands while another receives keystrokes. This may be useful in situations such as when a part manages a modal dialog box that is constructed from control parts with which it possesses a more intimate relationship than usually exists between container parts and embedded parts. Such an embedded part can manage content objects such as buttons, scroll boxes, or editable text fields. When one of these modal dialog parts is selected, it will request ownership of a set of specific foci. It requests a set of foci that typically includes button, keyboard, and menu foci, while a part controlling a radio button that is embedded within that modal dialog box will only request ownership of a subset of that set of foci; in this case, the radio button focus. This button, in turn, shares ownership of the radio button focus with any other radio buttons that are concurrently active within that modal dialog box. If this same dialog box contains an embedded editable-text-field part, the keystroke focus ownership is requested by that part when an event such as a mouseclick activates it and gives it the selection focus.

While most foci are exclusive, and can be owned by only one part at a time, this class also supports non-exclusive foci that can be owned by several parts. In such cases, the arbitrator merely provides a central location in which you can record and later obtain a list of the owners for that focus. Exclusivity is determined when registering newly defined foci in a focus module, by using the `IsFocusExclusive` method provided by instances of `XMPFocusModule`.

`XMPArbitrator` can be extended to accomodate the addition of exotic input devices and the parts that use them. You accomplish this by instantiating additional `XMPFocusModule` objects to handle the new focus types.

`XMPArbitrator` is instantiated by `XMPSession` when the `OpenXMPSession` method is called. The Tokenize method of `XMPSession` is used for tokenizing foci of type `XMPFocusType`. The foci are then returned as unique instances of `XMPTypeToken`, and as such, are stored in `XMPFocusSet` objects.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

For more information on foci, focus sets, focus types, and focus modules, see the class descriptions for `XMPFocusModule`, `XMPFocusOwnerIterator`, `XMPFocusSet`, and `XMPFocusSetIterator`.

## Methods

`XMPArbitrator` is involved in Part Activation. In general, its methods are used in the registration and transference of focus ownership.

#### Focus Ownership

| | |
|---|---|
| `CreateOwnerIterator` | Returns a focus-owner iterator that iterates over the parts that own the specified non-exclusive focus. |
| `GetFocusOwner` | Returns the frame that owns the specified focus. |
| `IsFocusExclusive` | Returns a Boolean specifying whether ownership of the specified focus is exclusive to one part or not. |

#### Change of Focus Ownership

| | |
|---|---|
| `RelinquishFocus` | Requests that the current owner of the frame relinquish ownership of the specified focus. |
| `RelinquishFocusSet` | Requests that the current owner of the frame relinquish ownership of the specified focus set. |
| `RequestFocus` | Requests that the ownership of the specified focus be assigned to the specified frame, provided that the existing owner is willing to relinquish the requested focus. |
| `RequestFocusSet` | Requests that the ownership of each focus in the specified focus set be assigned to the specified frame, provided that all existing owners are willing to relinquish the requested foci. |
| `TransferFocus` | Transfers a focus from its current owner to another. |
| `TransferFocusSet` | Sets the owner frame of each focus in the specified focus set to`newFrame`. |

#### Focus Registration

| | |
|---|---|
| `IsFocusRegistered` | Returns a Boolean specifying whether the focus is registered to a part or not. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `RegisterFocus` | Installs a new focus module that is used to manage the specified focus. |
| `UnregisterFocus` | Removes the specified focus module from registration by the arbitrator. |

**Memory**

| | |
|---|---|
| `Purge` | Frees up some memory by removing cached information that can be easily recomputed from persistent storage. |

Detailed descriptions of these methods follow, in alphabetical order.

## CreateOwnerIterator

The `CreateOwnerIterator` method returns a focus-owner iterator that iterates over the parts that own the specified non-exclusive focus.

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

| | |
|---|---|
| `focus` | A tokenized string of type `XMPTypeToken`, representing a registered focus type. |
| *return value* | An iterator that specifies the current owner of the specified focus. |

### DESCRIPTION

A part handler call this `XMPFocusModule` method when it is determined that the focus is non-exclusive and has more than one owner. It creates an iterator that enumerates the current owners of the specified focus. If the focus is exclusive, ownership of the specified focus is not relinquished.

Before calling this method, you must ensure that the specified token represents a valid, non-exclusive focus. After it executes successfully, there is no change to the internal state of `CreateOwnerIterator`.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

`kXMPErrOutOfMemory` Insufficient memory to call method.

## GetFocusOwner

The `GetFocusOwner` method returns the frame that owns the specified focus.

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

`focusType` A tokenized string of type `XMPTypeToken`, representing a registered focus type.

*return value* The frame that owns the focus. This method returns `kXMPNull` if there are no owners. If there are owners and the focus is non-exclusive, it returns `kXMPErrFocusIsExclusive`.

**DESCRIPTION**

Before calling this method, you must ensure that the specified focus is a constructed and initialized object of this class.

**EXCEPTIONS**

`kXMPErrFocusIsExclusive` The ownership of this focus is exclusive.

## IsFocusExclusive

The `IsFocusExclusive` method returns a Boolean specifying whether ownership of the specified focus is exclusive to one part or not.

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

`focus` A tokenized string of type `XMPTypeToken`, representing a registered focus type.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

*return value* `kXMPTrue` if the focus is exclusive, `kXMPFalse` if it is not.

**DESCRIPTION**

Before calling this method, you must ensure that the specified focus is a constructed and initialized object of this class.

## IsFocusRegistered

The `IsFocusRegistered` method returns a Boolean specifying whether the focus is registered to a part or not.

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

`focus` A tokenized string of type `XMPTypeToken`, representing a registered focus type.

*return value* `kXMPTrue` if the focus is registered, `kXMPFalse` if it is not.

**DESCRIPTION**

Before calling this method, you must ensure that the specified focus is a constructed and initialized object of this class.

## Purge

(For embedding application or document shell only.)
The `Purge` method frees up some memory by removing cached information that can be easily recomputed from persistent storage.

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

`size` The amount of memory requested.

*return value* The amount of memory freed.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

This method is called by OpenDoc in low memory situations. Under these conditions, the `XMPDraft` object requests all the parts that are stored in draft to purge their storage fields, and rely on persistent storage when they need to recompute.

Before calling this method, you must ensure that the specified object is a valid initialized instance.

## RegisterFocus

The `RegisterFocus` method installs a new focus module that is used to manage the specified focus.

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

`focus` A tokenized string of type `XMPTypeToken`, representing a registered focus type. Specifies the focus that is to to be managed by the specified focus module.

`focusModule`
A pointer to the focus module that is associated with the specified focus.

**DESCRIPTION**

Before calling this method, you must ensure that both the specified focus and the specified focus module are valid initialized objects of their respective classes. After this method is executed successfully, the table, indexed by focus type, which `XMPArbitrator` maintains, contains the new focus module.

**EXCEPTIONS**

`kXMPErrExistingFocusModule`
A focus module already exists for that focus.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## RelinquishFocus

The `RelinquishFocus` method requests that the current owner of the frame relinquish ownership of the specified focus.

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

`focus` A tokenized string of type `XMPTypeToken`, representing the focus that is to be relinquished.

`relinquishingFrame`
A pointer to the frame that is relinquishing ownership of the focus.

### DESCRIPTION

If the current focus owner is not the exclusive owner of this focus, then this method sets the focus owner field in the table that associates foci with focus modules to `kXMPNull`.

Before calling this method, you must ensure that both the specified focus and the relinquishing frame are valid initialized objects of their respective classes. After this method is executed successfully, the focus has no owner, and the focus owner field in the table that associates foci with focus modules is set to `kXMPNull`.

### EXCEPTIONS

| | |
|---|---|
| `kXMPErrInvalidFrame` | The relinquishing frame is not a valid, initialized object of its class. |
| `kXMPErrInvalidFocus` | The specifiedfocus is not a valid, initialized object of its class. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## RelinquishFocusSet

The `RelinquishFocusSet` method requests that the current owner of the frame relinquish ownership of the specified focus set.

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

`focusSet` A tokenized string of type `XMPTypeToken`, providing a pointer to the set of foci that are to be relinquished. Foci are grouped by resource type into what are called focus sets.

`relinquishingFrame`
A pointer to the frame that is relinquishing ownership of the set of foci.

### DESCRIPTION

If the members of the current focus set are not exclusive to their owner frame, then this method sets the focus owner fields in the table that associates these foci with their focus modules to `kXMPNull`.

Before calling this method, you must ensure that both the specified focus set and the relinquishing frame are all valid initialized objects of their respective classes.

### EXCEPTIONS

`kXMPErrInvalidFocusSet`
The members of the specified focus set are not valid, initialized objects of their respective classes.

`kXMPErrInvalidFrame` The relinquishing frame is not a valid, initialized object of its class.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## RequestFocus

The `RequestFocus` method requests that the ownership of the specified focus be assigned to the specified frame.

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

`focus` A tokenized string of type `XMPTypeToken` specifying the focus whose ownership reassignment is being requested.

`requestingFrame`
A pointer to the frame that is requesting the focus.

*return value* `kXMPTrue`, if the request is granted, `kXMPFalse` if it is not.

### DESCRIPTION

This method returns a Boolean specifying whether the request to relinquish the requested focus was granted. If the focus is determined to be non-exclusive then focus is obtained. Part editors can call this method when a single focus is requested; when multiple foci are requested, the `RequestFocusSet` method is generally called.

Before calling this method, you must ensure that both the specified focus and the requesting frame are valid initialized objects of their respective classes. After this method is executed successfully, and 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 remains intact, and `kXMPFalse` is returned.

### EXCEPTIONS

`kXMPErrFocusNotRegistered`
One of the requested foci is not registered.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## RequestFocusSet

The `RequestFocusSet` method requests that the ownership of each focus in the specified focus set be assigned to the specified frame, provided that all existing owners are willing to relinquish the requested foci.

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

`focusSet` A tokenized string of type `XMPTypeToken`, providing a pointer to the set of foci that are being requested. Foci are grouped by resource type into what are called focus sets.

`requestingFrame`
A pointer to the frame that is requesting the focus.

*return value* `kXMPTrue`, if the request was granted, `kXMPFalse` if it is not.

### DESCRIPTION

If some of the foci are non-exclusive, and all are attainable, then the request is granted. However, the operation is atomic. If one focus is unattainable then ownership of the set is not granted.

Before calling this method, you must ensure that both the specified focus set and the requesting frame are all valid initialized objects of their respective classes. After this method is executed successfully, and the request is granted, the new ownership relationships is stored in the relevant focus modules, and `kXMPTrue` is returned.

If the request fails, the existing ownership relationships remain intact, and `kXMPFalse` is returned.

### EXCEPTIONS

`kXMPErrFocusNotRegistered`
One of the requested foci is not registered.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## TransferFocus

The `TransferFocus` method transfers a focus from its current owner to another.

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

`focus` A tokenized string of type `XMPTypeToken`, representing the focus that is to be transferred.

`transferringFrame`
A pointer to the frame that is transferring ownership of the focus. It need not be the current owner.

`newOwner` A pointer to the new focus owner.

### DESCRIPTION

This method is called after a focus has been relinquished. Part editors can call this method; for instance, in the management of modal foci of the type that typically occur in dialog boxes. Only one part can bring up a modal dialog at one time. So, in a situation where there are nested modal dialog parts, this method both transfers ownership of foci to the innermost dialog part, such as a radio button, and then transfers this focus ownership back to its previous owner, in this case, the container dialog part.

Before calling this method, you must ensure that the specified focus is registered in the table maintained by `XMPArbitrator` that associates foci with focus modules. After this method is executed successfully, the focus is owned by `newOwner`.

### EXCEPTIONS

`kXMPErrInvalidFrame` The transferring frame is not a valid, initialized object of its class.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**SEE ALSO**

FocusLost(), FocusAcquired()

## TransferFocusSet

The `TransferFocusSet` method sets the owner frame of each focus in the specified focus set to `newFrame`.

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

`focusSet` A tokenized string of type `XMPTypeToken`, providing a pointer to the set of foci that are being requested. Foci are grouped by resource type into what are called focus sets.

`transferringFrame`
A pointer to the frame that is transferring ownership of the set of foci.

`newOwner` A pointer to the new frame that owns the focus set.

**DESCRIPTION**

This method is called after a set of foci have been relinquished.Before calling this method, you must ensure that the specified foci are registered in the table maintained by `XMPArbitrator` that associates foci with focus modules. After this method is executed successfully, the foci is owned by `newFrame`.

**EXCEPTIONS**

`kXMPErrInvalidFocusSet`
The members of the specified focus set are not valid, initialized objects of their respective classes.

`kXMPErrInvalidFrame` The transferring frame is not a valid, initialized object of its class.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**SEE ALSO**

FocusLost(), FocusAcquired()

## UnregisterFocus

The `UnregisterFocus` method removes the specified focus module from registration by the arbitrator.

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

`focus` A tokenized string of type `XMPTypeToken` specifying the focus that is to be removed.

**DESCRIPTION**

Before calling this method, you must ensure that both the specified focus and the specified focus module are valid initialized objects of their respective classes. After this method is executed successfully, the specified focus module is not in the table maintained by `XMPArbitrator` that associates foci with focus modules.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPCanvas

*Superclasses* `XMPObject`

*Subclasses* none

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

## Description

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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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.

Invariants: 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.

## Methods

```
XMPCanvas
~XMPCanvas
GetFacet
GetGraphicsSystem
GetOwner
GetPlatformCanvas
GetUpdateShape
InitCanvas
Invalidate
IsDynamic
IsOffscreen
Purge
ResetUpdateShape
SetFacet
SetOwner
Validate
```

Detailed descriptions of these methods follow, in alphabetical order.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## XMPCanvas

Class constructor.

```
XMPCanvas();
```

*return value* an XMPCanvas object.

**DESCRIPTION**

Imaging. Public No restrictions.

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

**EXCEPTIONS**

`kXMPErrInvalidPlatformCanvas`

`kXMPOutOfMemory`
cannot allocate XMPCanvas object.

## ~XMPCanvas

Class destructor. Delete internal data structures.

```
~XMPCanvas();
```

**DESCRIPTION**

Imaging. Public No restrictions.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetFacet

Return the facet this canvas is associated with.

```
XMPFacet* GetFacet();
```

*return value* This canvas' facet.

**DESCRIPTION**

None. Public

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

## GetGraphicsSystem

Return the code indicating the graphics system of this canvas.

```
XMPGraphicsSystem GetGraphicsSystem();
```

*return value* Graphics system code. Potential values are platform-dependent.

**DESCRIPTION**

None. Public

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetOwner

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

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

*return value* The owner part.

**DESCRIPTION**

None. Public

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

## GetPlatformCanvas

Return the platform canvas of this canvas.

```
XMPPlatformCanvas GetPlatformCanvas();
```

*return value* Pointer to the platform-specific canvas.

**DESCRIPTION**

None. Public

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

**EXCEPTIONS**

`kXMPErrInvalidGrafPort`
on the Mac



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetUpdateShape

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

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

*return value* The update shape of this canvas.

**DESCRIPTION**

None. Public

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

## InitCanvas

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

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

`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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

None. Public

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

Before calling this method, you must ensure that the canvas was just constructed. After this method executes successfully, the canvas can be used for imaging.

## Invalidate

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

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

`shape` The area to add to the update shape.

**DESCRIPTION**

None. Public

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

## IsDynamic

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

```
XMPBoolean IsDynamic();
```

*return value* `kXMPTrue` if this canvas is dynamic, `kXMPFalse` if static.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

None. Public

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

## IsOffscreen

Returns whether this canvas is offscreen.

```
XMPBoolean IsOffscreen();
```

*return value* `kXMPTrue` if this canvas is offscreen, `kXMPFalse` otherwise.

**DESCRIPTION**

None. Public No restrictions.

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

## Purge

Attempt to free up to "size" bytes of memory by releasing any unneeded internal storage.

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

`size` The number of bytes to attempt to free.

**DESCRIPTION**

None. Public

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## ResetUpdateShape

```
void ResetUpdateShape();
```

**DESCRIPTION**

None. Public

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

Set the update shape to be the empty shape.

## SetFacet

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

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

`facet` The facet for the canvas.

**DESCRIPTION**

None. Public

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

### Preconditions

This canvas should be the canvas for "facet".



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## SetOwner

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

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

`owner` The new owner part.

**DESCRIPTION**

None. Public

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

## Validate

Subtract “shape” from this canvas' update shape.

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

`shape` The area to subtract from the update shape.

**DESCRIPTION**

None. Public

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPClipboard

*Superclasses* `XMPObject`

*Subclasses* none

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

## Description

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.)

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: 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.)



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

The XMPStorageUnit returned is guaranteed to contain the content of the system clipboard.

## Methods

```
Clear
GetChangeID
GetContentStorageUnit
Lock
SetPlatformClipboard
Unlock
```

Detailed descriptions of these methods follow, in alphabetical order.

### Clear

This function removes all the user data stored in the XMPClipboard object and clears the system clipboard.

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

`key` A XMPClipboardKey value returned by Lock.

**DESCRIPTION**

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

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

The client must not be holding a storage unit reference returned by a prior call to GetContentStorageUnit.

Before calling this method, you must ensure that this clipboard object has been initialized. After this method executes successfully, the next call to GetContentStorageUnit returns an XMPStorageUnit with no properties.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

`<platform errors>`
Host platform errors.

`kXMPErrInvalidClipboardKey`
The argument clipboard key is not valid.

## ExportClipboard

(For embedding application or document shell only.)
If this clipboard has been changed since the last change to the host platform clipboard, update the host clipboard now.

```
void ExportClipboard (XMPClipboardKey key);
```

`key` A XMPClipboardKey value returned by Lock.

**DESCRIPTION**

Data Interchange

Protected. This method is for internal use by platform implementations of OpenDoc. It should not be called by parts.

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

After this method executes successfully, The host platform clipboard contains the most recently cut or copied items.

**EXCEPTIONS**

`kXMPErrOutOfMemory`
Out of memory.

`<platform errors>`
Host platform errors.

`kXMPErrInvalidClipboardKey`
The argument clipboard key is not valid.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetChangeID

Returns the change id identifying the clipboard content.

```
XMPChangeID GetChangeID();
```

*return value* the change identification of the current clipboard content.

**DESCRIPTION**

Data Interchange. Public

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

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.

## GetContentStorageUnit

This function returns a XMPStorageUnit object.

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

`key` A XMPClipboardKey value returned by Lock.

*return value* An XMPStorageUnit containing the content of the clipboard. This object should not be released.

**DESCRIPTION**

Data Interchange, Storage. 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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

The clipboard lock should be acquired by caller prior to calling this method, and the lock should be relinquished after the data transfer.

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

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.

Before calling this method, you must ensure that this clipboard object has been initialized. After this method executes successfully, The returned XMPStorageUnit contains the content of the system clipboard.

**EXCEPTIONS**

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

## Lock

Acquire exclusive access to the content storage unit of the clipboard for the current thread.

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

| | |
|---|---|
| `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. |
| `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. |
| *return value* | `kXMPTrue` if access is granted and `kXMPFalse` if denied. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

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

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

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. After this method executes successfully, If `kXMPTrue` is returned, the current thread has exclusive access to the clipboard, and the key parameter is valid.

## SetPlatformClipboard

If this OpenDoc clipboard has changed since the last change to the host platform clipboard, put data on the host clipboard.

```
void SetPlatformClipboard  (XMPPlatformTypeList* typeList,
                            XMPClipboardKey key);
```

`typeList` The list of platform types to deposit on the host clipboard, if present on this clipboard.

`key` A non-zero XMPClipboardKey value returned by Lock.

**DESCRIPTION**

Data Interchange. Public

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

The data transferred is specified by the typeList argument, which is a set of platform-dependent types. Data is placed on the platform clipboard in the



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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 typeList 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.

Before calling this method, you must ensure that this clipboard object has been initialized. After this method executes successfully, any of the argument platform types from the last cut or copy operation are present on the host clipboard.

#### EXCEPTIONS

`kXMPErrOutOfMemory`
Out of Memory

`kXMPErrInvalidClipboardKey`
`The argument clipboard key is not valid.`

## Unlock

Relinquish a previously acquired clipboard lock.

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

`key` A XMPClipboardKey value returned by Lock.

#### DESCRIPTION

Thread safety. Public.

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

Before calling this method, you must ensure that the key argument is a valid clipboard key, implying exclusive access to the clipboard by this thread. After this method executes successfully, the key argument is no longer valid. The


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

current thread retains exclusive access to the clipboard if the thread already had exclusive access when the argument key was returned by Lock.

**EXCEPTIONS**

`kXMPErrInvalidClipboardKey`
The argument clipboard key is not valid.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPContainer

*Superclasses* `XMPRefCntObject` → `XMPObject`

*Subclasses* none

A Container is a physical collection of bytes, like a file or a section of memory. XMPContainer is a class used to manipulate this physical Container.

## Description

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 manipulates containers. Similarly, XMPDocument is a class used to manipulate a Document; XMPDraft manipulates 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.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

XMPStorageSystem method will then instantiate a XMPContainer object to manipuate the physical Container.

Invariants: 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.

## Methods

```
XMPContainer
~XMPContainer
Close
Create
GetDocument
GetID
GetName
GetStorageSystem
IncrementRefCount
InitContainer
Open
Purge
Release
ReleaseDocument
SetName
```

Detailed descriptions of these methods follow, in alphabetical order.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## XMPContainer

(For embedding application or document shell only.)
Constructor of the class.

```
XMPContainer();
```

*return value* a valid XMPContainer object

**DESCRIPTION**

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

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

After this method executes successfully, an un-Initialized XMPContainer object is created.

## ~XMPContainer

(For embedding application or document shell only.)
Deletes its XMPDocument object (if exists), and closes the container.

```
~XMPContainer();
```

**DESCRIPTION**

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

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

Before calling this method, you must ensure that this is a valid XMPContainer object. After this method executes successfully, the container associated with this XMPContainer object is closed.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

this is no longer a valid XMPContainer object.

## Close

(For embedding application or document shell only.)
Closes an access session to the container associated with this XMPContainer object.

```
XMPContainer* Close();
```

*return value* this

**DESCRIPTION**

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

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

Before calling this method, you must ensure that there is an open access session to the container associated with this XMPContainer object. After this method executes successfully, there is no access session to the container associated with this XMPContainer object

## Create

(For embedding application or document shell only.)
Creates the physical Container using this XMPContainer object and associates the created container with this XMPContainer object.

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

*return value* this



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

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

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

This function does NOT open the container.

After this method executes successfully, there exists a physical Container which conforms to the specification contained in this XMPContainer object.

**EXCEPTIONS**

`kXMPErrCannotCreateContainer`
Cannot create the physical Container.

## GetDocument

(For embedding application or document shell only.)
Returns a fully functional XMPDocument object associated with the given id.

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

| | |
|---|---|
| `id` | ID of the document requested |
| *return value* | A fully functional XMPDocument object. |

**DESCRIPTION**

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

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

(A fully functional object is one that has been initialized and can be used right away).

After this method executes successfully, 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.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

`kXMPErrInvalidDocument`
Cannot get desired Document because the id is invalid.

## GetID

Returns the ID of this object.

```
XMPContainerID GetID();
```

*return value* ID of the Container.

**DESCRIPTION**

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

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

This ID should be the one used in XMPAbsStorageSystem::CreateContainer or XMPAbsStorageSystem::CreateContainer.

## GetName

Returns a copy of the name of this container.

```
XMPContainerName GetName();
```

*return value* Name of the container.

**DESCRIPTION**

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

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

If the Container does not have a name, `kXMPNull` is returned.

## GetStorageSystem

Returns the XMPStorageSystem object associated with this XMPContainer object.

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

*return value* XMPStorageSystem object associated with this XMPContainer Object.

**DESCRIPTION**

Getter. Public.

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

## IncrementRefCount

Increments the reference count of the Container.

```
void  IncrementRefCount();
```

**DESCRIPTION**

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

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

After this method executes successfully, the reference count for this object is incremented by 1.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## InitContainer

(For embedding application or document shell only.)
Initializes this XMPContainer object.

```
void InitContainer();
```

**DESCRIPTION**

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

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

After this method executes successfully, this XMPContainer object has been properly initialized.

## Open

(For embedding application or document shell only.)
Opens an access session to the container associated with this XMPContainer object.

```
XMPContainer* Open();
```

*return value* this

**DESCRIPTION**

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

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

Before calling this method, you must ensure that there is no access session to the container associated with this XMPContainer object. After this method



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

executes successfully, there is an open access session to the container associated with this XMPContainer object.

**EXCEPTIONS**

`kXMPErrCannotOpenContainer`
Cannot open the physical Container.

## Purge

(For embedding application or document shell only.)
Purges memory from ephemeral store until size bytes (not necessarily contiguous) have been freed up.

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

| | |
|---|---|
| `size` | Number of bytes to purge |
| *return value* | Number of bytes actually purged |

**DESCRIPTION**

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

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

After this method executes successfully, 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.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Release

(For embedding application or document shell only.)
Decrements the ref count of this XMPContainer object.

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

*return value* fStorage

**DESCRIPTION**

Object Storage. Public.

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

After this method executes successfully, This XMPContainer object is no longer guaranteed to be a valid XMPContainer object

## ReleaseDocument

(For embedding application or document shell only.)
Releases the XMPDocument object.

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

`document` the document to be released
*return value* this XMPContainer object.

**DESCRIPTION**

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

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

Before calling this method, you must ensure that document is a valid XMPDocument object in this container which was obtained via GetDocument.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

The reference count of document is 0. After this method executes successfully, document is no longer a valid XMPDocument object.

## SetName

(For embedding application or document shell only.)
Sets the name of this container.

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

`name` the new name of this container

### DESCRIPTION

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

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

Before calling this method, you must ensure that the name is a valid XMPContainerName. After this method executes successfully, the name of the container is name.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPDispatcher

*Superclasses* `XMPObject`
*Subclasses* none

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

## Description

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.

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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Invariants: 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.

## Methods

`AddDispatchModule`
`AddMonitor`
`Dispatch`
`Purge`
`RemoveDispatchModule`
`RemoveMonitor`
`ShouldTerminateEventLoop`
`TerminateEventLoop`

Detailed descriptions of these methods follow, in alphabetical order.

### AddDispatchModule

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

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

`eventType` An event code

`dispatchModule`
A dispatch module for that event

**DESCRIPTION**

Adding and Removing Dispatch Modules. Public. Not called by most parts.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

Before calling this method, you must ensure that a valid initialize instance exists. A valid initialized dispatch module. After this method executes successfully, the dictionary contains the specified dispatch module.

#### EXCEPTIONS

`kXMPErrExistingDispatchModule`
module is already installed.

`kXMPErrInvalidDispatchModule`
specified dispatch module is invalid.

## AddMonitor

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

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

`eventType` An event code

`dispatchModule`
A dispatch module for that event

#### DESCRIPTION

Adding and Removing Dispatch Modules. Public. Not called by most parts.

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

Before calling this method, you must ensure that a valid initialize instance exists. A valid initialized dispatch module. After this method executes successfully, the dictionary contains the specified dispatch module as a monitor.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

`kXMPErrExistingDispatchModule`
module is already installed.

`kXMPErrInvalidDispatchModule`
specified dispatch module is invalid.

## Dispatch

Dispatches the event to the appropriate part.

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

`event` A platform-dependent event structure

*return value* `kXMPTrue` if the event was handled, `kXMPFalse` otherwise.

**DESCRIPTION**

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

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

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

Before calling this method, you must ensure that a valid initialized instance exists. After this method executes successfully, the result contains kXMPTrue if any part handled the event, kXMPFalse otherwise.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Purge

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

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

| | |
|---|---|
| `size` | The amount of memory requested. |
| *return value* | The amount of memory freed. |

**DESCRIPTION**

Low Memory. Public. Called by OpenDoc.

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

Before calling this method, you must ensure that a valid initialized instance exists. After this method executes successfully, some memory has been freed up.

## RemoveDispatchModule

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

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

| | |
|---|---|
| `eventType` | an event type |

**DESCRIPTION**

Adding and Removing Dispatch Modules. Public. Not called by most parts.

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

Before calling this method, you must ensure that a dispatch module is installed for the specified event type. After this method executes successfully, the specified event type no longer has its associated dispatch module.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## RemoveMonitor

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

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

`eventType` an event code

`dispatchModule`
A dispatchmodule which monitors the event type

**DESCRIPTION**

Adding and Removing Dispatch Modules. Public. Not called by most parts.

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

Before calling this method, you must ensure that the dispatch module is installed as a monitor for the specified event type. After this method executes successfully, the dispatch module is no longer a monitor for the specified event type.

## ShouldTerminateEventLoop

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

```
XMPBoolean ShouldTerminateEventLoop();
```

*return value* The value of fTerminateEventLoop

**DESCRIPTION**

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

Before calling this method, you must ensure that a valid and initialized instance exists. After this method executes successfully, the result contains the value of fTerminateEventLoop.

## TerminateEventLoop

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

```
void TerminateEventLoop();
```

**DESCRIPTION**

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

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

Before calling this method, you must ensure that a valid initialized instance exists. After this method executes successfully, fTerminateEventLoop contains the value kXMPTrue.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPDocument

*Superclasses* `XMPRefCntObject` → `XMPObject`

*Subclasses* none

A XMPDocument object is used to manipulate a Document and its Drafts.

## Description

A Container contains one or more Document. Each Document has a unique XMPDocumentID within a Container. Each of these documents 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.

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: There is at most one XMPDraft object for every XMPDraftID requested via GetDraft or CreateDraft.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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.

## Methods

```
XMPDocument
~XMPDocument
CollapseDrafts
CreateDraft
GetBaseDraft
GetContainer
GetDocumentProperties
GetDraft
GetID
GetName
IncrementRefCount
InitDocument
Purge
Release
ReleaseDraft
SaveToAPrevDraft
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

```
SetBaseDraftFromForeignDraft
SetName
```

Detailed descriptions of these methods follow, in alphabetical order.

## XMPDocument

(For embedding application or document shell only.)
Creates an uninitialized XMPDocument object.

```
XMPDocument();
```

*return value* A valid XMPDocument object.

**DESCRIPTION**

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

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

## ~XMPDocument

(For embedding application or document shell only.)
This is the destructor of the XMPDocument class object. It releases all the memory associated with this XMPDocument class object.

```
~XMPDocument();
```

**DESCRIPTION**

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


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

Before calling this method, you must ensure that this is a valid XMPDocument object.

After this method executes successfully, this is no longer a valid XMPDocument object.

## CollapseDrafts

(For embedding application or document shell only.)
This call is used for getting rid of unnecessary Drafts.

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

| | |
|---|---|
| `from` | Draft to collapse from |
| `to` | Draft to collapse to |
| *return value* | this XMPDocument object |

### DESCRIPTION

Draft Manipulation. Public.

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

Removes all the drafts between from (inclusive) and to (exclusive). All the Drafts between from (inclusive) and to (exclusive) must be empty.

Before calling this method, you must ensure that 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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**After this method executes successfully,** 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.

#### EXCEPTIONS

`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.

## CreateDraft

(For embedding application or document shell only.)
Creates a new Draft on top of below and returns a XMPDraft object referring to the created Draft.

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

`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.

*return value* a fully functional XMPDraft object with Exclusive-Write permissions.

#### DESCRIPTION

Draft Manipulation. Public.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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.

Before calling this method, you must ensure that below is the Top Draft of the Document.

If below is Exclusive-Write-only, releaseBelow must be `kXMPTrue`.

#### EXCEPTIONS

`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.

## GetBaseDraft

(For embedding application or document shell only.)
Returns an XMPDraft object which refers to the base draft of the document with the given permissions.

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

`perms` permissions

*return value* XMPDraft object which refers to the base draft of this document

#### DESCRIPTION

Draft Manipulation. Public.

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

Before calling this method, perms can be Read-Only if the Draft has not been gotten for Exclusive -Write-only.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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.

**EXCEPTIONS**

`kXMPErrInvalidPermissions`
Invalid permissions.

## GetContainer

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

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

*return value* fContainer

**DESCRIPTION**

Getter. Public.

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

## GetDocumentProperties

(For embedding application or document shell only.)
Returns a XMPStorageUnit object where meta-information on this Document can be stored.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

*return value* XMPStorageUnit object where meta-information on this Document is stored.

**DESCRIPTION**

Getter. Public.

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

Note that the client has to call XMPStorageUnit::Release when the XMPStorageUnit is not needed anymore.

**EXCEPTIONS**

`kXMPErrNoDocumentProperties`
Cannot create XMPStorageUnit for Document Properties.

## GetDraft

(For embedding application or document shell only.)
Returns an XMPDraft object referring to the desired draft .

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

`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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `release` | XMPBoolean to show whether draft should be released when getting the desired one. (Only used when relative positioning is done). |
| *return value* | a fully functional XMPDraft object |

**DESCRIPTION**

Draft Manipulation. Public.

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

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.

Before calling this method, you must ensure that 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.

After this method executes successfully, the specified draft, opened with the given perms is returned.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

If release is `kXMPTrue` and the draft is a valid XMPDraft object referring to a draft in the same document, it should be released.

**EXCEPTIONS**

`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.

## GetID

Returns the ID of the document referred to by this XMPDocument object.

```
XMPDocumentID GetID();
```

*return value* ID of the Document.

**DESCRIPTION**

Getter. Public.

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

The returned ID should be the one used for getting this object in XMPContainer::GetDocument.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetName

Returns a copy of the name of this Document.

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

*return value* Name of the Document. `kXMPNull` if the Document does not have any name.

### DESCRIPTION

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

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

If this Document does not have a name, `kXMPNull` is returned.

## IncrementRefCount

(For embedding application or document shell only.)
Increments the reference count of this XMPDocument object.

```
void IncrementRefCount();
```

### DESCRIPTION

Reference Counting. Public.

Derived class **must** override.

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

**After this method executes successfully,** Reference Count of this object is incremented by 1.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## InitDocument

(For embedding application or document shell only.)
Initializes this XMPDocument object.

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

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

**DESCRIPTION**

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

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

Before calling this method, you must ensure that this is a valid XMPDocument object.

After this method executes successfully, this is an initialized XMPDocument object.

## Purge

(For embedding application or document shell only.)
Purge memory from ephemeral store until size bytes (not necessarily contiguous) have been freed up.

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

| | |
|---|---|
| `size` | the number of bytes to purge |
| *return value* | the number of bytes actually purged |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

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

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

**After this method executes successfully,** 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

(For embedding application or document shell only.)
Releases this XMPDocument object.

```
void Release();
```

**DESCRIPTION**

Reference counting. Public.

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

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

**After this method executes successfully,** this document is no longer guaranteed to be a valid XMPDocument object

## ReleaseDraft

(For embedding application or document shell only.)
Releases the given XMPDraft object.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `draft` | the draft to be released |
| *return value* | this XMPDocument object. |

**DESCRIPTION**

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

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

Before calling this method, you must ensure that draft has a refCount of 0.

## SaveToAPrevDraft

(For embedding application or document shell only.)
Moves changes from the from one draft to another draft.

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

| | |
|---|---|
| `from` | draft to move changes from |
| `to` | draft to move changes to |

**DESCRIPTION**

Draft Manipulation. Public.

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

If to is `kXMPNull`, the changes are saved to the draft immediately below from. All the Drafts from `from` (inclusive) to `to` (exclusive) are empty after this operation.

This method can be used together with CollapseDrafts to get rid of unwanted Drafts.

Before calling this method, you must ensure that from Draft is above to Draft.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

After this method executes successfully, 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'.

#### EXCEPTIONS

`kXMPErrInvalidDraft`
from is not above to.

`kXMPErrOutstandingDraft`
There is one or more outstanding Draft between from (exclusive) and to (exclusive).

## SetBaseDraftFromForeignDraft

(For embedding application or document shell only.)
Takes a version from one Document and copies its content to set up a new Document.

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

`draft` draft to copy to base

#### DESCRIPTION

Document Manipulation; Draft Manipulation. Public.

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

Before calling this method, you must ensure that draft is not from the same Document.

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

After this method executes successfully, this Document's Base Draft contains the same data as draft.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## SetName

(For embedding application or document shell only.)
Sets the name of this document.

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

`name` New name of this Document

**DESCRIPTION**

Setter. Public.

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

Before calling this method, you must ensure that name is a valid XMPDocumentName.

After this method executes successfully, the name of the document is name.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPDraft

*Superclasses* `XMPRefCntObject` → `XMPObject`

*Subclasses* none

An OpenDoc Draft contains information pertaining to a particular version of an OpenDoc Document.

## Description

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

whater AOCE says

o memory high water mark

long

o has been 'Save'd by user

boolean

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: 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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

A Draft can have at most one name.

## Methods

`XMPDraft`
`~XMPDraft`
`AbortClone`
`BeginClone`
`ChangedFromPrev`
`CreateFrame`
`CreateLinkSource`
`CreateLinkSpec`
`CreatePart`
`CreateStorageUnit`
`EndClone`
`Externalize`
`GetDocument`
`GetDraftProperties`
`GetFrame`
`GetID`
`GetLink`
`GetLinkSource`
`GetName`
`GetPart`
`GetPermissions`
`GetStorageUnit`
`IncrementRefCount`
`InitDraft`
`Purge`
`Release`
`ReleaseFrame`
`ReleaseLink`


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

```
ReleasePart
ReleaseStorageUnit
RemoveChanges
RemoveFrame
RemoveFromDocument
RemoveLink
RemoveLinkSource
RemovePart
RemoveStorageUnit
SaveToAPrevious
SetChangedFromPrev
SetName
```

Detailed descriptions of these methods follow, in alphabetical order.

## XMPDraft

(For embedding application or document shell only.)
This function is the constructor of the class.

```
XMPDraft();
```

**DESCRIPTION**

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

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## ~XMPDraft

(For embedding application or document shell only.)
This function is the destructor of the class.

```
~XMPDraft();
```

**DESCRIPTION**

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

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

## AbortClone

Aborts the transacton started by BeginClone.

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

`key` XMPDraftKey which identifies the transaction.

**DESCRIPTION**

Object Management. Public.

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

When this method returns, none of the XMPStorageUnits in the source XMPDraft should be copied to the destination Draft.

**EXCEPTIONS**

`kXMPErrInvalidDraftKey`
Invalid Clone.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## BeginClone

Begins a transaction on a data transfer between one XMPDraft to another.

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

`kind` The kind of clone operation being performed.

*return value* Key to ensure thread safety.

### DESCRIPTION

Object Management. Public.

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

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.

### EXCEPTIONS

`kXMPErrCloningInProgress`
Another Clone has started already.

## ChangedFromPrev

(For embedding application or document shell only.)
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.

```
XMPBoolean  ChangedFromPrev();
```

*return value* whether or not this Draft contains any changes from previous Draft



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Draft Manipulation. Public.

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

Before you call this method, you must first ensure that this XMPDraft object does not refere to the base draft of the document.

**EXCEPTIONS**

`kXMPInvalidDraft`
Invalid Draft.

## CreateFrame

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

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

`containingFrame`
see XMPFrame documentation

`frameShape`

`part`

`isOverlaid`

*return value* new XMPFrame object

**DESCRIPTION**

Creation. Public.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Before you call this method, you must have exclusive-write or Shared-write permissions on this Draft. After this method executes successfully, this draft is marked dirty.

**EXCEPTIONS**

`kXMPErrCannotCreateFrame`
Cannot create XMPFrame object.

## CreateLinkSource

Create a new XMPLinkSource object in this draft.

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

| | |
|---|---|
| `part` | The part containing the source content of the link. |
| *return value* | new XMPLinkSource object |

**DESCRIPTION**

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

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

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.

Before you call this method, you must have exclusive-write or Shared-write permissions on this draft. After this method executes successfully, the returned instance is a fully functional XMPLinkSource object.

This draft is marked dirty.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

`kXMPErrCannotCreateLink`
Cannot create the XMPLinkSource or companion XMPLink object.

## CreateLinkSpec

Create a link spec for content in the argument part.

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

| | |
|---|---|
| `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. |
| *return value* | A new LinkSpec object. |

**DESCRIPTION**

Creation. Public.

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

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. After this method executes successfully, This Draft is marked dirty.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## CreatePart

(For embedding application or document shell only.)
Creates a new Part in this Draft, constructs a XMPPart to refer to the new Part and returns the XMPPart object.

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

`partType` Type of Part to create
*return value* new XMPPart object

### DESCRIPTION

Creation. Public.

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

Before you call this method, you must have exclusive-write or Shared-write permissions on this Draft. After this method executes successfully, this draft is marked dirty.

### EXCEPTIONS

`kXMPErrCannotCreatePart`
Cannot create XMPPart object.

## CreateStorageUnit

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

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

*return value* new XMPStorageUnit object



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Creation. Public.

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

Before you call this method, you must have exclusive-write or Shared-write permissions on this Draft. After this method executes successfully, this draft is marked dirty.

## EndClone

This function is used to end the transaction started by BeginClone.

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

`key` XMPDraftKey which identifies the transaction.

**DESCRIPTION**

Object Management. Public.

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

When this call returns, all the XMPStorageUnits whose CloneTo method have been called should be copied to the destination XMPDraft. After this method executes successfully, all the XMPStorageUnits whose CloneTo method have been called should be copied to the destination XMPDraft.

**EXCEPTIONS**

`kXMPErrInvalidDraftKey`



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Externalize

(For embedding application or document shell only.)
Externalizes any internal structures of this XMPDraft object

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

*return value* this XMPDraft object.

**DESCRIPTION**

Externalization. Public.

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

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

Before you call this method, you must hav exclusive-write or Shared-write permissions on this Draft. After this method executes successfully, all XMPStorageUnits and XMPPersistentObjects (and derived objects) have been asked to Externalize.

## GetDocument

(For embedding application or document shell only.)
Returns the XMPDocument object from which this XMPDraft object is created.

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

*return value* fDocument

**DESCRIPTION**

Getter. Public.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetDraftProperties

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

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

*return value* this Draft's Draft Properties Storage Unit

**DESCRIPTION**

Meta-information. Public.

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

**EXCEPTIONS**

`kXMPErrNoDraftProperties`
Draft Properties Storage Unit cannot be created.

## GetFrame

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

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

`id` ID of the desired frame
*return value* A fully functional XMPFrame object

**DESCRIPTION**

Getter. Public.

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


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Before you call this method, you must first ensure that ID refers to a valid Frame. After this method executes successfully, the return value is a fully functional XMPFrame object with a reference count >= 1.

**EXCEPTIONS**

`kXMPErrCannotGetFrame`
Cannot create XMPFrame object.

## GetID

(For embedding application or document shell only.)
Returns the ID associated with this Draft.

```
XMPDraftID  GetID();
```

*return value* ID of this Draft.

**DESCRIPTION**

Getter. Public.

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

## GetLink

Returns a XMPLink object corresponding to the specification.

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

`id` ID of the desired link

`theLinkSpec`
a link spec from which a link is to be resolved or constructed



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

*return value* a fully functional XMPLink object

**DESCRIPTION**

Getter. Public.

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

If id is not 0, linkSpec is ignored. Otherwise, linkSpec is used to retrieve the Link.

Before you call this method, you must first ensure that id or linkSpec represents a valid Link on this Draft. After this method executes successfully, the return value is a fully functional XMPLink object with a reference count >= 1.

**EXCEPTIONS**

`kXMPErrCannotGetLink`
Cannot create XMPLink object.

## GetLinkSource

Returns the XMPLinkSource object identified by the argument id.

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

`id` ID of the desired link

*return value* a fully functional XMPLink object

**DESCRIPTION**

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

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

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

Before you call this method, you must first ensure that id represents a valid XMPLinkSource object in this draft. After this method executes successfully, the return value is a fully functional XMPLinkSource object with a reference count >= 1.

**EXCEPTIONS**

`kXMPErrCannotGetLink`
Cannot create XMPLink object.

## GetName

Returns the name of this Draft.

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

*return value* Name of this Draft.

**DESCRIPTION**

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

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

If this Draft does not have a name, `kXMPNull` is returned.

## GetPart

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

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `id` | the id of the desired part |
| *return value* | a fully functional XMPPart object |

**DESCRIPTION**

Part. Public.

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

Before you call this method, you must first ensure that id represents a valid Part in this Draft. After this method executes successfully, the return value is a fully functional XMPPart object with a reference count >= 1.

**EXCEPTIONS**

`kXMPErrCannotGetPart`
Cannot create XMPPart object.

## GetPermissions

Returns the Permissions on the Draft.

```
XMPDraftPermissions GetPermissions();
```

| | |
|---|---|
| *return value* | Permissions on the Draft. |

**DESCRIPTION**

Getter. Public.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetStorageUnit

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

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

| | |
|---|---|
| `id` | the id of the storage unit des9ired |
| *return value* | a fully functional XMPStorageUnit object |

**DESCRIPTION**

Getter. Public.

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

Before you call this method, you must first ensure that id represents a valid Storage Unit in this Draft. After this method executes successfully, the return value is a fully functional XMPStorageUnit object with a reference count >= 1.

## IncrementRefCount

(For embedding application or document shell only.)
Increments the reference count of this XMPDraft object.

```
void IncrementRefCount();
```

**DESCRIPTION**

Object Storage. Public.

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

After this method executes successfully, the reference count of this object is incremented by 1.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## InitDraft

(For embedding application or document shell only.)
Initializes this XMPDraft object with the supplied parameters.

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

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

**DESCRIPTION**

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

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

After this method executes successfully, this is an initialized XMPDraft object.

## Purge

(For embedding application or document shell only.)
Purge memory from ephemeral store until size bytes (not necessarily contiguous) have been freed up.

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

| | |
|---|---|
| `size` | Number of bytes to purge |
| *return value* | Number of bytes actually purged |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

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

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

After this method executes successfully, 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

(For embedding application or document shell only.)
Releases this XMPDraft object.

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

*return value* fDocument

**DESCRIPTION**

Reference counting. Public.

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

After this method executes successfully, this XMPDraft object is no longer a valid XMPDraft object.

## ReleaseFrame

(For embedding application or document shell only.)
Releases the XMPFrame object.

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

`frame` XMPFrame object to be released



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

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

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

Before you call this method, you must first ensure that frame is a XMPFrame object with a refCount of 0.

**EXCEPTIONS**

`kXMPErrInvalidRefCount`
RefCount of object is not 0.

## ReleaseLink

(For embedding application or document shell only.)
Releases the XMPLink object.

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

`link` XMPLink object to be released

**DESCRIPTION**

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

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

Before you call this method, you must first ensure that link is a XMPLink object with a refCount of 0.

**EXCEPTIONS**

`kXMPErrInvalidRefCount`
Refcount of link is not 0.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## ReleasePart

Releases the given XMPPart object.

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

`part` XMPPart object to be released

**DESCRIPTION**

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

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

Before you call this method, you must first ensure that part is a XMPPart object with a refCount of 0.

**EXCEPTIONS**

`kXMPErrInvalidRefCount`
RefCount of part is not 0.

## ReleaseStorageUnit

(For embedding application or document shell only.)
Decrement the reference count for the given XMPStorageUnit object.

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

`storageUnit`
XMPStorageUnit object to be released

*return value* this



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Reference counting

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

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

Before you call this method, you must first ensure that storageUnit is a XMPStorageUnit object with a refCount of 0.

**EXCEPTIONS**

`kXMPErrInvalidRefCount`
Refcount of storageUnit is not 0.

## RemoveChanges

(For embedding application or document shell only.)
Removes any changes been made between this Draft and the Draft immediately below it.

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

*return value* this

**DESCRIPTION**

Draft Management. Public.

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

Before you call this method, you must first ensure that this Draft is "gotten" with kDPExclusiveWrite perms. After this method executes successfully, this Draft is 'empty'.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## RemoveFrame

(For embedding application or document shell only.)
Removes the frame persistently from the Draft and destroys the XMPFrame associated with it.

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

`frame` the frame to be removed

### DESCRIPTION

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

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

Before you call this method, you must first ensure that the refcount of the given XMPFrame object is 1.

The permissions of this XMPDraft are kDPExclusiveWrite or kDPSharedWrite. After this method executes successfully, frame is no longer a valid XMPFrame object.

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

### EXCEPTIONS

`kXMPErrInvalidStorageUnit`
Invalid XMPFrame with no XMPStorageUnit associated with it.

`kXMPErrInvalidRefCount`
RefCount of frame is not 1.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## RemoveFromDocument

(For embedding application or document shell only.)
Removes this Draft persistently from its Document and destroys the XMPDraft associated with it.

```
void    RemoveFromDocument();
```

**DESCRIPTION**

Draft Management. Public.

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

Before you call this method, you must first ensure that this XMPDraft is gotten with write permissions.

This Draft is 'empty'.

This Draft is not the Base Draft. After this method executes successfully, 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

(For embedding application or document shell only.)
Removes the Link referred to by link from this Draft.

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

`link` Link to be removed

**DESCRIPTION**

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

Before you call this method, you must first ensure that the refcount of the given XMPLink object is 1.

The permissions of this XMPDraft are kDPExclusiveWrite or kDPSharedWrite. After this method executes successfully, link is no longer a valid XMPLink object

This Draft no longer contains a Link with the corresponding id.

#### EXCEPTIONS

`kXMPErrInvalidRefCount`
Refcount of link is not 1.

`kXMPErrInvalidStorageUnit`
Invalid XMPLink with no XMPStorageUnit associated with it.

## RemoveLinkSource

(For embedding application or document shell only.)
Removes the XMPLinkSource referred to by link from this draft.

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

`link` Link to be removed

#### DESCRIPTION

Destruction. Public. Private for use only by XMPLinkSource.

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

Before you call this method, you must first ensure that the refcount of the link argument is 1.

The permissions of this draft are kDPExclusiveWrite or kDPSharedWrite. After this method executes successfully, link is no longer a valid XMPLinkSource object.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

**EXCEPTIONS**

`kXMPErrInvalidRefCount`
Refcount of link is not 1.

`kXMPErrInvalidStorageUnit`
Invalid XMPLinkSource with no XMPStorageUnit associated with it.

## RemovePart

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

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

`part` Part to be removed

**DESCRIPTION**

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

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

Before you call this method, you must first ensure that the refcount of the given XMPPart object is 1.

The permissions of this XMPDraft object are kDPExclusiveWrite or kDPSharedWrite. After this method executes successfully, part is no longer a valid XMPPart object.

This Draft no longer contains a Part with the corresponding id.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

`kXMPErrInvalidRefCount`
Refcount of part is not 1.

`kXMPErrInvalidStorageUnit`
Invalid XMPPart with no XMPStorageUnit associated with it.

## RemoveStorageUnit

(For embedding application or document shell only.)
Removes from this Draft the Storage Unit referred to by the XMPStorageUnit object.

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

`storageUnit`
Storage unit to be removed

**DESCRIPTION**

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

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

Before you call this method, you must first ensure that the refcount of the given XMPStorageUnit object is 1.

The permissions of this XMPDraft are kDPExclusiveWrite or kDPSharedWrite. After this method executes successfully, storageUnit is no longer a valid XMPStorageUnit object.

This Draft no longer contains a storage unit with the corresponding id.

**EXCEPTIONS**

`kXMPErrInvalidStorageUnit`
Invalid storageUnit.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## SaveToAPrevious

(For embedding application or document shell only.)
Saves the contents of one draft to another.

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

| | |
|---|---|
| `to` | draft to move changes to |
| *return value* | this |

### DESCRIPTION

Object Management. Public.

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

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

## SetChangedFromPrev

Marks this Draft dirty (i.e., ChangedFromPrev).

```
void SetChangedFromPrev();
```

### DESCRIPTION

Externalization. Public.

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

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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## SetName

(For embedding application or document shell only.)
Sets the name of this Draft.

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

`name` New name of this draft

**DESCRIPTION**

Setter. Public.

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

Before you call this method, you must first ensure that name represents a valid XMPDraftName.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPDragAndDrop

*Superclasses* `XMPObject`

*Subclasses* none

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.

## Description

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 transferred 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).

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

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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

Invariants: 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.

## Methods

```
XMPDragAndDrop
~XMPDragAndDrop
Clear
GetContentStorageUnit
InitDragAndDrop
Purge
StartDrag
```

Detailed descriptions of these methods follow, in alphabetical order.

### XMPDragAndDrop

(For embedding application or document shell only.)
This function is the constructor of the XMPDragAndDrop object.

```
XMPDragAndDrop();
```

**DESCRIPTION**

Initialization; Drag-and-drop. Public. This method is called once by XMPSession::InitSession.

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

It is called once at the process' startup time.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## ~XMPDragAndDrop

(For embedding application or document shell only.)
This function is the destructor for the XMPDragAndDrop object.

```
~XMPDragAndDrop();
```

**DESCRIPTION**

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

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

It is called once when OpenDoc quits. It should close all the required system services that it has initialized.

## Clear

This function removes all the user data stored in the XMPDragAndDrop object.

```
void Clear();
```

**DESCRIPTION**

Embedding, Drag-and-Drop. Public. None.

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

This is called by a Part before it is going to put data into this object for transfer.

After this method executes successfully, the XMPStorageUnit should be empty.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetContentStorageUnit

This function returns an XMPStorageUnit object.

```
XMPStorageUnit* GetContentStorageUnit();
```

*return value* An empty XMPStorageUnit object for the caller to copy the data to.

**DESCRIPTION**

Embedding, Drag-and-Drop. Public.

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

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 GetContentStorageUnit() is called.

After this method executes successfully, an empty XMPStorageUnit is returned.

## InitDragAndDrop

(For embedding application or document shell only.)
It initializes all the required system services which include a drag mechanism and a data transfer mechanism.

```
void InitDragAndDrop();
```

**DESCRIPTION**

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

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

After this method executes successfully, this is an initialized XMPDragAndDrop object.

## Purge

(For embedding application or document shell only.)
Purges all the ephemeral structures in this object.

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

| | |
|---|---|
| `size` | Number of bytes to purge. |
| *return value* | Number of bytes actually purged. |

**DESCRIPTION**

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

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

## StartDrag

This function initiates a drag.

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

| | |
|---|---|
| `srcFrame` | Frame in which the drag is initiated. |
| `imageType` | Type of the image being dragged. |
| `image` | Data of the image being dragged. |


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `destPart` | If kDropSucceed is returned as the function result, destPart contains the destination part. Otherwise, the content of this parameter is undefined. |
| `refCon` | Extra information needed for dragging (mainly for platform specific data). |
| *return value* | kDropSucceed means a drop is successful. kDropFail means the drop is unsuccessful. kDropUnfinished means the drop is not completed yet. |

**DESCRIPTION**

Embedding, Drag-and-Drop. Public.

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

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.

Before calling this method, you must ensure that XMPStorageUnit must contain client data from the Part initiating the drag.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPDragItemIterator

*Superclasses* none

*Subclasses* none

XMPDragItemIterator is a companion of XMPDragAndDrop, allowing a Part to iterate over all the XMPStorageUnits in the drag.

## Description

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.

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

## Methods

`First`

`IsNotComplete`

`Next`

Detailed descriptions of these methods follow, in alphabetical order.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## First

Returns the first storage unit in the drag item list.

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

*return value* The first storage unit in the drag item list. `kXMPNull` if none.

**DESCRIPTION**

Embeddeding, Drag and Drop.. Public.

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

Before calling this method, you must ensure that a constructed object of this class exists. After this method executes successfully, the result contains the first storage unit.

## IsNotComplete

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

```
XMPBoolean IsNotComplete();
```

*return value* `kXMPTrue`, if there are more windows

**DESCRIPTION**

Embeddeding, Drag and Drop.. Public.

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

Before calling this method, you must ensure that a constructed object of this class exists. After this method executes successfully, the result contains `kXMPTrue` if there are more storage units, `kXMPFalse` otherwise.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Next

Returns the next storage unit in the drag item list.

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

*return value* The next storage unit in the drag item list. `kXMPNull` if none.

**DESCRIPTION**

Embeddeding, Drag and Drop.. Public.

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

Before calling this method, you must ensure that First() has been called. After this method executes successfully, the result contains the next storage unit.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPEmbeddedFramesIterator

*Superclasses* none

*Subclasses* none

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

## Description

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.

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

Invariants: Varies according to implementation.

## Methods

```
First
IsNotComplete
Next
```

Detailed descriptions of these methods follow, in alphabetical order.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## First

The `First` method initializes the iteration sequence and returns the first embedded frame of the part.

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

*return value* The first embedded frame of the part.

**DESCRIPTION**

Embedding. Public. Derived class **can** override. Derived class **can** call base class behavior, during derived class behavior.

## IsNotComplete

The `IsNotComplete` method answers whether there are more frames left in the iteration sequence.

```
XMPBoolean IsNotComplete();
```

*return value* Returns a Boolean reflecting whether to continue the iteration.

**DESCRIPTION**

Embedding. Public. Derived class **can** override. Derived class **can** call base class behavior, during derived class behavior.

## Next

The `Next` method returns the next embedded frame in the iteration sequence.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

*return value* The next embedded frame of the part.

**DESCRIPTION**

Embedding. Public. Derived class **can** override. Derived class **can** call base class behavior, during derived class behavior.

Before calling this method, you must ensure that `First()` must previously have been called. `IsNotComplete()` must also return`kXMPTrue`. After it executes successfully, the iteration sequence is advanced to the next frame.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPFacet

`XMPFacet -> XMPObject`

The `XMPFFacet` class is used to describe the layout environment of individual frames, and to enable placement of those frames on a canvas for display and for event dispatching.

## Description

**Facet** objects hold non-persistent information regarding the layout of the individual frames that appear in an OpenDoc document. Parts create facets for each location where they intend to display their embedded frames. A part can have multiple frames. Each of these frames can be displayed in multiple facets. Each embedded frame in a document has at least one corresponding facet that is used in the display and manipulation of that frame. One facet exists for each and every location where a frame is visible in a window.

Facets are organized hierarchically, and are all descended from a single **root facet**. There can only be one root facet for each window or printing page. All other facets contained in that window are descended from this root facet. All facets describing one specific frame must be contained within the facet of that frame's containing frame.

In most cases, a containing part will create only one facet for each embedded 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 multiple facets, one for each place where the containing part wants to display the embedded frame.

All parts should support the capability for creating multiple facets that reference an individual frame. In most cases, all facets that display the same frame also display that frame's contents identically. In the exceptional cases where a part has a reason to display a specific facet differently acccording to its location, it may store `partInfo` data in each of the instances of that facet. This partInfo data is used to distinguish between these different instances.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Facets may exist for frames that have been previously visible, and which have been scrolled out of view, or they may eagerly be created by parts for frames that are expected to become visible soon. Facets that are not currently visible in a window are commonly purged from memory under low-memory conditions.

Facets hold additional information regarding the geometry of their corresponding frames. In addition to the frame's own frame shape, each facet also maintains a **clip shape** and an **active shape**. The clip shape describes the area in which the facet's part may display itself. The containing part controls the clip shape. The active shape describes the regions 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**. The external transform describes where the facet is located within its containing part's coordinate space.

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 transform are both distinguished by which environment the part wants to display in, either canvas or window. In most cases, the part will be displayed in the canvas environment. But in cases of real-time interaction, such as when rubber-banding or dragging, the part may need to display directly in the window. In such cases, it should use the aggregates for the window environment.

A facet can possess its own canvas. When this occurs, the facet and all its children 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 this decision must also create that canvas and register itself as the owner of that canvas. The registered part is then responsible for moving the data from the facet's canvas to the parent canvas.

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

`XMPFacet` is instantiated by part objects. Part developers will not subclass `XMPFacet`. Platform implementors can subclass if needed, but that should not normally be necessary.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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 possess a containing facet.

For more information on XMPFacet, see the class descriptions for XMPShape, XMPTransform, XMPCanvas, and XMPFrame.

## Methods

The methods of XMPFacet are primarily used for accessing, imaging, and manipulating facet objects.

### Constructor

| | |
|---|---|
| XMPFacet | Used for contructing the XMPFacet object. |

### Initialization

| | |
|---|---|
| InitFacet | Initializes a newly constructed facet. |

### Access

| | |
|---|---|
| ChangeActiveShape | Sets the facet's active shape, and notifies UI subsystem dependents. |
| ChangeExternalTransform | Sets the external transform of the facet, and notifies the facet's part of the change via Part::ExternalTransformChanged(). |
| ChangeClipShape | Sets the facet's clip shape, and notifies the facet's part of the change via Part::ClipShapeChanged(). |
| ChangeGeometry | Sets the facet's clip shape and/or its externalTransform, and notify the facet's part of the change via Part::GeometryChanged(). |
| ChangeHighlight | Sets the highlight state of this facet, and notifies its part via Part::HighlightChanged(). |
| GetActiveShape | Returns the facet's active shape. |
| GetAggregateClipShape | Returns the shape that is the intersection of this facet's clip shape with all the containing facets' clip shapes displayed on this facet's canvas. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `GetCanvas` | Returns the canvas that this facet's part should image on. |
| `GetClipShape` | Returns the clip shape of the receiver. |
| `GetContainingFacet` | Returns the containing facet of the receiver. |
| `GetContentTransform` | Calculates and returns the facet's content transform. |
| `GetExternalTransform` | Returns the external transform of the facet. |
| `GetFrame` | Returns the frame this facet displays. |
| `GetFrameTransform` | Calculates and returns the facet's frame transform. |
| `GetHighlight` | Returns the highlight state of the facet. |
| `GetPartInfo` | Returns the `partInfo` data that a part has stored in this facet. |
| `GetWindow` | Returns the window that this facet is displayed in. |
| `GetWindowAggregateClipShape` | Returns the shape that is the intersection of this facet's clip shape with all the clip shapes of all its containing facets currently displayed in this facet's window. |
| `GetWindowContentTransform` | Calculates and returns the facet's window content transform. |
| `GetWindowFrameTransform` | Calculates and returns the facet's window frame transform. |
| `SetCanvas` | Sets the canvas of this facet. |
| `SetPartInfo` | Sets the `partInfo` data for this facet. |
| `SetSelected` | Sets the facet's `isSelected` flag. |
| `SetWindow` | Sets the facet's window. |

#### Imaging

| | |
|---|---|
| `Draw` | Tells the facet's part to draw itself within `invalidShape` in this facet. |
| `DrawActiveBorder` | Draws the facet's active border adornment. |
| `DrawChildren` | Draws all this facet's child facets that are in need of updating. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `DrawChildrenAlways` | Draws all this facet's child facets, whether they need updating or not. |
| `DrawnIn` | Notifies the facet it has been drawn in using a different path than the path that is used in response to an update event. |
| `Invalidate` | Causes the underlying graphics system to mark the area in `invalidShape` as needing repainting. |
| `InvalidateActiveBorder` | Marks the area under the facet's active border adornment as in need of redrawing. |
| `Update` | Updates the facet's canvas in `invalidShape` by drawing this facet and any of its children whose `clipShape` intersects `invalidShape`. |
| `Validate` | Causes the underlying graphics system to mark the area in `validShape` as no longer in need of repainting. |

#### Facets

| | |
|---|---|
| `ActiveBorderContainsPoint` | Indicates whether the point is within the active border adornment for this facet. |
| `ContainsPoint` | Indicates whether the point is within the facet. |
| `CreateEmbeddedFacet` | Asks a facet to create an embedded facet. |
| `CreateFacetIterator` | Creates an iterator for the children facet of the receiver. |
| `HasCanvas` | Indicates whether this facet has its own canvas. |
| `IsSelected` | Indicates whether the facet is selected within its container. |
| `MoveBefore` | Moves one of a facet's child facets in front of a sibling facet. |
| `MoveBehind` | Moves one of a facet's child facets behind a sibling facet. |
| `RemoveFacet` | Asks a facet to remove one of its child facets. |

#### Destructor

| | |
|---|---|
| `~XMPFacet` | Used as the `XMPFacet` class destructor. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# Method Descriptions

## XMPFacet

The `XMPFacet` method is used for contructing the`XMPFacet` object.

```
XMPFacet();
```

**DESCRIPTION**

You must override this method.

After executing this method successfully, you must call `InitFacet` before the returned instance of `XMPFacet` is ready to use.

## ~XMPFacet

The `~XMPFacet` method is used as the `XMPFacet` class destructor.

```
~XMPFacet();
```

**DESCRIPTION**

You must override this method.

## ActiveBorderContainsPoint

The `ActiveBorderContainsPoint` method indicates whether the point is within the active border adornment for this facet.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `point` | Indicates the location to test, expressed in window coordinates. |
| *return value* | Returns a Boolean indicating whether the point is within the facet's border shape. |

## ChangeActiveShape

The `ChangeActiveShape` method sets the facet's active shape, and notifies UI subsystem dependents.

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

`activeShape`
A reference to the new active shape for the facet.

**DESCRIPTION**

Only the facet's part should change its active shape.

## ChangeClipShape

The `ChangeClipShape` method sets the facet's clip shape, and notifies the facet's part of the change via `Part::ClipShapeChanged()`.

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

`clipShape` A reference to the new clip shape for the facet.

**DESCRIPTION**

This method invalidates any cached aggregate clip shape. Only the facet's containing part (or window) should change its clip shape.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## ChangeExternalTransform

The `ChangeExternalTransform` method sets the external transform of the facet.

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

`transform` A reference to the new external transform for the facet.

**DESCRIPTION**

After changing the transform, this method notifies the facet's part of the change via `Part::ExternalTransformChanged()`. Only a facet's containing part should change its external transform.

## ChangeGeometry

The `ChangeGeometry` method sets the facet's clip shape and/or its externalTransform, and notify the facet's part of the change via Part::GeometryChanged().

```
void ChangeGeometry  (XMPShape* clipShape,
                      XMPTransform* transform);
```

`clipShape` A reference to the new clip shape for the facet.

`transform` A reference to the new external transform for the facet.

**DESCRIPTION**

Called by the containing part.

This method invalidates any cached aggregate clip shape and transforms. Only the facet's containing part (or window) should change its geometry.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## ChangeHighlight

The `ChangeHighlight` method sets the highlight state of this facet, and notifies its part via `Part::HighlightChanged()`.

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

`highlight` The new highlight state of the facet. Must be one of the following: `kXMPNoHighlight`, `kXMPFullHighlight`, `kXMPDimHighlight`.

## ContainsPoint

The `ContainsPoint` method indicates whether the point is within the facet.

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

`point` Indicates the location to test, expressed in window coordinates.

*return value* A Boolean indicating whether the point is within the facet's shape.

**DESCRIPTION**

This method transforms the point to frame coordinates and tests the result against the intersection of the clip and active shapes.

## CreateEmbeddedFacet

The `CreateEmbeddedFacet` method asks a facet to create an embedded facet.

```
XMPFacet* CreateEmbeddedFacet
                        (XMPFrame* frame,
                         XMPShape* clipShape,
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

```
                                XMPTransform* externalTransform,
                                XMPFacet* siblingFacet,
                                XMPFramePosition position);
```

`frame` A reference to the frame that the new facet will display.

`clipShape` A reference to the initial clip shape for the new facet.

`externalTransform`
A reference to the initial external transform for the new facet.

`siblingFacet`
A reference to 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.

*return value* A reference to the new facet.

#### DESCRIPTION

This method 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)].

Before calling this method, you must ensure that the receiving facet is a facet of the containingFrame of the `frame` parameter. `siblingFacet` is a child facet of the receiving facet. After executing this method successfully, the facet that is returned is a child facet of the receiving facet, with `clipShape` and `externalTransform` set to the indicated values.

#### EXCEPTIONS

`kXMPInvalidFacet`
The designated `siblingFacet` parameter is not a child of the receiver.

`kXMPErrInvalidPositionCode`
The designated `position` parameter is not a valid position code.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## CreateFacetIterator

The `CreateFacetIterator` method creates an iterator for the children facet of the receiver.

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

`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.

*return value* A reference to the new iterator. Caller assumes responsibility for memory.

### DESCRIPTION

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 enumerate the receiver's immediate children.

After calling this method successfully, a new iterator is created and is ready to use.

## Draw

The `Draw` method tells the facet's part to draw itself within `invalidShape` in this facet.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`invalidShape`
A reference to the area in which the part should draw itself, expressed in window coordinates.

**DESCRIPTION**

The facet transforms `invalidShape` into frame coordinates, and passes the request to the part via `Part::Draw()`.

## DrawActiveBorder

The `DrawActiveBorder` method draws the facet's active border adornment.

```
void DrawActiveBorder();
```

## DrawChildren

The `DrawChildren` method draws all this facet's child facets that are in need of updating.

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

`invalidShape`
A reference to the shape in which the facets should draw.

## DrawChildrenAlways

The `DrawChildrenAlways` method draws all this facet's child facets, whether they need updating or not.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`invalidShape`
A reference to the shape in which the facets should draw the children.

## DrawnIn

The `DrawnIn` method notifies the facet it has been drawn in using a different path than the path that is used in response to an update event.

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

`shape` A reference to the shape in which the facet has been drawn, expressed in frame coordinates.

**DESCRIPTION**

This method notifies the owner of the facet's canvas that the parent canvas must be updated with the altered contents.

## GetActiveShape

The `GetActiveShape` method returns the facet's active shape.

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

*return value* A reference to the facet's active shape.

**DESCRIPTION**

If no active shape has been set, this method returns its frame's frame shape.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetAggregateClipShape

The `GetAggregateClipShape` method returns the shape that is the intersection of this facet's clip shape with all the containing facets' clip shapes displayed on this facet's canvas, up to the topmost facet.

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

*return value* A reference to the facet's aggregate clip shape.

## GetCanvas

The `GetCanvas` method returns the canvas that this facet's part should image on.

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

*return value* A reference to the facet's canvas.

### DESCRIPTION

If the facet has no canvas of its own, it will return its containing facet's canvas, recursively.

### EXCEPTIONS

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

## GetClipShape

The `GetClipShape` method returns the clip shape of the receiver.

```
XMPShape* GetClipShape();
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

*return value* A reference to the facet's clip shape.

## GetContainingFacet

The `GetContainingFacet` method returns the containing facet of the receiver.

```
XMPFacet* GetContainingFacet();
```

*return value* A reference to the containing facet of the receiver. May be `kXMPNull`.

## GetContentTransform

The `GetContentTransform` method calculates and returns the facet's content transform.

```
XMPTransform* GetContentTransform();
```

*return value* A reference to the facet's content transform.

**DESCRIPTION**

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetExternalTransform

The `GetExternalTransform` method returns the external transform of the facet.

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

*return value* A reference to the facet's external transform.

## GetFrame

The `GetFrame` method returns the frame this facet displays.

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

*return value* A reference to the facet's frame.

## GetFrameTransform

The `GetFrameTransform` method calculates and returns the facet's frame transform.

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

*return value* A reference to the facet's frame transform.

**DESCRIPTION**

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 that describes the frame coordinate space of this facet on its canvas.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetHighlight

The `GetHighlight` method returns the highlight state of the facet.

```
XMPHighlight GetHighlight();
```

*return value* The highlight state of the facet.

**DESCRIPTION**

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

## GetPartInfo

The `GetPartInfo` method returns the `partInfo` data that a part has stored in this facet.

```
XMPInfoType GetPartInfo();
```

*return value* The facet's partInfo.

## GetWindow

The `GetWindow` method returns the window that this facet is displayed in.

```
XMPWindow* GetWindow();
```

*return value* A reference to the facet's window.

**DESCRIPTION**

This value may be `kXMPNull` if the facet is a printing facet and does not actually appear in any window.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetWindowAggregateClipShape

The `GetWindowAggregateClipShape` method returns the shape that is the intersection of this facet's clip shape with all the clip shapes of all its containing facets currently displayed in this facet's window.

```
XMPShape* GetWindowAggregateClipShape();
```

*return value* A reference to the facet's aggregate clip shape.

## GetWindowContentTransform

The `GetWindowContentTransform` method calculates and returns the facet's window content transform.

```
XMPTransform* GetWindowContentTransform();
```

*return value* A reference to the facet's window content transform.

**DESCRIPTION**

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

## GetWindowFrameTransform

The `GetWindowFrameTransform` method calculates and returns the facet's window frame transform.

```
XMPTransform* GetWindowFrameTransform();
```

*return value* A reference to the facet's window frame transform.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

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

## HasCanvas

The `HasCanvas` method indicates whether this facet has its own canvas.

```
XMPBoolean HasCanvas();
```

*return value* A Boolean indicating whether the facet has its own canvas.

## InitFacet

The `InitFacet` method initializes a newly constructed facet.

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

`frame` A reference to the facet's frame. This value never changes over the life of the facet.

`clipShape` A reference to the initial `clipShape` of the facet.

`externalTransform`
A reference to the initial `externalTransform` of the facet.

**DESCRIPTION**

This method MUST be called before the facet can be used.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Before calling this method, you must ensure that the facet is newly constructed. After calling this method successfully, the facet can be used for display or for event dispatching.

## Invalidate

The `Invalidate` method causes the underlying graphics system to mark the area in `invalidShape` as needing repainting.

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

`invalidShape`
A reference to the area to be marked as invalid, expressed in frame coordinates.

**DESCRIPTION**

This area is marked on this facet's canvas and all parent canvases, so that changes in this facet can be reflected in those canvases.

**EXCEPTIONS**

`kXMPErrInvalidFacet`
Can't find the facet for the parent canvas.

## InvalidateActiveBorder

The `InvalidateActiveBorder` method marks the area under the facet's active border adornment as in need of redrawing.

```
void InvalidateActiveBorder();
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

This method can be used to erase the active border.

## IsSelected

The `IsSelected` method indicates whether the facet is selected within its container.

```
XMPBoolean IsSelected();
```

*return value* Value of the facet's `isSelected` flag.

**DESCRIPTION**

Information garnered by this method is used by the dispatcher to determine where to direct mouse events.

## MoveBefore

The `MoveBefore` method moves one of a facet's child facets in front of a sibling facet.

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

`child` A reference to the child facet that is to be repositioned.

`sibling` A reference to the facet to move `child` in front of. If the value is `kXMPNull`, `child` is moved to the front of all its siblings.

**DESCRIPTION**

Before calling this method, you must ensure that both `child` and `sibling` are children of the receiver. After executing this method successfully, the specified `child` has been repositioned in front of `sibling` or in front of all siblings.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

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

## MoveBehind

The `MoveBehind` method moves one of a facet's child facets behind a sibling facet.

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

`child` A reference to the child facet that is to be repositioned.

`sibling` A reference to the facet behind which to move `child`. If the value is `kXMPNull`, `child` is moved behind all its siblings.

**DESCRIPTION**

Before calling this method, you must ensure that both `child` and `sibling` are children of the receiver. After executing this method successfully, `child` has been repositioned behind `sibling` or behind all siblings.

**EXCEPTIONS**

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

## RemoveFacet

The `RemoveFacet` method asks a facet to remove one of its child facets.

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

`facet` A reference to the child facet that is to be removed.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

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)]. You should then delete the removed facet.

Before calling this method, you must ensure that the `facet` is a child of the receiver. After executing this method successfully, `facet` should not be used again. You can then delete `facet`.

**EXCEPTIONS**

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

## SetCanvas

The `SetCanvas` method sets the canvas of this facet.

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

`canvas` A reference to the new canvas.

**DESCRIPTION**

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

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## SetPartInfo

The `SetPartInfo` method sets the `partInfo` data for this facet.

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

`partInfo` The new partInfo for this facet.

**DESCRIPTION**

`partInfo` should only be set by a facet's part.

## SetSelected

The `SetSelected` method sets the facet's `isSelected` flag.

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

`isSelected` A Boolean specifying the new value of the flag.

## SetWindow

The `SetWindow` method sets the facet's window.

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

`window` A reference to the window for the facet.

**DESCRIPTION**

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Update

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

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

`invalidShape`
A reference to the shape in which the facet's canvas must be repainted, expressed in window coordinates.

## Validate

The `Validate` method causes the underlying graphics system to mark the area in `validShape` as no longer in need of repainting.

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

`validShape` A reference to the shape to be marked as valid, expressed in frame coordinates.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPFacetIterator

*Superclasses* none

*Subclasses* none

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

## Description

Ancestors: none

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.

## Methods

`XMPFacetIterator`

`~XMPFacetIterator`

`First`

`IsNotComplete`

`Next`

`SkipChildren`



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Detailed descriptions of these methods follow, in alphabetical order.

## XMPFacetIterator

Construct an iterator on facet and its child facets.

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

`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`

### DESCRIPTION

None.. Public.

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

## ~XMPFacetIterator

Destructor.

```
~XMPFacetIterator();
```

### DESCRIPTION

None.. Public.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

## First

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

```
XMPFacet* First();
```

*return value* The first facet in the sequence.

**DESCRIPTION**

None.. Public.

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

## IsNotComplete

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

```
XMPBoolean IsNotComplete();
```

*return value* True if iterator is not yet complete, false otherwise.

**DESCRIPTION**

None.. Public.

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

## Next

Answer the next facet in the sequence.

```
XMPFacet* Next();
```

*return value* The next facet.

**DESCRIPTION**

None.. Public.

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

## SkipChildren

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.

```
void SkipChildren();
```

**DESCRIPTION**

None.. Public.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPFocusOwnerIterator

*Superclasses* none

*Subclasses* none

The `XMPFocusOwnerIterator` class 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.

## Description

`XMPFocusOwnerIterator` is an abstract class. This class has no superclass. Focus modules for non-exclusive foci must implement a subclass of this class.

Once a focus is determined to be non-exclusive, the focus module in which it is registered invokes the `CreateOwnerIterator` method so that the owners of the specified focus can be enumerated.

Related classes are `XMPArbitrator` and `XMPFocusModule`

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

## Methods

The methods of `XMPFocusOwnerIterator` are used for iteration.

#### Creation and Deletion

`XMPFocusOwnerIterator`
: Constructor for the `XMPFocusOwnerIterator` object.

`~XMPFocusOwnerIterator`
: Frees the memory that is allocated by this class.

`InitFocusOwnerIterator`
: Initializes the iterator.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**Iteration**

| | |
|---|---|
| `First` | Returns the first frame which is an owner of this non-exclusive focus. |
| `IsNotComplete` | Specifies whether or not the iteration is complete. |
| `Next` | Returns the next owner frame of this non-exclusive focus. |

Detailed descriptions of these methods follow, in alphabetical order.

## XMPFocusOwnerIterator

The `XMPFocusOwnerIterator` method is the constructor for the `XMPFocusOwnerIterator` object.

```
XMPFocusOwnerIterator();
```

**DESCRIPTION**

Part editors should call `XMPArbitrator::CreateOwnerIterator()`. Implementors of focus modules should provide their own constructor.

This constructor is never called directly. It can only be called by invoking a constructor for a derived class.

After this method executes successfully, the iterator object is constructed, but is not usable until `InitFocusOwnerIterator` is called.

## ~XMPFocusOwnerIterator

The `~XMPFocusOwnerIterator` method frees the memory that is allocated by this class.

```
~XMPPart();
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Implementors of focus modules should provide their own constructor.

Before calling this method, you must ensure that the specified focus is a constructed and initialized object. After this method executes successfully, the memory used by this instance has been freed.

## First

The `First` method returns the first frame which is an owner of this non-exclusive focus.

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

**RETURN VALUE** The first owner frame.

**DESCRIPTION**

Before calling this method, you must ensure that the specified focus is a constructed and initialized object of this class. After it executes successfully, the internal position is advanced.

## InitFocusOwnerIterator

The `InitFocusOwnerIterator` method initializes the iterator.

```
void InitFocusOwnerIterator(
XMPTypeToken* focus
XMPFocusModule* module);
```

`focus` A tokenized string of type `XMPTypeToken`, representing a registered focus type.

`module` A reference to the focus module that stores the owners of a non-exclusive focus.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Before calling this method, you must ensure that the specified focus module represents a valid focus module for a non-exclusive focus. After it executes successfully, a valid iterator exists.

**EXCEPTIONS**

`kXMPErrInvalidFocusModule`
The specified focus module is invalid.

## IsNotComplete

The `IsNotComplete` method specifies whether or not the iteration is complete.

```
XMPBoolean IsNotComplete();
```

*return value* Returns a Boolean: `kXMPTrue` if iteration is not complete, `kXMPFalse` otherwise.

**DESCRIPTION**

Before calling this method, you must ensure that the specified focus is a constructed and initialized object of this class. After it executes successfully, this method returns `kXMPTrue` if the iteration is not complete, `kXMPFalse` otherwise.

## Next

The `Next` method returns the next owner frame of this non-exclusive focus.

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

*return value* The next owner frame.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Before calling this method, you must ensure that the specified focus is a constructed and initialized object of this class. After it executes successfully, the internal position is advanced.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPFocusSet

*Superclasses* none
*Subclasses* none

The `XMPFocusSet` class is used to store a set of tokenized focus types.

## Description

`XMPFocusSet` has no superclass.

When a part is activated, it requests 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 of type `XMPTypeToken`.

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

Instances of `XMPFocusSet` can contain an empty set of foci. In a set where there are repeated items, such as in a bag, repeated items are ignored by the `FocusSetIterator`.

Related classes are `XMPFocusSetIterator` and `XMPArbitrator`. The `XMPSession` object contains a `Tokenize` method for converting ISO strings into tokenized strings.

## Methods

`XMPFocusSet` is involved in Part Activation. Its methods are used for adding and removing foci, and for testing membership.

### Creation and Deletion

| | |
|---|---|
| `XMPFocusSet` | Constructs the focus set. |
| `~XMPFocusSet` | Frees the memory that is allocated by this object. |


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `InitFocusSet` | Initializes the set. |

**Iteration**

| | |
|---|---|
| `CreateIterator` | Creates and returns an iterator for this focus set. |

**Elements**

| | |
|---|---|
| `Add` | Adds the specified focus to the set if it is not already present. |
| `Contains` | Tests the specified focus for membership in the set. |
| `Remove` | Removes the specified focus from the set, if it is present. |

Detailed descriptions of these methods follow, in alphabetical order.

## XMPFocusSet

The `XMPFocusSet` method constructs the focus set.

```
XMPFocusSet();
```

**DESCRIPTION**

Before calling this method, you must ensure that the `InitFocusSet` method is also called. After it executes successfully, the set is constructed, but is not ready for use until `InitFocusSet` is called.

## ~XMPFocusSet

The `~XMPFocusSet` method frees the memory that is allocated by this object.

```
~XMPFocusSet();
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Before calling this method, you must ensure that the object is a constructed and initialized object of this class. After it executes successfully, the memory used by this object is freed.

## Add

The `Add` method adds the specified focus to the set if it is not already present.

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

`focus` A tokenized string of type `XMPTypeToken` representing a focus to add to the set.

**DESCRIPTION**

Can be called by part editors. Before calling this method, you must ensure that the object is a constructed and initialized object of this class. After it executes successfully, the set contains the new focus.

**EXCEPTIONS**

`kXMPErrOutOfMemory` Insufficient memory to call method.

## Contains

The `Contains` method tests the specified focus for membership in the set.

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

`focus` A tokenized string of type `XMPTypeToken` representing a focus to test for membership.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

*return value* `kXMPTrue`, if the set contains the focus, `kXMPFalse` otherwise.

**DESCRIPTION**

Before calling this method, you must ensure that the object is a constructed and initialized object of this class.

## CreateIterator

The `CreateIterator` method creates and returns an iterator for this focus set.

```
XMPFocusSetIterator* CreateIterator();
```

*return value* The iterator

**DESCRIPTION**

The caller is responsible for deleting the iterator.

Before calling this method, you must ensure that the object is a constructed and initialized object of this class. After it executes successfully, the result contains the new iterator.

## InitFocusSet

The `InitFocusSet` method initializes the set.

```
void InitFocusSet();
```

**DESCRIPTION**

Before calling this method, you must ensure that the object is a constructed and initialized object of this class. After it executes successfully, the instance is usable.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Remove

The `Remove` method removes the specified focus from the set, if it is present.

```
void Remove(
XMPTypeToken focus);
```

focus A token for a focus to remove from the set

**DESCRIPTION**

Before calling this method, you must ensure that the object is a constructed and initialized object of this class. After it executes successfully, the specified focus is no longer in the set.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPFocusSetIterator

*Superclasses* none

*Subclasses* none

The `XMPFocusSetIterator` class is a companion of the `XMPFocusSet` class, and is used to iterate over the elements of a focus set.

## Description

`XMPFocusSetIterator` is an abstract class. This class has no superclass.

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.

This class must always have a pointer to the focus set that it is iterating over.

Related classes are `XMPArbitrator`, `XMPFocusModule` and `XMPFocusSet`.

## Methods

The methods of `XMPFocusOwnerIterator` are used primarily for iteration.

#### Creation and Deletion

`~XMPFocusSetIterator`
: Frees memory that is allocated by this class.

#### Iteration

`First`
: Resets the iterator to the beginning, returns the first element, and sets it as the current element.

`IsNotComplete`
: Declares whether the iteration is complete.

`Next`
: Returns the next element of the set, and advances the iterator state to the next element.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Detailed descriptions of these methods, follow, in alphabetical order.

## ~XMPFocusSetIterator

The `~XMPFocusSetIterator` method frees memory that is allocated by this class.

```
~XMPFocusSetIterator();
```

**DESCRIPTION**

Before calling this method, you must ensure that the specified focus is a constructed and initialized object of this class. After it executes successfully, the memory used by this object is freed.

## First

The `First` method resets the iterator to the beginning, returns the first element, and sets it as the current element.

```
XMPTypeToken First();
```

*return value* The first focus in the set

**DESCRIPTION**

This method allocates an iterator supplied by the implementation collection class.

Before calling this method, you must ensure that the specified focus is a constructed and initialized object of this class. After it executes successfully, the first element is returned, and the iterator is advanced to be ready to return the next element.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

`kXMPErrOutOfMemory`
Not enough memory to allocate the LinkedListIterator

## IsNotComplete

The `IsNotComplete` method declares whether the iteration is complete.

```
XMPBoolean IsNotComplete();
```

*return value* Returns a Boolean: `kXMPTrue`, if the iteration is not complete, `kXMPTrue` otherwise.

**DESCRIPTION**

Before calling this method, you must ensure that the specified focus is a constructed and initialized object of this class.

## Next

The `Next` method returns the next element of the set, and advances the iterator state to the next element.

```
XMPTypeToken Next();
```

*return value* The next focus in the set

**DESCRIPTION**

Before calling this method, you must ensure that `First` has been called. After it executes successfully, it returns the next element of the set, and advances the iterator state to the next element.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPFrame

*Superclasses* `XMPPersistentObject` → `XMPRefCntObject` → `XMPObject`

*Subclasses* none

An `XMPFrame` object, in conjunction with its facet, describes the geometric boundary between an embedded part and its containing part.

## Description

A **Frame** is created by a containing part as a place in which to embed another part. Once a part has been embedded, its frame mediates negotiations between the embedded part and its containing part.

A part may be displayed in multiple frames. Each frame represents a particular view of that part's content. A frame may be displayed in multiple locations within a window. Each of these locations corresponds to one facet of that frame.

Each frame maintains a list of facets. These facets describe where the frame is displayed in a window. This list may be empty if the specified frame is currently scrolled out of view. Since the facet hierarchy is navigable, it is possible to find some of a frame's embedded frames by requesting that a frame's facet iterate over its child facets. In this way, it is possible to locate an embedded frame through the facet that references that frame. This technique works in the location of frames that are currently displayed in a window, but is unreliable in the location of all the embedded frames. This is because some of the embedded frames may not currently be displayed and will therefore not have existing facets that reference them.

A frame always maintains a reference to its containing frame. The value of this reference will be returned as `kXMPNull` if the specified frame is the root frame of a window. This reference value doesn't usually change, unless the frame was created before an embedding location was selected. A frame does not hold direct references to its embedded frames. Only the frame's part knows which frames are embedded in it.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

A frame of course maintains a reference to its part. The frame ensures that it is registered as one of that part's display frames when it is created, and ensures that it is removed from that set of display frames when it is deleted. A frame's part may store arbitrary, uninterpreted data in the frame's **partInfo** field. The part alone interprets or manipulates this data, but the partInfo is stored with the frame, and only internalized when it is necessary for the manipulation of that frame.

A frame has two basic shapes that define its geometry. The **frameShape** defines the area that the containing part has delegated to the embedded part. The **usedShape** is the area the embedded part actually requires. Where appropriate, the containing part will wrap content to the contour of the usedShape. A frame maintains a large amount of state information. Some of this data references the embedded part, some the containing part, and some the UI and Imaging subsystems.

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: the coordinate space of the frame itself, and the coordinate space of the frame's content. The internalTransform represents the mapping from the content coordinates to the frame coordinates.

There are two fields that 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 view of a part is shown in the frame. Examples of view types are: table, bar chart or pie chart; drawing or palette; text or outline. Parts MUST support all standard viewTypes, but the set of presentations may vary 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 that the containing part may use in grouping the frames. In cases where a containing part embeds multiple frames that display the same embedded part, the containing part 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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`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 because there is no part editor to display its part

`doesPropagateEvents` - the frame's part delegates unhandled events to its containing part

Frames must always be related to a part, existing as a display frame of that part. Frames must always possess a valid `frameShape`. The `usedShape` need not be set, in which case it will default to be the same as the `frameShape`. A frame must always possess a valid `internalTransform`, that defaults to the identity transformation. `isRoot`, `isSubframe` and `isOverlaid` flags are immutable for the lifetime of a frame.

For more information on`XMPFrame`, see the class descriptions for `XMPPart`, `XMPFacet`, `XMPShape`, `XMPDraft`, `XMPStorageUnit`, and `XMPTransform`.

## Methods

The methods of `XMPFrame` are primarily used for accessing, displaying, and manipulating frame objects.

#### UI Events

| | |
|---|---|
| `Close` | Prepares the frame for being removed from its draft. |
| `DoesPropagateEvents` | Returns a Boolean specifying whether this frame delegates unhandled events to its containing frame. |
| `GetContainingFrame` | Returns the containing frame of this frame. |
| `GetFrameGroup` | Returns the `groupID` of this frame. |
| `GetFrameShape` | Returns this frame's `frameShape`. |
| `GetPart` | Returns this frame's part. |
| `GetPartInfo` | Returns the `partInfo` data the part has stored in this frame. |
| `GetPresentation` | Returns the presentation kind of this frame. |
| `GetSequenceNumber` | Returns the sequence number that this frame's containing part has assigned it. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `GetUsedShape` | Returns the `usedShape` of this frame. |
| `IsFrozen` | Returns a Boolean specifying whether this frame is frozen. |
| `IsOverlaid` | Returns a Boolean specifying whether this frame is overlaid. |
| `IsRoot` | Returns a Boolean specifying whether this is the root frame. |
| `IsSubframe` | Returns a Boolean specifying whether this is a sub-frame. |
| `Release` | Called by another object that had a reference to this frame upon releasing that reference. |
| `Remove` | Prepares the frame for being removed from its draft. |
| `RequestFrameShape` | Called by the frame's part to request a new `frameShape`. |
| `SetContainingFrame` | Sets the containing frame of this frame. |
| `SetDragging` | Sets whether this frame is in process of being dragged. |
| `SetDroppable` | Called by the frame's part to indicate whether it supports drag-and-drop in this frame. |
| `SetFrameGroup` | Sets the `groupID` of this frame. |
| `SetFrozen` | Sets whether this frame is frozen. |
| `SetPartInfo` | Sets the frame's `partInfo` data. |
| `SetPresentation` | Sets the presentation kind of this frame to `presentation`. |
| `SetPropagateEvents` | Sets whether this frame should delegate unhandled events to its containing frame. |
| `SetSequenceNumber` | Sets the sequence number of this frame. |
| `SetSubframe` | Sets whether this frame is a subframe of a containing frame. |
| `SetViewType` | Sets the view type of this frame to `viewType`. |
| `Validate` | Asks the frame to mark `validShape` as no longer needing to be redrawn. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

### Display

| | |
|---|---|
| `ChangeFrameShape` | Called by containing part to change the frame's `frameShape`. |
| `ChangePresentation` | Called by the containing part to request that the frame change its presentation. |
| `ChangeUsedShape` | Called by the frame's part to change its `usedShape`. |
| `ChangeViewType` | Called by the containing part to request that the frame change its `viewType`. |
| `DrawActiveBorder` | Asks the frame to redraw its active border adornment. |
| `GetViewType` | Asks the frame to mark `invalidShape` as in need of redrawing. |
| `Invalidate` | Asks the frame to mark `invalidShape` as in need of redrawing. |
| `InvalidateActiveBorder` | Asks the frame to mark the shape of its active border adornment as needing redrawing. |

### Transform

| | |
|---|---|
| `ChangeInternalTransform` | Called by the frame's part to change its `internalTransform`. |
| `GetInternalTransform` | Returns the `internalTransform` of the frame. |

### Linking

| | |
|---|---|
| `ChangeLinkStatus` | Sets the link status for the frame to the given value. |
| `ContentChanged` | Called by the part displayed by the frame object if the frame is affected by a content change in the part. |
| `GetLinkStatus` | Returns the `XMPLink` status of the frame. |

### Drag and Drop

| | |
|---|---|
| `IsDragging` | Returns a Boolean specifying whether this frame is being dragged, so that the Dispatcher knows not to drag it into itself or into its contents. |
| `IsDroppable` | Returns a Boolean specifying whether this frame's part supports drag-and-drop. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**Facets**

| | |
|---|---|
| `ChangePart` | Removes all of this frame's facets from its old part via `Part::FacetRemoved()`. |
| `CreateFacetIterator` | Creates an iterator for this frame's facets. |
| `FacetAdded` | Notifies the frame that it has a new facet. |
| `FacetRemoved` | Notifies the frame that one of its facets has been removed. |

**Reading and Writing**

| | |
|---|---|
| `Externalize` | Writes out the state of this frame onto its storage unit, suitable for subsequent reconstruction of the frame. |

**Storage**

| | |
|---|---|
| `CloneTo` | Externalizes this `XMPFrame` object and recursively clones any of its references to other Storage Units or Parts. |
| `Purge` | Asks the frame to free up to `size` the specified number of bytes of memory, by releasing non-essential internal data structures. |

Detailed descriptions of these methods follow, in alphabetical order.

## ChangeFrameShape

The `ChangeFrameShape` method is called by containing part to change the frame's `frameShape`.

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

`shape` A reference to the new `frameShape`, expressed in frame coordinates.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

The frame sets its `frameShape` field to `shape` and notifies its part of the change via `Part::FrameShapeChanged()`.

## ChangeInternalTransform

The `ChangeInternalTransform` method is called by the frame's part to change its `internalTransform`.

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

`transform` A reference to the new `internalTransform`.

**DESCRIPTION**

The frame sets its `internalTransform` to be `transform`, and notifies its facets of the change via `Facet::InternalTransformChanged()`.

## ChangeLinkStatus

The `ChangeLinkStatus` method sets the link status for the frame to the given value.

```
void  ChangeLinkStatus (XMPLinkStatus status);
```

**DESCRIPTION**

Parts call this for any `XMPFrame`s that are involved in a link when a link is created, broken, moved, etc.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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. After executing this method successfully, the frame's link status may or may not have changed and frame will call the LinkStatusChanged method of its part.

## ChangePart

The FacetRemoved method removes all of this frame's facets from its old part via Part::FacetRemoved().

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

part A reference to the frame's new part.

**DESCRIPTION**

Set part to be part. Add all its facets to the new part via Part::FacetAdded().

## ChangePresentation

The ChangePresentation method is called by the containing part to request that the frame change its presentation.

```
void  ChangePresentation (XMPTypeToken presentation);
```

presentation
The frame's presentation.

**DESCRIPTION**

This method sets the presentation to presentation, and notifies the frame's part of the change via Part::PresentationChanged(). If the part does not



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

support the requested presentation, it may then change the presentation again using the `Frame::SetPresentation()` method.

## ChangeUsedShape

The `ChangeUsedShape` method is called by the frame's part to change its `usedShape`.

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

`shape` A reference to the new `usedShape`, expressed in frame coordinates.

**DESCRIPTION**

The frame sets its `usedShape` to `shape`, and notifies its containing part of the change via `Part::UsedShapeChanged()`.

## ChangeViewType

The `ChangeViewType` method is called by the containing part to request that the frame change its `viewType`.

```
void  ChangeViewType (XMPTypeToken viewType);
```

`viewType` The frame's new viewType.

**DESCRIPTION**

Sets the view type 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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## CloneTo

The `CloneTo` method externalizes this `XMPFrame` object and recursively clones any of its references to other Storage Units or Parts.

```
XMPStorageUnit* CloneTo (XMPDraftKey key,
                         XMPDraft* destDraft);
```

| | |
|---|---|
| `key` | `XMPDraftKey` identifying the Clone transaction. |
| `destDraft` | A reference to the `destDraft` |
| *return value* | A reference to the Storage Unit to where this frame is cloned. |

**DESCRIPTION**

Note that the actual copying may not be completed until after `EndClone` is finished.

## Close

The `Close` method prepares the frame for being removed from its draft.

```
void Close();
```

**DESCRIPTION**

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.Before calling this method, you must ensure that the frame has a valid part, and is a display frame of that part.After executing this method successfully, the frame makes no references to other frames or parts.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## ContentChanged

The `ContentChanged` method is called by the part displayed by the frame object if the frame is affected by a content change in the part.

```
void ContentChanged (XMPChangeID change);
```

`change` A unique change ID.

**DESCRIPTION**

Public. Called by the part displayed by this frame object.

Causes the `EmbeddedFrameChanged()` method of all containing parts in the frame hierarchy to be called, through 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. After calling this method successfully, the `EmbeddedFrameChange` method of its part will be called subsequently.

## CreateFacetIterator

The `CreateFacetIterator` method creates an iterator for this frame's facets.

```
XMPFrameFacetIterator* CreateFacetIterator();
```

*return value* The iterator. Caller assumes storage responsibility.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## DoesPropagateEvents

The `DoesPropagateEvents` method returns a Boolean specifying whether this frame delegates unhandled events to its containing frame.

```
XMPBoolean DoesPropagateEvents();
```

*return value* The `doesPropagateEvents` flag.

## DrawActiveBorder

The `DrawActiveBorder` method asks the frame to redraw its active border adornment.

```
void DrawActiveBorder();
```

**DESCRIPTION**

Called by the arbitrator object.

The frame forwards the request to all its facets via `Facet::DrawActiveBorder()`.

## Externalize

The `Externalize` method writes out the state of this frame onto its storage unit, suitable for subsequent reconstruction of the frame.

```
void Externalize();
```

**DESCRIPTION**

Before calling this method, you must ensure that the frame has a valid storage unit.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## FacetAdded

The `FacetAdded` method notifies the frame that it has a new facet.

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

`facet` The new facet.

**DESCRIPTION**

The frame adds the facet to its list of facets, and notifies its part of the new facet via `Part::FacetAdded()`.

## FacetRemoved

The `FacetRemoved` method notifies the frame that one of its facets has been removed.

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

`facet` A reference to the facet that is being removed.

**DESCRIPTION**

Only called by the facet object.

The frame removes the facet from its list of facets, and notifies its part via `Part::FacetRemoved()`.

## GetContainingFrame

The `GetContainingFrame` method returns the containing frame of this frame.

```
XMPFrame* GetContainingFrame();
```


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

*return value* This frame's containing frame.

**DESCRIPTION**

This method will return `kXMPNull` if this is a root frame.

## GetFrameGroup

The `GetFrameGroup` method returns the `groupID` of this frame.

```
XMPULong GetFrameGroup();
```

*return value* The `groupID` of this frame.

## GetFrameShape

The `GetFrameShape` method returns this frame's `frameShape`.

```
XMPShape* GetFrameShape();
```

*return value* The frame's `frameShape`, expressed in frame coordinates. Caller may not alter or delete.

## GetInternalTransform

The `GetInternalTransform` method returns the `internalTransform` of the frame.

```
XMPTransform* GetInternalTransform();
```

*return value* The frame's `internalTransform`. Caller may not alter or deallocate.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

This transform specifies how the content displayed in the frame is transformed relative to the frame's coordinate space.

## GetLinkStatus

The `GetLinkStatus` method returns the `XMPLink` status of the frame.

```
XMPLinkStatus GetLinkStatus();
```

*return value* The link status of this frame

**DESCRIPTION**

Frame objects will want to call this for their enclosing frames when `ChangeLinkStatus` is invoked. This will determine how they handle `ChangeLinkStatus`.

## GetPart

The `GetPart` method returns this frame's part.

```
XMPPart* GetPart();
```

*return value* This frame's part.

## GetPartInfo

The `GetPartInfo` method returns the `partInfo` data the part has stored in this frame.

```
XMPInfoType GetPartInfo();
```


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

*return value* The frame's `partInfo` data.

## GetPresentation

The `GetPresentation` method returns the presentation kind of this frame.

```
XMPTypeToken GetPresentation();
```

*return value* This frame's presentation kind.

## GetSequenceNumber

The `GetSequenceNumber` method returns the sequence number that this frame's containing part has assigned it.

```
XMPULong GetSequenceNumber();
```

*return value* The sequence number of this frame in its frame group.

## GetUsedShape

The `GetUsedShape` method returns the `usedShape` of this frame.

```
XMPShape* GetUsedShape();
```

*return value* The `usedShape`, expressed in frame coordinates. Caller may not alter or deallocate.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetViewType

The `GetViewType` method returns the `viewType` of this frame.

```
XMPTypeToken GetViewType();
```

*return value* The `viewType` of this frame.

## Invalidate

The `Invalidate` method asks the frame to mark `invalidShape` as in need of redrawing.

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

`invalidShape`
A reference to the shape that is to be redrawn, expressed in frame coordinates.

**DESCRIPTION**

The frame forwards the request to all its facets via `Facet::Invalidate()`.

## InvalidateActiveBorder

The `InvalidateActiveBorder` method asks the frame to mark the shape of its active border adornment as needing redrawing.

```
void InvalidateActiveBorder();
```

**DESCRIPTION**

This call is used to erase the active border. The frame forwards the request to all its facets via `Facet::InvalidateActiveBorder()`.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## IsDragging

The `IsDragging` method returns a Boolean specifying whether this frame is being dragged, so that the Dispatcher knows not to drag it into itself or into its contents.

```
XMPBoolean IsDragging();
```

*return value* The frame's `isDragging` flag.

## IsDroppable

The `IsDroppable` method returns a Boolean specifying whether this frame's part supports drag-and-drop.

```
XMPBoolean IsDroppable();
```

*return value* The frame's `isDroppable` flag.

## IsFrozen

The `IsFrozen` method returns a Boolean specifying whether this frame is frozen.

```
XMPBoolean IsFrozen();
```

*return value* The `isFrozen` flag of this frame.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## IsOverlaid

The `IsOverlaid` method returns a Boolean specifying whether this frame is overlaid.

```
XMPBoolean IsOverlaid();
```

*return value* The frame's `IsOverlaid` flag.

## IsRoot

The `IsRoot` method returns a Boolean specifying whether this is the root frame.

```
XMPBoolean IsRoot();
```

*return value* The frame's `IsRoot` flag.

## IsSubframe

The `IsSubframe` method returns a Boolean specifying whether this is a sub-frame.

```
XMPBoolean IsSubframe();
```

*return value* The frame's `isSubframe` flag.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Purge

The `Purge` method asks the frame to free up to `size` the specified number of bytes of memory, by releasing non-essential internal data structures.

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

| | |
|---|---|
| `size` | The number of bytes to attempt to free. |
| *return value* | The number of bytes actually freed. |

## Release

The `Release` method is called by another object that had a reference to this frame upon releasing that reference.

```
void Release();
```

**DESCRIPTION**

Decrements the frame's reference count. If there are no more references to the frame, ask the draft to release its storage. Before calling this method, you must ensure that the specified frame has a positive reference count.

## Remove

The `Remove` method prepares the frame for being removed from its draft.

```
void Remove();
```

**DESCRIPTION**

The frame notifies its part it is being removed via `Part::RemoveDisplayFrame()`, then releases its references to its part and



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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, the part asks the draft to remove the frame. Before calling this method, you must ensure that the specified frame has a valid part, and that it is a display frame of that part. After executing this method successfully, the frame makes no references to other frames or parts. Draft may have removed frame.

## RequestFrameShape

The RequestFrameShape method is called by the frame's part to request a new frameShape.

```
XMPShape* RequestFrameShape (XMPShape* shape);
```

| | |
|---|---|
| shape | A reference to the requested shape, expressed in frame coordinates. |
| *return value* | The new frameShape, expressed in frame coordinates. Caller may not alter or deallocate. |

**DESCRIPTION**

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.

## SetContainingFrame

The SetContainingFrame method sets the containing frame of this frame.

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

| | |
|---|---|
| frame | A reference to the new containing frame. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Use of this method is only necessary if the frame is being moved, as will usually be created with correct value.

## SetDragging

The `SetDragging` method sets whether this frame is in process of being dragged.

```
void  SetDragging(
XMPBoolean isDragging);
```

`isDragging` The new Boolean value for the frame's `isDragging` flag.

## SetDroppable

The `SetDroppable` method is called by the frame's part to indicate whether it supports drag-and-drop in this frame.

```
void  SetDroppable (XMPBoolean isDroppable);
```

`isDroppable`
The new value for this frame's `isDroppable` flag.

## SetFrameGroup

The `SetFrameGroup` method sets the `groupID` of this frame.

```
void  SetFrameGroup (XMPULong groupID);
```

`groupID` The new `groupID`.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

The `groupID` should only be set by the containing part.

## SetFrozen

The `SetFrozen` method sets whether this frame is frozen.

```
void  SetFrozen (XMPBoolean isFrozen);
```

`isFrozen` The new Boolean value for the `isFrozen` flag.

## SetPartInfo

The `SetPartInfo` method sets the frame's `partInfo` data.

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

`partInfo` The frame's new `partInfo`.

**DESCRIPTION**

This method should only be called by this frame's part.

## SetPresentation

The `SetPresentation` method sets the presentation kind of this frame to `presentation`.

```
void  SetPresentation (XMPTypeToken presentation);
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`presentation`
The frame's presentation kind.

**DESCRIPTION**

This method should only be called by this frame's part.

## SetPropagateEvents

The `SetPropagateEvents` method sets whether this frame should delegate unhandled events to its containing frame.

```
void  SetPropagateEvents (XMPBoolean doesPropagateEvents);
```

`doesPropagateEvents`
New Boolean value for `doesPropagateEvents` flag.

## SetSequenceNumber

The `SetSequenceNumber` method sets the sequence number of this frame.

```
void SetSequenceNumber (XMPULong sequenceNumber);
```

`sequenceNumber`
The new sequence number.

**DESCRIPTION**

The number should be unique within the frame's group in its containing part. Should only be called by the containing part.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## SetSubframe

The `SetSubframe` method sets whether this frame is a subframe of a containing frame.

```
void SetSubframe (XMPBoolean isSubframe);
```

`isSubframe` The value for the `isSubframe` flag.

## SetViewType

The `SetViewType` method sets the view type of this frame to `viewType`.

```
void SetViewType (XMPTypeToken viewType);
```

`viewType` The frame's new viewType.

**DESCRIPTION**

This method should only be called by frame's part.

## Validate

The `Validate` method asks the frame to mark `validShape` as no longer needing to be redrawn.

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

`validShape` A reference to the shape that no longer needs to be redrawn, expressed in frame coordinates.

**DESCRIPTION**

The frame forwards the request to all its facets via `Facet::Validate()`.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPFrameFacetIterator

*Superclasses* none
*Subclasses* none

XMPFrameFacetIterator lists the facets of a frame.

## Description

Ancestors: none

## Methods

```
XMPFrameFacetIterator
~XMPFrameFacetIterator
First
IsNotComplete
Next
```

Detailed descriptions of these methods follow, in alphabetical order.

### XMPFrameFacetIterator

Construct an iterator on the facets of a frame.

```
XMPFrameFacetIterator    (XMPFrame* frame);
```

`frame` The frame who's facets to iterate.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

None. Public.

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

## ~XMPFrameFacetIterator

Destructor.

```
~XMPFrameFacetIterator();
```

**DESCRIPTION**

None. Public.

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

## First

Return the first facet of the frame.

```
XMPFacet* First();
```

*return value* The first facet in the sequence.

**DESCRIPTION**

None. Public.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## IsNotComplete

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

```
XMPBoolean IsNotComplete();
```

*return value* True if iterator is not yet complete, false otherwise.

**DESCRIPTION**

None. Public.

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

## Next

Answer the next facet in the sequence.

```
XMPFacet* Next();
```

*return value* The next facet.

**DESCRIPTION**

None. Public.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPLink

*Superclasses* `XMPPersistentObject` → `XMPRefCntObject` → `XMPObject`

*Subclasses* none

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.

## Description

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.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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: Dependent parts must be maintained in a list for there may be more than one.

## Methods

`CloneTo`
`Externalize`
`GetChangeID`
`GetChangeTime`
`GetContentStorageUnit`
`GetStatus`
`Lock`
`RegisterDependent`
`ShowSourceContent`
`Unlock`
`UnregisterDependent`

Detailed descriptions of these methods follow, in alphabetical order.

### CloneTo

The `CloneTo` method clones this object into the destination draft.

```
XMPStorageUnit* CloneTo(
XMPDraftKey draftKey,
XMPDraft* destDraft);
```

`draftKey` `XMPDraftKey` identifying the Clone transaction.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`destDraft` A reference to the draft that the specified object is to be cloned into.

*return value* The storage unit containing the persistent representation of this object in the destination draft.

**DESCRIPTION**

Object Storage

Public.

**EXCEPTIONS**

`kXMPErrCloneNotStarted`
A Clone was started unsuccessfully.

`kXMPErrInvalidDraftKey`
Invalid Clone.

## Externalize

Externalizes persistent data

```
void Externalize();
```

**DESCRIPTION**

Storage. Public.

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

After this method executes successfully, the link's persistent storage unit is consistent with the link.

**EXCEPTIONS**

<Storage exceptions>
Storage exceptions may be signalled.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetChangeID

Returns the current change identification of the link content.

```
XMPChangeID GetChangeID();
```

**DESCRIPTION**

Linking. Public. Can be used by parts containing a destination of this link to determine if new content is available.

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

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.

## GetChangeTime

The `GetChangeTime` method returns the time of the last update to the source of the link.

```
time_t GetChangeTime();
```

*return value* The time the source of the link was updated.

**DESCRIPTION**

Linking. Public.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetContentStorageUnit

Returns a reference to the XMPStorageUnit containing the content of the link.

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

`key` A valid key returned by Lock().

*return value* A reference to the content storage unit.

### DESCRIPTION

Linking, Storage. Public. No restrictions.

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

Before calling this method, you must ensure that the key argument is a valid key returned by Lock(). After this method executes successfully, the returned storage unit is guaranteed to be valid until the key is relinquished by Unlock().

### EXCEPTIONS

`kXMPErrInvalidLinkKey`

## GetStatus

Returns the status of the link.

```
XMPError GetStatus();
```

*return value* Zero if the link is functioning normally. Non-zero if some error occurred.

### DESCRIPTION

Linking. Public. Should be called by parts when the user explicitly attempts to get updates from the link.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

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.

## Lock

Acquire exclusive access to the content storage unit of the link for the current thread.

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

| | |
|---|---|
| `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. |
| `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. |
| *return value* | `kXMPTrue` if access is granted and `kXMPFalse` if denied. |

**DESCRIPTION**

Thread Safety. Public. Parts must call this method to acquire a key required by other methods.

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

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.

After this method executes successfully, If kXMPTrue is returned, the current thread has exclusive access to the link's content, and the key parameter is valid.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## RegisterDependent

Puts the clientPart reference on a list for notification of changes to the link's content.

```
void RegisterDependent  (XMPPart* clientPart,
                         XMPChangeID id);
```

`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.

### DESCRIPTION

Linking. Public. Parts containing the destination of this link call this method to receive notification of changes to the link's content.

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

### EXCEPTIONS

`kXMPErrOutOfMemory`
Not enough memory

## ShowSourceContent

Causes the source part of the link to become active, makes the source content visible, and displays the link border.

```
void ShowSourceContent();
```


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Linking. Public.

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

Throws an exception if the source part of the link cannot be located, or if the link has been broken at the source.

**EXCEPTIONS**

`kXMPErrLinkBroken`
Cannot locate the source of the link.

## Unlock

Relinquish a key returned by Lock().

```
void Unlock (XMPLinkKey key);
```

`key` A valid key returned by Lock().

**DESCRIPTION**

Thread safety. Public. Parts must call this method when done accessing or modifying the content of a link.

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

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.

Before calling this method, you must ensure that the key argument is a valid key returned by Lock().

After this method executes successfully, 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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

`kXMPErrInvalidLinkKey`

## UnregisterDependent

Removes the previously registered XMPPart reference from our internal list.

```
void UnregisterDependent   (XMPPart* clientPart);
```

`clientPart` A destination part that had previously registered itself as a dependent of the link.

**DESCRIPTION**

Linking. Public. The destination XMPPart will call this.

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

Return without error if the part was not previously registered.

Before you call this method, clientPart should have previously registered with this link.

After this method executes successfully, clientPart is not registered with this link.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPLinkManager

*Superclasses* none

*Subclasses* none

The Link Manager coordinates the creation and maintenance of cross-document links.

## Description

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.

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 Link Manager and the Link and Link Source 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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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.

#### 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.

## Methods

`AnyLinkImported`

`DraftClosing`

`DraftOpened`

`DraftSaved`

`UnsavedExportedLinks`

Detailed descriptions of these methods follow, in alphabetical order.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## AnyLinkImported

(For embedding application or document shell only.)
Returns `kXMPTrue` if one or more links to source in other documents have automatically updated since the draft was last saved.

```
XMPBoolean AnyLinkImported (XMPDraft* draft);
```

| | |
|---|---|
| `draft` | The draft of interest. |
| *return value* | `kXMPTrue` if one or more links have automatically updated, and `kXMPFalse` otherwise. |

**DESCRIPTION**

Linking. Public. For use by container apps only. Parts do not call.

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

The shell should silently save the imported content if no other changes to the draft were made by the user.

## DraftClosing

(For embedding application or document shell only.)
Notifies the Link Manager that you are about to close a draft.

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

| | |
|---|---|
| `draft` | The draft about to be closed. |

**DESCRIPTION**

Linking. Public. For use by container apps only. Parts do not call.

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

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

links at the source. The shell can call UnsavedExportedLinks() to determine if closing the document would delete edition files.

## DraftOpened

(For embedding application or document shell only.)
Notifies the Link Manager that you have opened a draft.

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

`draft` The draft opened.

**DESCRIPTION**

Linking. Public. For use by container apps only. Parts do not call.

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

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.

## DraftSaved

(For embedding application or document shell only.)
Notifies the Link Manager that you have saved a draft.

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

`draft` The draft saved.

**DESCRIPTION**

Linking. Public. For use by container apps only. Parts do not call.

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


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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.

**EXCEPTIONS**

<platform errors>
An edition file could not be successfully written.

## UnsavedExportedLinks

(For embedding application or document shell only.)
Notifies you whether there are cross-document links that have been established since the last time the draft was saved.

```
XMPBoolean UnsavedExportedLinks  (XMPDraft* draft);
```

`draft` The draft of interest.

*return value* `kXMPTrue` if cross-document links have been established and `kXMPFalse` otherwise.

**DESCRIPTION**

Linking. Public. For use by container apps only. Parts do not call.

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

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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPLinkSource

*Superclasses* `XMPPersistentObject` → `XMPRefCntObject` → `XMPObject`

*Subclasses* none

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.

## Description

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.

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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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: The link always owns a valid storage unit.

## Methods

```
Clear
CloneTo
ContentChanged
Externalize
GetChangeID
GetChangeTime
GetContentStorageUnit
GetLink
IsAutoUpdate
Lock
SetSourcePart
SetAutoUpdate
Unlock
```

Detailed descriptions of these methods follow, in alphabetical order.

### Clear

```
void Clear(
XMPChangeID id,
XMPLinkKey key);
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**PARAMETERS**

| | |
|---|---|
| `id` | The change identification associated with the new empty content. |
| `key` | A valid key returned by Lock(). |

**DESCRIPTION**

Linking. Public. Parts call to remove the link's previous contents.

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

Empties the contents of the link. The key parameter should be obtained by acquiring the lock for this thread.

The change identification argument should be obtained by calling the XMPSession::UniqueChangeID method. After executing this method successfully, the next call to `GetContentStorageUnit` returns a storage unit with no properties.

## CloneTo

```
XMPStorageUnit* CloneTo (XMPDraftKey draftKey,
                         XMPDraft* destDraft,
                         XMPStorageUnit* newSourcePartSU);
```

| | |
|---|---|
| `draftKey` | `XMPDraftKey` identifying the Clone transaction. |
| `destDraft` | A reference to the draft to clone this object into. |
| `newSourcePartSU` | A reference to the storage unit pointer to a part in 'destDraft', or `kXMPNull`. |
| *return value* | A reference to the storage unit containing the persistent representation of this object in the destination draft. |

**DESCRIPTION**

Object Storage



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Public.

Derived class **can** override. Derived class **can** call base class behavior.

Clone this object into the destination draft. If 'newSourcePartSU' is non-nil, it must reference a storage unit in the destination draft; the source part property of the cloned object is set to reference this su.

**EXCEPTIONS**

`kXMPErrCloneNotStarted`
A Clone was started unsuccessfully.

`kXMPErrInvalidDraftKey`
Invalid Clone.

## ContentChanged

```
void ContentChanged (XMPChangeID id);
```

**PARAMETERS**

`id` The change identification associated with the new content.

**DESCRIPTION**

Linking. Public. Called by parts after modifying the link's content.

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

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.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Externalize

```
void Externalize();
```

**DESCRIPTION**

Storage. Public.

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

Externalizes persistent data. After executing this method successfully, the link's persistent storage unit is consistent with the link.

**EXCEPTIONS**

<Storage exceptions>
Storage exceptions may be signalled.

## GetChangeID

```
XMPChangeID GetChangeID();
```

**DESCRIPTION**

Linking. Public. Can be used by Parts containing a destination of this link to determine if new content is available.

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

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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetContentStorageUnit

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

**PARAMETERS**

`key` A valid key returned by Lock().

**RETURN VALUE** A reference to the content storage unit.

**DESCRIPTION**

Linking, Storage. Public.

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

Returns a reference to the XMPStorageUnit containing the content of the link. Before calling this method, you must ensure that the key argument is a valid key returned by Lock(). After executing this method successfully, the returned storage unit is guaranteed to be valid until the key is relinquished by Unlock().

**EXCEPTIONS**

`kXMPErrInvalidLinkKey`

## GetLink

```
XMPLink* GetLink();
```

**DESCRIPTION**

Linking. Public. Called by the source part to obtain the associated XMPLink object.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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.

## IsAutoUpdate

```
XMPBoolean IsAutoUpdate();
```

**DESCRIPTION**

Linking. Public.

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

Returns `kXMPTrue` if updates are automatically exported to other documents when the link's draft is saved. This setting defaults to `kXMPTrue`.

## Lock

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

**PARAMETERS**

| | |
|---|---|
| `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. |
| 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. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**RETURN VALUE** `kXMPTrue` if access is granted and `kXMPFalse` if denied.

**DESCRIPTION**

Thread Safety. Public. Parts must call this method to acquire a key required by other methods.

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

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. After executing this method successfully, it will return `kXMPTrue`, if the current thread has exclusive access to the link's content, and the key parameter is valid.

## SetAutoUpdate

```
void SetAutoUpdate (XMPBoolean automatic);
```

**DESCRIPTION**

Linking. Public.

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

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.

## SetSourcePart

```
void SetSourcePart(
XMPStorageUnit* sourcePartSU);
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`sourcePartSU`
The storage unit for the new source part.

**DESCRIPTION**

Linking

Public.

Derived class **can** override.

Derived class **can** call base class behavior.

Change the part containing the source content of this link to the argument storage unit reference. This method is typically called on a cloned `XMPLinkSource` object during data interchange operations involving linked content. After executing this method successfully, the reference count of the previous source part storage unit is decremented. The reference count of the argument storage unit is incremented.

## Unlock

```
void Unlock (XMPLinkKey key);
```

**PARAMETERS**

`key` A valid key returned by Lock().

**DESCRIPTION**

Thread safety. Public. Parts must call this method when done accessing or modifying the content of a link.

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

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. Before calling this method, you must ensure that the key argument is a valid key returned by Lock(). After executing this



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

method successfully, 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.

**EXCEPTIONS**

`kXMPErrInvalidLinkKey`



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPLinkSpec

*Superclasses* none
*Subclasses* none

This class represents potential links to part content.

## Description

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.

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 receiving 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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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: 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.

## Methods

```
XMPLinkSpec
~XMPLinkSpec
ReadLinkSpec
WriteLinkSpec
```

Detailed descriptions of these methods follow, in alphabetical order.

### XMPLinkSpec

Creates an XMPLinkSpec object.

```
XMPLinkSpec();
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Creation and Deletion. Public. Source parts should call XMPDraft::CreateLinkSpec. The destination part calls this routine prior to reading a link specification value.

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

After this method executes successfully, the object is initialized.

## ~XMPLinkSpec

Destroys the XMPLinkSpec object.

```
~XMPLinkSpec();
```

**DESCRIPTION**

Creation and Deletion. Public.

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

After this method executes successfully, this link specification object doesn't exist.

## ReadLinkSpec

Initialize this link specification from a link spec value in the argument storage unit.

```
void ReadLinkSpec (XMPStorageUnit* su);
```

`su` A storage unit that has been pre-focused to a property containing a link specification value.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Linking. Public.

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

The storage unit must be pre-focused to a property.

Before calling this method, you must ensure that the argument storage unit is pre-focused to a property containing a link spec value. After this method executes successfully, this object is initialized to the link specification in the pre-focused storage unit.

**EXCEPTIONS**

`kXMPErrNoLinkSpecValue`
: Focused property does not contain a link specification value.

`kXMPErrCorruptLinkSpecValue`
: Focused storage unit contains an invalid link specification value.

## WriteLinkSpec

Write this object into a pre-focused storage unit.

```
void WriteLinkSpec (XMPStorageUnit* su);
```

`su`
: A storage unit that has been pre-focused to a property to contain the link specification as a value.

**DESCRIPTION**

Linking. Public. Called by parts to put a link specification on the clipboard or drag-and-drop storage unit.

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

The argument storage unit must be pre-focused to a property.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Before calling this method, you must ensure that the argument storage unit is focused to a property. After this method executes successfully, the argument storage unit contains a link specification value in its focused property.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPMenuBar

*Superclasses* `XMPRefCntObject` → `XMPObject`

*Subclasses* none

XMPMenuBar represents a composite menu bar, made up of menus from the shell and the active part.

## Description

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.

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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Methods

```
XMPMenuBar
~XMPMenuBar
AddMenuBefore
AddMenuLast
Copy
Display
GetMenu
InitMenuBar
Purge
Release
RemoveMenu
```

Detailed descriptions of these methods follow, in alphabetical order.

### XMPMenuBar

Constructs the menu bar object. InitMenuBar must also be called.

```
XMPMenuBar();
```

**DESCRIPTION**

Creation and Deletion. Public. Part editors usually call XMPWindowState::CopyBaseMenuBar() to create a menu bar.

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

After this method executes successfully, instance is in a safe state



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## ~XMPMenuBar

Frees the memory allocated by this instance.

```
~XMPMacMenuBar();
```

**DESCRIPTION**

Creation and Deletion. Public.

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

After this method executes successfully, the memory allocated by this instance is freed, and the instance is no longer usable.

## AddMenuBefore

Inserts a menu before the menu with the specified ID.

```
void AddMenuBefore    (XMPMenuID menuID,
                       XMPPlatformMenu menu,
                       XMPPart* part,
                       XMPMenuID beforeID);
```

| | |
|---|---|
| `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. |
| `beforeID` | The ID of the menu before which to insert the new one. |

**DESCRIPTION**

Insertion and Deletion. Public.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Before you call this method, you must ensure that the specified menu is a valid initialized instance. After calling this method successfully, the collection of menus contains the specified menu.

**EXCEPTIONS**

`kXMPErrOutOfMemory`
Out of memory

## AddMenuLast

Appends a menu to the end of the menu bar.

```
void AddMenuLast  (XMPMenuID menuID,
                   XMPPlatformMenu menu,
                   XMPPart* part);
```

| | |
|---|---|
| `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. |

**DESCRIPTION**

Insertion and Deletion. Public.

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

Before you call this method, you must ensure that the specified menu is a valid initialized instance.

After this method executes successfully, the collection of menus contains the specified menu.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

`kXMPErrOutOfMemory`
Out of memory

## Copy

Clones this object, including its command table.

```
XMPMenuBar* Copy();
```

*return value* A copy of this menu bar

**DESCRIPTION**

Creation and Deletion. Public. Call by WindowState::CopyBaseMenuBar. Parts don't typically call this method.

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

Before you call this method, you must ensure that the specified menubar is a valid and initialized instance.

After this method executes successfully, there is no effect on this instance.

## Display

Installs this menu bar as the current menu bar.

```
void Display();
```

**DESCRIPTION**

Display. Public.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

Before you call this method, you must ensure that the specified object is a valid initialized instance. After this method executes successfully, the current menu bar contains the menus in this instance.

## GetMenu

Returns the platform-specific menu structure (menu handle on Macintosh) for the specified menu ID.

```
XMPPlatformMenu GetMenu (XMPMenuID menu);
```

| | |
|---|---|
| `menu` | The ID of the desired menu. |
| *return value* | The menu handle |

### DESCRIPTION

Accessing. Public.

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

## InitMenuBar

Initializes the menu bar object.

```
InitMenuBar (XMPSession* session,
             XMPPlatformMenuBar menuBar);
```

| | |
|---|---|
| `session` | The OpenDoc session object. |
| `menuBar` | The shared system-wide menu bar handle |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Creation and Deletion. Public. Part editors usually call XMPWindowState::CopyBaseMenuBar()

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

Must be called immediately after new().

## Purge

(For embedding application or document shell only.)
Tries to free up some memory.

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

| | |
|---|---|
| `size` | The amount of memory requested |
| *return value* | The amount of memory freed. |

**DESCRIPTION**

Low Memory. Public.

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

Before you call this method, you must ensure that the specified `XMPSize` object is a constructed and initialized object. After this method executes successfully, some internal structures which are readily rebuilt may have been freed.

## Release

Deletes this instance; decreases its reference count by 1.

```
void Release();
```


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Reference Counting. Public.

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

Overridden to delete this object if the reference count has gone to zero.

## RemoveMenu

Removes the menu with the specified ID.

```
void RemoveMenu (XMPMenuID menu);
```

`menu` The id of the menu to remove

**DESCRIPTION**

Insertion and Deletion. Public.

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

The displayed menu bar is not affected. Before you call this method, you must ensure that the specified menu is a valid initialized instance.

After this method executes successfully, the menu is no longer in this menu bar. The displayed menu bar is not affected.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPMessageInterface

*Superclasses* `XMPObject`

*Subclasses* none

This class encapsulates AppleEvent message sending functionality. It works in concert with the XMPSemanticInterface class.

## Description

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.

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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Methods

```
CreateEvent
CreatePartAddrDesc
CreatePartObjSpec
ProcessSemanticEvent
Purge
Send
```

Detailed descriptions of these methods follow, in alphabetical order.

### CreateEvent

Creates an Apple event.

```
virtual void CreateEvent    (AEEventClass theAEEventClass,
                             AEEventID theAEEventID,
                             AEAddressDesc target,
                             XMPSLong transactionID,
                             AppleEvent* result);
```

`theAEEventClass`
<See AppleEvent Manager documentation>

`theAEEventID`
<See AppleEvent Manager documentation>

`target` <See AppleEvent Manager documentation>

`transactionID`
<See AppleEvent Manager documentation>

`result` Apple event created

*return value* The return id assigned to this event.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Semantic Events. Public.

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

See AppleEvent Manager documentation. The returnID parameter has been removed since OpenDoc will use this to track the origin of a Send.

After this method executes successfully, an event will have been created.

**EXCEPTIONS**

`kXMPOutOfMemory`
Couldn't allocate memory for the event.

others <See AppleEvent Manager documentation>

## CreatePartAddrDesc

Create an address descriptor to be used when sending events to the part in question.

```
XMPVMethod void CreatePartAddrDesc
                                    (AEDesc* theAddressDesc,
                                     XMPPart* thePart);
```

`thePart` The XMPPart* to be addressed.

`theAddressDesc`
The descriptor to be initialized.

**DESCRIPTION**

Semantic Events. Public.

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

After this method executes successfully, the address descriptor is initialized.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

`kXXMPErrNotAValidPart`
XMPPart* was not valid.

## CreatePartObjSpec

Creates an object specifier that can be used to refer to the part in question.

```
XMPVMethod void CreatePartObjSpec   (AEDesc* theObjSpec,
                                     XMPPart* thePart);
```

`theObjSpec` The object specifier to be initialized
`thePart` The part for which an object specifier will be constructed.

**DESCRIPTION**

Semantic Events. Public.

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

Should not be stored persistently. Is only valid while the XMPPart is instantiated.

After this method executes successfully, the object specifier is initialized

**EXCEPTIONS**

`kXXMPErrNotAValidPart`
XMPPart* was not valid.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## ProcessSemanticEvent

(For embedding application or document shell only.)
Dispatches a semantic event to its correct object.

```
virtual XMPBoolean ProcessSemanticEvent
                              (const XMPEventData* theEvent);
```

`theEvent` <See AppleEvent Manager documentation>

*return value* Whether the event was processed.

### DESCRIPTION

Semantic Events. Public. Probably only called from an XMPDispatchModule.

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

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.

After this method executes successfully, the event will have been dispatched.

### EXCEPTIONS

`kXMPEResolveError`
Couldn't determine correct object to dispatch to.

## Purge

Frees memory.

```
XMPSize Purge  (XMPSize howMuch);
```

`howMuch` Amount of memory requested to be purged.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

*return value* Amount of memory that was released.

**DESCRIPTION**

Semantic Events, XMPObject. Public.

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

After this method executes successfully, memory may have been freed.

## Send

Sends an Apple event.

```
virtual XMPSShort Send  (XMPPart* thePart,
                         AppleEvent* theAppleEvent,
                         AppleEvent* reply,
                         AESendMode sendMode,
                         AESendPriority sendPriority,
                         XMPULong timeOutInTicks,
                         XMPIdleProcPtr idleProc,
                         XMPEventFilterProcPtr filterProc);
```

`thePart` The sending XMPPart or `kXMPAppShell` if the sender is the application shell / document.

`theAppleEvent`
<See AppleEvent Manager documentation>

`reply` <See AppleEvent Manager documentation>

`sendMode` <See AppleEvent Manager documentation>

`sendPriority`
<See AppleEvent Manager documentation>

`timeOutInTicks`
<See AppleEvent Manager documentation>

`idleProc` <See AppleEvent Manager documentation>



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`filterProc` <See AppleEvent Manager documentation>

**DESCRIPTION**

Semantic Events. Public.

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

See AppleEvent Manager documentation. The XMPPart* parameter specifies the sender of the event.

Before calling this method, you must have a valid AppleEvent created by XMPMessageInterface::CreateEvent is required.

**EXCEPTIONS**

`kXMPEInvalidPart`
: Invalid part.

`kXMPEInvalidSemanticEvent`
: Invalid AppleEvent (not created with XMPMessageInterface::CreateEvent).

others
: <See AppleEvent Manager documentation>



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPNameResolver

*Superclasses* `XMPObject`

*Subclasses* none

This class encapsulates structured name or object specifier resolution functionality. It works in concert with the XMPSemanticInterface class.

## Description

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.

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.

## Methods

`CallObjectAccessor`

`CreateSwapToken`

`CreateToken`

`DisposeToken`



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

```
GetContainingFrame
GetContextInfo
Purge
Resolve
```

Detailed descriptions of these methods follow, in alphabetical order.

## CallObjectAccessor

See AppleEvent Manager documentation for AECallObjectAccessor.

```
virtual void CallObjectAccessor
                                (XMPPart* part,
                                 DescType desiredClass,
                                 XMPOSLToken* containerToken,
                                 DescType containerClass,
                                 DescType keyForm,
                                 AEDesc* keyData,
                                 XMPOSLToken* token);
```

`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.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Content Model. Public.

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

See AppleEvent Manager documentation for AECallObjectAccessor. part is the part whose accessor is to be called.

After this method executes successfully, whatever side effects the invocation of the accessor produces will have occurred.

**EXCEPTIONS**

See AppleEvent Mgr documentation.

## CreateSwapToken

Construct an AppleEvent token that encodes an XMPFrame.

```
virtual CreateSwapToken (XMPOSLToken* token,
                         XMPFrame* frame);
```

`token` The token constructed.

`frame` The XMPFrame represented by this token

**DESCRIPTION**

Content Model. Public.

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

After this method executes successfully, a new token is constructed.

**EXCEPTIONS**

`kXMPErrOutOfMemory`
Out of Memory



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## CreateToken

Initialize an AppleEvent token that has an encoded XMPPart context.

```
virtual CreateToken  (XMPOSLToken* token
                      DescType type,
                      XMPPart* inPart);
```

| | |
|---|---|
| `token` | The initialized token. |
| `type` | The DescType for the AEDesc of this token |
| `inPart` | The XMPPart* specifying the root container for this token |

**DESCRIPTION**

Content Model. Public.

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

After this method executes successfully, a new token is constructed.

**EXCEPTIONS**

`kXMPErrOutOfMemory`
Not enough memory to initialize token.

## DisposeToken

See AppleEvent Object Model documentation.

```
virtual void DisposeToken  (XMPOSLToken* theToken);
```

| | |
|---|---|
| `theToken` | The token to dispose. |

**DESCRIPTION**

Content Model. Public.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

After this method executes successfully, the token is destroyed.

#### EXCEPTIONS

`kXMPErrInvalidPart`
Token does not contain a valid part description.

## GetContainingFrame

Returns the frame that was used to find a part during object specifier resolution.

```
XMPFrame* GetContainingFrame();
```

*return value* The containing frame of this part.

#### DESCRIPTION

Semantic Events. Public.

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

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.

Before calling this method, you must ensure that CreateSwapToken must have been called, in order for a valid XMPFrame* to be returned.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetContextInfo

(For embedding application or document shell only.)
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.

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

`frame` The last destination frame for the AppleEvent.

`part` The last destination part for the AppleEvent.

**DESCRIPTION**

Content Model. 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.

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

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.

To call this method, an AppleEvent and/or the resolution of the object specifier in its direct parameter must have failed.

**EXCEPTIONS**

`kXMPErrInvalidPart`
Not a valid part


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Purge

Free unused memory.

```
XMPSize Purge  (XMPSize howMuch);
```

`howMuch` The memory requested to be freed.
*return value* The amount of memory that was freed.

**DESCRIPTION**

Content Model, XMPObject. Public.

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

After this method executes successfully, some memory may have been freed.

## Resolve

Resolves an object specifier, producing a token.

```
virtual XMPPart* Resolve   (XMPObjectSpec* theObject,
                            XMPOSLToken* token,
                            XMPPart* contextPart);
```

`theObject` The object specifier to be resolved.
`token` The final token produced by the resolution.
`contextPart` The part from which the call is being made.

**DESCRIPTION**

Content Model. Public.

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

See AppleEvent Object Model documentation for AEResolve. A new type, XMPOSLToken is used for the token instead of an AEDesc. The fields of the


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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.

After this method executes successfully, a new token will have been created for the object described in theObject. Many part-specific routines will be called back to.

**EXCEPTIONS**

`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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPNameSpace

*Superclasses* `XMPObject`
*Subclasses* none

This class represents a name space.

## Description

Objects of this class are created by member functions of the XMPNameSpaceManager class. The platform vendor will implement. XMPNameSpace is a subclass of XMPObject.

This class allows registering XMPPtrs with three different types of keys: XMPISOStr, XMPOSType and XMPSLong. Values are hashed for quick lookup.

Invariants: There is always a valid hash table instantiated to hold the key/value pairs. It may be empty.

## Methods

`Exists`
`GetName`
`GetValue`
`Purge`
`ReadFromFile`
`ReadFromStorage`
`Register`
`Unregister`
`WriteToFile`
`WriteToStorage`


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Detailed descriptions of these methods follow, in alphabetical order.

## Exists

Returns boolean for the existance of the given key.

```
virtual XMPPtr Exists   (XMPISOStr key);
```

`key` The key for the entry requested.

*return value* Boolean. True if key exists.

**DESCRIPTION**

Name Binding. Public.

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

## GetName

Return the ISO string assigned to this object at initialization time.

```
virtual XMPISOStr GetName();
```

*return value* name of this name space.

**DESCRIPTION**

Name Binding. Public.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetValue

Returns the XMPPtr corresponding to the given key

```
virtual XMPPtr GetValue (XMPISOStr key);
```

| | |
|---|---|
| `key` | The key for the entry requested. |
| *return value* | XMPPtr. The entry. |

**DESCRIPTION**

Name Binding. Public.

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

## Purge

Frees up unused memory.

```
XMPSize Purge  (XMPSize howMuch);
```

| | |
|---|---|
| `howMuch` | The amount of memory to purge. |
| *return value* | The amount of memory that was freed. |

**DESCRIPTION**

Name Binding, XMPObject. Public.

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

After this method executes successfully, some memory may be freed for use.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## ReadFromFile

Reads the name space from the file into this `XMPNameSpace`.

```
void ReadFromFile (XMPFileRefNum view);
```

`view` The storage unit from which to read the data.

**DESCRIPTION**

Name Binding. Public.

Derived class **cannot** override. Derived class **cannot** call base class behavior, during derived class behavior.

Assumes that the data was written with `WriteToFile`. If the name of the name space doesn't match the name that was stored in the stream, then an exception will be thrown. It is assumed that the file's cursor has been placed at the beginning of the data stream.

Before you call this method, the data must have been streamed with `WriteToFile`. After this method executes successfully, the name space will contain entries corresponding to the streamed data.

**EXCEPTIONS**

`kXMPErrInvalidName`
The stored name does not match that of the name space.

## ReadFromStorage

Reads the name space from storage into this `XMPNameSpace`.

```
void ReadFromStorage (XMPStorageUnitView* view);
```

`view` A reference to the storage unit from which to read the data.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Name Binding. Public.

Derived class **cannot** override. Derived class **cannot** call base class behavior, during derived class behavior.

Assumes that the data was written with `WriteToStorage`. If the name of the name space doesn't match the name that was stored in the stream, then an exception will be thrown.

Before you call this method, the data must have been streamed with `WriteToStorage`. After this method executes successfully, the name space will contain entries corresponding to the streamed data.

**EXCEPTIONS**

`kXMPErrInvalidName`
The stored name does not match that of the name space.

## Register

Register an XMPPtr given a key.

```
virtual void Register   (XMPISOStr key,
                         XMPPtr value,
                         XMPULong valueLength);
```

`key` The key to use for this XMPPtr

`value` Client-defined pointer value.

`valueLength`
Size of value

**DESCRIPTION**

Name Binding. Public.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

After this method executes successfully, the XMPPtr is registered under the key.

**EXCEPTIONS**

`kXMPErrOutOfMemory`
Not enough memory.

`kXMPErrWrongType`
Request for type other than type with which this name space was originally created.

`kXMPErrInvalidValue`
value was `kXMPNull`.

## Unregister

Removes the given entry from the XMPNameSpace.

```
virtual void Unregister (XMPISOStr key);
```

`key` Key for the entry to be removed.

**DESCRIPTION**

Name Binding. Public.

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

Before you call this method, the given entry must have been registered. No error is signalled, however, if the key does not already exist. After this method executes successfully, the given entry will be removed.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## WriteToFile

Writes the name space out as a stream to the file.

```
void WriteToFile  (XMPStorageUnitView* view);
```

`view` The storage unit to which to write the data.

**DESCRIPTION**

Name Binding. Public.

Derived class **cannot** override. Derived class **cannot** call base class behavior, during derived class behavior.

This stream may be read back in by `ReadFromFile`.

After this method executes successfully, the data will be written out to the given file.

**EXCEPTIONS**

`misc` File system errors and/or storage errors..

## WriteToStorage

Writes the name space out as a stream to the storage unit view.

```
void WriteToStorage  (XMPStorageUnitView* view);
```

`view` A reference to the storage unit from which to read the data.

**DESCRIPTION**

Name Binding. Public.

Derived class **cannot** override. Derived class **cannot** call base class behavior, during derived class behavior.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

This stream may be read back in by `ReadFromStorage`.

After this method executes successfully, the data will be written out to storage.

**EXCEPTIONS**

| | |
|---|---|
| `misc` | File system errors and/or storage errors. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPNameSpaceManager

*Superclasses* `XMPObject`

*Subclasses* none

This class represents a  a bottleneck for the creation of XMPNameSpaces to be used by the XMPBinding object.

## Description

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.

This class  allows inheritance of XMPNameSpaces so that a search can take place in more than one XMPNameSpace.

Invariants: A valid list of XMPNameSpaces (possibly empty) exists.

## Methods

`CreateNameSpace`

`DeleteNameSpace`

`HasNameSpace`

`Purge`

Detailed descriptions of these methods follow, in alphabetical order.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## CreateNameSpace

Creates an XMPNameSpace object with specified parameters.

```
virtual XMPNameSpace* CreateNameSpace
                              (XMPISOStr spaceName,
                               XMPNameSpace* inheritsFrom,
                               XMPULong numExpectedEntries);
```

`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.

*return value* Reference to XMPNameSpace created.

### DESCRIPTION

Name Binding. Public.

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

An exception is signalled if an XMPNameSpace with the given name already exists.

After this method executes successfully, a new XMPNameSpace object has been created and recorded in a list.

### EXCEPTIONS

`kXMPErrOutOfMemory`
Not enough memory

`kXMPErrKeyAlreadyExists`
An XMPNameSpace with that name already exists.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## DeleteNameSpace

Delete specified XMPNameSpace object.

```
virtual void DeleteNameSpace  (XMPNameSpace* theNameSpace);
```

`theNameSpace`
The XMPNameSpace to be destroyed

**DESCRIPTION**

Name Binding. Public.

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

After this method executes successfully, the specified XMPNameSpace object will have been destroyed.

## HasNameSpace

Finds out whether a given XMPNameSpace exists.

```
virtual XMPBoolean HasNameSpace  (XMPISOStr spaceName);
```

`spaceName` the name of the XMPNameSpace to find.
*return value* If true, the XMPNameSpace exists.

**DESCRIPTION**

Name Binding. Public.

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


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Purge

Free unused memory

```
XMPSize Purge  (XMPSize howMuch);
```

| | |
|---|---|
| `howMuch` | The amount request to free |
| *return value* | The amount of memory that was able to be freed. |

**DESCRIPTION**

Name Binding, XMPObject. Public.

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

After this method executes successfully, some memory may be freed.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPObject

*Superclasses* none

*Subclasses* `XMPArbitrator`, `XMPCanvas`, `XMPClipboard`, `XMPDispatcher`, `XMPDragAndDrop`, `XMPFacet`, `XMPMessageInterface`, `XMPNameResolver`, `XMPNameSpace`, `XMPNameSpaceManager`, `XMPRefCntObject`, `XMPSession`, `XMPShape`, `XMPStorageSystem`, `XMPStorageUnitView`, `XMPTransform`, `XMPTranslation`, `XMPUndo`, `XMPWindowState`

This class is the common base class for most XMP classes.

## Description

It provides a general mechanism for extensibility through Extension objects. XMPObject also provides a general memory recovery system with the Purge method.

XMPObject does not have any real operation in it, for the most part it serves as a template.

Invariants: 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.

## Methods

`XMPObject`

`~XMPObject`

`GetExtension`

`HasExtension`



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

```
InitObject
Purge
ReleaseExtension
```

Detailed descriptions of these methods follow, in alphabetical order.

## XMPObject

Constructor of the class.

```
XMPObject();
```

**DESCRIPTION**

None.. Public.

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

After this method executes successfully, this is an uninitialized XMPObject object.

## ~XMPObject

Destructor of the class.

```
~XMPObject();
```

**DESCRIPTION**

None.. Public.

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


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetExtension

GetExtension is intended to return the extension object with name extensionName.

```
XMPExtension* GetExtension (XMPName* extensionName);
```

`extensionName`
name of extension to get

*return value* the requested Extension

**DESCRIPTION**

Extensions. Public.

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

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.

Before you call this method, this object must have extension extensionName

**EXCEPTIONS**

`kXMPErrInvalidExtension`
not an extension that this object knows about

## HasExtension

HasExtension returns whether or not the object has an extension with the name extensionName.

```
XMPBoolean HasExtension (XMPName* extensionName);
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`extensionName`
name of extension to look for

*return value* whether or not this object has the given extension

**DESCRIPTION**

Extensions. Public.

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

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.

Before you call this method, extensionName must be an Extension that this object has.

## InitObject

InitObject performs initialization operations for an object.

```
void   InitObject();
```

**DESCRIPTION**

Initialization. Public.

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

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.

After this method executes successfully, this is an initialized XMPObject object.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Purge

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.

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

| | |
|---|---|
| `size` | the number of bytes to try to purge |
| *return value* | the number of bytes actually purged |

**DESCRIPTION**

Memory Management. Public.

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

The number of bytes actually freed should be returned. XMPObject does not allocate any memory and always return 0 bytes purged.

After this method executes successfully, at least size or as many as possible bytes have been freed up by the object

## ReleaseExtension

ReleaseExtension is intended to release the extension object passed in.

```
void   ReleaseExtension (XMPExtension* extension);
```

| | |
|---|---|
| `extension` | the extension to be released |

**DESCRIPTION**

Extensions. Public.

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

If the extension passed in is not one which was acquired from the object using GetExtension then an exception is raised.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

XMPObject has no extensions and always raises an exception. Derived classes should expect an extension which was previously acquired using GetExtension, and release it.

Before you call this method, this object must have given out the given extension. After this method executes successfully, the given extension is no longer guaranteed to be valid.

**EXCEPTIONS**

`kXMPErrInvalidExtension`
not an extension that this object knows about.



**Draft. Preliminary. © Apple Computer, Inc. 6/20/94**

# XMPPersistentObject

*Superclasses* `XMPRefCntObject` → `XMPObject`

*Subclasses* `XMPFrame`, `XMPLink`, `XMPLinkSource`, `XMPPart`

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.

## Description

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.

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: InitPersistentObjectFromStorage is called only once on XMPPersistentObject, when it is first created.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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 `kXMPIntlText`

## Methods

`XMPPersistentObject`
`~XMPPersistentObject`
`Externalize`
`GetID`
`GetStorageUnit`
`InitPersistentObject`
`InitPersistentObjectFromStorage`

Detailed descriptions of these methods follow, in alphabetical order.

### XMPPersistentObject

Creates a persistent object.

```
XMPPersistentObject  (XMPStorageUnit* storageUnit);
```

**DESCRIPTION**

None.. 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.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

set fSU = storageUnit

#### Postconditions

fSU = storageUnit

## ~XMPPersistentObject

release the object's storageUnit.

```
~XMPPersistentObject();
```

#### DESCRIPTION

None.. Public. Parts and Container Apps do not call this method directly. But whenever a persistent object is deleted, this method is called.

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

## Externalize

A derived class should write out its persistent state to the properties and values it created during InitXXX.

```
void Externalize();
```

#### DESCRIPTION

ObjectStorage. Public.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

Updates the moddate and modifiedUser properties of the persistent object.

## GetID

Returns the ID of this object.

```
XMPID  GetID();
```

*return value* ID of this object.

**DESCRIPTION**

ObjectStorage. Public.

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

Note that this ID is not persistent. Therefore, the same persistent object may have a different ID in every session.

## GetStorageUnit

returns the XMPStorageUnit associated with the persistent object.

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

*return value* XMPStorageUnit associated with this object.

**DESCRIPTION**

ObjectStorage. Public.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Derived class **cannot** override. Derived class **can** call base class behavior, during derived class behavior.

## InitPersistentObject

Initializes a persistent object.

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

`storageUnit`
the storageUnit of this persistent object

**DESCRIPTION**

Object Storage. 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.

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

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.

## InitPersistentObjectFromStorage

Initializes a persistent object from stored data.

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

`storageUnit`
XMPStorageUnit of this object



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

ObjectStorage. 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.

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

Initializes the internal fields of this object.

A derived class must call this method in its InitXXXFromStorage method.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPPlatformTypeList

*Superclasses* none

*Subclasses* none

XMPPlatformTypeList is an ordered collection class of XMPPlatformType objects.

## Description

XMPPlatformTypeList is an ordered 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 XMPPlatformTypeList. Subclassing of XMPPlatformTypeList is not anticipated.

The main client of this class are XMPClipboard and parts. The XMPClipboard class uses XMPPlatformTypeList as parameters to its functions. Parts can use XMPPlatformTypeList to store a set of Platform Types for repeated use.

XMPPlatformTypeList uses OpenDoc's private collection class.

Invariants: Every item in a XMPPlatformTypeList object is unique. Therefore, a XMPPlatformTypeList object contains a set of different XMPPlatformTypes.

A XMPPlatformTypeList object can contain zero or more XMPPlatformTypes. In other words, the type set can either be empty or non-empty.

## Methods

`XMPPlatformTypeList`

`~XMPPlatformTypeList`

`Add`

`AddLast` Adds the argument platform type to the end of the list.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `Contains` | |
| `Count` | |
| `CreatePlatformTypeListIterator` | |
| `Difference` | |
| `InitPlatformTypeList` | Initializes data members and sets the list to a copy of the argument list. |
| `Remove` | |
| `Union` | |

Detailed descriptions of these methods follow, in alphabetical order.

## XMPPlatformTypeList

Class instance constructor.

```
XMPPlatformTypeList();
```

**DESCRIPTION**

Creation and Deletion. Public.

Derived class **cannot** override. Derived class **must** call base class behavior, before derived class behavior. After this method executes successfully, members are initialized to a known state.

**EXCEPTIONS**

`kXMPErrOutOfMemory`
Could not construct this instance.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## ~XMPPlatformTypeList

Class instance destructor.

```
~XMPPlatformTypeList();
```

**DESCRIPTION**

Creation and Deletion. Public.

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

Before calling this method, you must ensure that a list has been created.

## Add

Adds the argument platform type to the set.

```
void Add(
XMPPlatformType type);
```

`type` Platform type to be added.

**DESCRIPTION**

None.. Public.

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

If the platform type is already present in the set, no action is taken.

After this method executes successfully, the set contains the argument platform type.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

`kXMPErrOutOfMemory`
Could not add the argument platform type to the set.

## AddLast

The `AddLast` method adds the argument platform type to the end of the list.

```
void AddLast(
XMPPlatformType type);
```

`type` The platform type to be added.

**DESCRIPTION**

None.

Public.

Derived class **can** override.

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

If the platform type is already present in the list, no action is taken.

### Pre conditions

The list has been initialized.

### Post conditions

The list contains the argument platform type.

**EXCEPTIONS**

| | |
|---|---|
| kXMPErrOutOfMemory | Could not add the argument platform type to the list. |


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Contains

Test this set for the presence of the argument platform type.

```
XMPBoolean Contains  (XMPPlatformType type);
```

| | |
|---|---|
| `type` | A platform type to test for inclusion in this set. |
| *return value* | `kXMPTrue` if the argument platform type is in the set and `kXMPFalse` otherwise. |

**DESCRIPTION**

None. Public.

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

Before calling this method, you must ensure that the list has been inititialized.

## Count

This function returns the number of platform types in the set.

```
XMPULong Count();
```

| | |
|---|---|
| *return value* | Number of platform types in the set. |

**DESCRIPTION**

None.. Public.

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

If the set is empty, zero is returned.

Before calling this method, you must ensure that the list has been inititialized.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## CreatePlatformTypeListIterator

Return an iterator for this set.

```
XMPPlatformTypeListIterator*
CreatePlatformTypeListIterator();
```

*return value* An iterator for this set.

**DESCRIPTION**

Iteration. Public.

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

The client should dispose of the iterator using the delete operator.

Before calling this method, you must ensure that the list has been inititialized.

**EXCEPTIONS**

`kXMPErrOutOfMemory`
Could not create the iterator.

## Difference

Remove the specified set of platform types from this object.

```
void Difference   (XMPPlatformTypeList* typeList);
```

`typeList` The set of platform types to be removed.

**DESCRIPTION**

None.. Public.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

Modify this object to be the set difference between this object and the argument set; that is, exclude all platform types in the argument typeList from this set. Platform types in the argument set which are absent from this set are ignored.

After this method executes successfully, this set does not contain any platform types in the argument set.

## InitPlatformTypeList

The `InitPlatformTypeList` method initializes data members and sets the list to a copy of the argument list.

```
void InitPlatformTypeList(
XMPPlatformTypeList* initList);
```

`initList` A reference to a list of XMPPlatformTypes from which to initialize this object. A copy of each item in 'initList' is added to this list, in the same order as in 'initList'. A kXMPNULL value results in an intially empty list.

**DESCRIPTION**

Initialization

Public.

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

### Pre conditions

The list has been created.

### Post conditions

The list has been initialized to copy of the argument list.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrOutOfMemory` | Could not initialize the list. |

## Remove

Removes the specified platform type from the set.

```
void Remove (XMPPlatformType type);
```

`type` Platform type to be removed.

**DESCRIPTION**

None.. Public.

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

If the type is not in the set, no action is taken.

After this method executes successfully, the argument platform type is not in the set.

## Union

Add the specified set of platform types to this object.

```
void Union  (XMPPlatformTypeList* typeList);
```

`typeList` The set of platform types to be added.

**DESCRIPTION**

None.. Public.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

Removes the argument platform type from the set. If the type is not in the set, no action is taken.. Public.

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

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 typeList not already present in this set. If an exception is thrown, this set is left in an indeterminate state.

After this method executes successfully, this set contains every type in the argument set.

**EXCEPTIONS**

`kXMPErrOutOfMemory`
Cannot form the set union with the argument set.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPPlatformTypeListIterator

*Superclasses* none

*Subclasses* none

XMPPlatformTypeListIterator is an iterator for the class XMPPlatformTypeList.

## Description

XMPPlatformTypeListIterator has no base class.

This class is used to iterate over the elements of a XMPPlatformTypeList instance in order. Iteration is forward only, from first to last. The iteration can be restarted at any time by calling the First method. If the XMPPlatformTypeList 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 XMPPlatformTypeListIterator by calling the CreatePlatformTypeListIterator method of the XMPPlatformTypeList instance to be iterated over. The iterator should be disposed of using the delete operator.

Subclassing of XMPPlatformTypeListIterator is not anticipated.

The implementation of XMPPlatformTypeListIterator uses OpenDoc's private collection class.

## Methods

```
~XMPPlatformTypeListIterator
First
IsNotComplete
Next
```

Detailed descriptions of these methods follow, in alphabetical order.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## ~XMPPlatformTypeListIterator

Class instance destructor.

```
~XMPPlatformTypeListIterator();
```

**DESCRIPTION**

Creation and Deletion. Public.

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

Before calling this method, you must ensure that the iterator has been created.

## First

Initiates iteration and returns the first XMPPlatformType object in the list.

```
XMPPlatformType First();
```

*return value* The first XMPType object in the list. If the list is empty kXMPNULL is returned, although the client should test for completion by calling IsNotComplete.

**DESCRIPTION**

Iteration. Public.

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

After creating an iterator using XMPPlatformTypeList::CreatePlatformTypeListIterator, 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.

After this method executes successfully, ther first item in the list is returned, or `kXMPNull` if the list is empty.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

`kXMPErrIteratorOutOfSync`
The XMPPlatformTypeList instance was changed during the iteration.

## IsNotComplete

Test if iteration is complete.

```
XMPBoolean IsNotComplete();
```

*return value* Returns `kXMPTrue` if the last call to First or Next returned a valid XMPPlatformType element, and `kXMPFalse` otherwise.

**DESCRIPTION**

Iteration. Public.

Derived class **can** override. Derived class **can** call base class behavior, during derived class behavior. Before calling this method, you must ensure that the `First` method has been called to start the iteration.

## Next

Returns the next element from the ordered list.

```
XMPPlatformType Next();
```

*return value* The item in the ordered list after the last item returned by First or Next since the iteration was started. If the iteration is complete kXMPNULL is returned, although the client should call IsNotComplete to test for completion.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Iteration. Public.

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

Next can be called repeatedly to iterate through the remaining items in the list. IsNotComplete returns kXMPFalse if the preceeding call to Next did not return an item from the list.

Before you call this method, you must ensure that First has been called to start the iteration.

**EXCEPTIONS**

`kXMPErrIteratorOutOfSync`
: The XMPPlatformTypeList instance was changed during the iteration.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPRefCntObject

*Superclasses* `XMPObject`

*Subclasses* `XMPContainer`, `XMPDocument`, `XMPDraft`, `XMPMenuBar`, `XMPPersistentObject`, `XMPStorageUnit`, `XMPWindow`

This is the base class for all XMP classes that require reference counting.

## Description

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.

Invariants: 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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Methods

XMPRefCntObject
~XMPRefCntObject
IncrementRefCount
InitRefCntObject
Release

Detailed descriptions of these methods follow, in alphabetical order.

### XMPRefCntObject

Creates the object.

```
XMPRefCntObject();
```

**DESCRIPTION**

None.. Public.

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

### ~XMPRefCntObject

Deletes this object.

```
~XMPRefCntObject();
```

**DESCRIPTION**

Cleanup. Public.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

## IncrementRefCount

Increment the reference count of this object by 1.

```
void IncrementRefCount();
```

**DESCRIPTION**

ObjectStorage. Public.

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

Before you call this method, reference count of this object = x, where x ≥ 1.
After this method executes successfully, reference count of this object = x+1

## InitRefCntObject

Initializes this object and sets its refCount to 1.

```
void InitRefCntObject();
```

**DESCRIPTION**

None. Public.

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

After this method executes successfully, refCount of this object == 1.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Release

Decrements its reference count by 1.

```
void Release();
```

### DESCRIPTION

ObjectStorage. Public.

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

Before you call this method, reference count of this object = x, where x>0. After this method executes successfully, reference count of this object = x-1

### EXCEPTIONS

`kXMPErrZeroRefCount`
: The reference count cannot be decremented because the reference count is already 0.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPSemanticInterface

*Superclasses* `XMPExtension`

*Subclasses* none

This class encapsulates the functionality necessary to install semantic event handlers and object accessors.

## Description

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.

XMPSemanticInterface interacts primarily with XMPMessageInterface, XMPNameResolver, XMPSession, and XMPPart.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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 Apple events.

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 accommodate 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.

## Methods

`XMPSemanticInterface`

`~XMPSemanticInterface`

`GetCoercionHandler`

`GetEventHandler`

`GetObjectAccessor`

`GetOSLSupportFlags`

`GetSpecialHandler`

`InitSemanticInterface`



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

```
InstallAdjustMarksProc
InstallCoercionHandler
InstallCompareProc
InstallCountProc
InstallDisposeTokenProc
InstallEventHandler
InstallGetErrDescProc
InstallGetMarkTokenProc
InstallMarkProc
InstallObjectAccessor
InstallSpecialHandler
RemoveCoercionHandler
RemoveEventHandler
RemoveObjectAccessor
RemoveSpecialHandler
SetOSLSupportFlags
```

Detailed descriptions of these methods follow, in alphabetical order.

## XMPSemanticInterface

Initialize the object.

```
XMPSemanticInterface(
XMPObject* base);
```

`base` the object whose XMPSemanticInterface this is.

**DESCRIPTION**

Semantic Events. Public. Only parts should create XMPSemanticInterfaces. The application shell should obtain its copy through the XMPSession.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Derived class **cannot** override. Derived class **cannot** call base class behavior.
After this method executes successfully, the object is instantiated.

## ~XMPSemanticInterface

Destroy the object

```
XMPVMethod ~XMPSemanticInterface();
```

**DESCRIPTION**

Semantic Events. Public. See notes for XMPMacSemanticInterface
Derived class **cannot** override. Derived class **cannot** call base class behavior.
After this method executes successfully, the object is destroyed.

## GetCoercionHandler

See AppleEvent Manager documentation.

```
virtual void GetCoercionHandler
                                (DescType fromType,
                                 DescType toType,
                                 XMPCoercionHandler* handler,
                                 XMPSLong* handlerRefcon,
                                 XMPBoolean* fromTypeIsDesc);
```

| | |
|---|---|
| `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. |


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`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.

**DESCRIPTION**

Semantic Events. Public.

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

**EXCEPTIONS**

`memFullErr` Not enough room in heap zone

`errAEHandlerNotFound`
No coercion handler found

## GetEventHandler

See AppleEvent Manager documentation.

```
virtual void GetEventHandler
                          (AEEventClass theAEEventClass,
                           AEEventID theAEEventID,
                           XMPEventHandlerProcPtr* handler,
                           XMPSLong* handlerRefcon);
```

`theAEEventClass`
The event class for the Apple event or events to be dispatched for this entry.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`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.

#### DESCRIPTION

Semantic Events. Public.

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

#### EXCEPTIONS

`errAEHandlerNotFound`
: No handler found for an Apple event

## GetObjectAccessor

See AppleEvent Manager documentation.

```
virtual void GetObjectAccessor
                          (DescType desiredClass,
                           DescType containerType,
                           XMPAccessorProcPtr* theAccessor,
                           XMPSLong* accessorRefcon);
```

`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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`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.

**DESCRIPTION**

Content Model. Public.

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

**EXCEPTIONS**

See AppleEvent Manager documentation.

## GetOSLSupportFlags

Get the flags indicating amount of support for the OSL.

```
virtual XMPShort GetOSLSupportFlags();
```

*return value* flags.The flags indicating amount of support for the OSL.

**DESCRIPTION**

Semantic Events. Public.

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


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetSpecialHandler

See AppleEvent Manager documentation.

```
virtual void GetSpecialHandler(
AEKeyword functionClass,
XMPSpecialHandlerPtr* handler);
```

`functionClass`
: The keyword for the special handler that is installed.

`handler`
: A pointer to the special handler.

**DESCRIPTION**

Semantic Events. Public.

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

**EXCEPTIONS**

`memFullErr` Not enough room in heap zone

`errAENotASpecialFunction`
: Wrong keyword for a special function

## InitSemanticInterface

Initialize the object.

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

`part`
: The part to which this XMPSemanticInterface belongs. Ordinarily, XMPSemanticInterface objects can be shared among several parts. However, you must create one XMPSemanticInterface object per part if your part wants to install a predispatch proc. In this case, the parameter part is used to identify the part whose predispatch should be called.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Semantic Events. Public. Only parts should instantiate these. The application shell can get its copy through the XMPSession.

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

After this method executes successfully, fEventHandlerTable, fObjectAccessorTable, and fCoercionHandlerTable have been successfully instantiated.

**EXCEPTIONS**

`kXMPErrOutOfMemory`
Not enough room to allocate internal structures.

## InstallAdjustMarksProc

Install a reference to a callback function that gets called to unmark objects that have previously been marked.

```
InstallAdjustMarksProc  (XMPMarkPtr adjustMarksProc,
                         XMPSLong refCon);
```

`countProc` the procedure pointer

`refCon` a reference constant

**DESCRIPTION**

Semantic Events. Public.

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

After this method executes successfully, the callback function and refCon are stored and available for future use by the Semantic Events subsystem.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## InstallCoercionHandler

See AppleEvent Manager documentation.

```
virtual void InstallCoercionHandler
                                (DescType fromType,
                                 DescType toType,
                                 XMPCoercionHandler handler,
                                 XMPSLong handlerRefcon,
                                 XMPBoolean fromTypeIsDesc);
```

`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.

### DESCRIPTION

Semantic Events. Public.

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

After this method executes successfully, the function will be installed in the coercion handler table.

### EXCEPTIONS

`memFullErr` Not enough room in heap zone



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## InstallCompareProc

Install a reference to a callback function that compares objects.

```
InstallCompareProc   (XMPCompareProcPtr compareProc,
                      XMPSLong refCon);
```

`compareProc`
the procedure pointer

`refCon` a reference constant

**DESCRIPTION**

Semantic Events. Public.

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

After this method executes successfully, the callback function and refCon are stored and available for future use by the Semantic Events subsystem.

## InstallCountProc

Install a reference to a callback function that counts objects.

```
InstallCountProc  (XMPCountProcPtr countProc,
                   XMPSLong refCon);
```

`countProc` the procedure pointer

`refCon` a reference constant

**DESCRIPTION**

Semantic Events. Public.

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


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

After this method executes successfully, the callback function and refCon are stored and available for future use by the Semantic Events subsystem.

## InstallDisposeTokenProc

Install a reference to a callback function that disposes tokens.

```
InstallDisposeTokenProc
                  (XMPDisposeTokenProcPtr disposeTokenProc,
                   XMPSLong refCon);
```

`countProc` the procedure pointer

`refCon` a reference constant

**DESCRIPTION**

Semantic Events. Public.

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

After this method executes successfully, the callback function and refCon are stored and available for future use by the Semantic Events subsystem.

## InstallEventHandler

See AppleEvent Manager documentation.

```
virtual void InstallEventHandler
                             (AEEventClass theAEEventClass,
                              AEEventID theAEEventID,
                              XMPEventHandlerProcPtr handler,
                              XMPSLong handlerRefcon);
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`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.

**DESCRIPTION**

Semantic Events. Public.

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

After this method executes successfully, the function will be installed in the event handler table.

**EXCEPTIONS**

`memFullErr` Not enough room in heap zone

## InstallGetErrDescProc

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.

```
InstallGetErrDescProc   (XMPMarkPtr getErrDescProc,
                         XMPSLong refCon);
```

`countProc` the procedure pointer

`refCon` a reference constant


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Semantic Events. Public.

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

After this method executes successfully, the callback function and refCon are stored and available for future use by the Semantic Events subsystem.

## InstallGetMarkTokenProc

Install a reference to a callback function that gets a token that the client will use to mark set of objects.

```
InstallGetMarkTokenProc
                  (XMPGetMarkTokenProcPtr getMarkTokenProc,
                   XMPSLong refCon);
```

`countProc` the procedure pointer

`refCon` a reference constant

**DESCRIPTION**

Semantic Events. Public.

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

After this method executes successfully, the callback function and refCon are stored and available for future use by the Semantic Events subsystem.

## InstallMarkProc

Install a reference to a callback function that gets called repeatedly to mark a set of objects.

```
InstallMarkProc   (XMPMarkPtr markProc,
                   XMPSLong refCon);
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`countProc` the procedure pointer

`refCon` a reference constant

**DESCRIPTION**

Semantic Events. Public.

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

After this method executes successfully, the callback function and refCon are stored and available for future use by the Semantic Events subsystem.

## InstallObjectAccessor

See AppleEvent Manager documentation.

```
virtual void InstallObjectAccessor
                              (DescType desiredClass,
                               DescType containerType,
                               XMPAccessorProcPtr theAccessor,
                               XMPSLong accessorRefcon);
```

`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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Content Model. Public.

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

After this method executes successfully, the function will be installed in the object accessor table. An existing function will be replaced.

**EXCEPTIONS**

See AppleEvent Manager documentation See AppleEvent Manager documentation

`kXMPErrOutOfMemory`
out of memory

## InstallSpecialHandler

See AppleEvent Manager documentation.

```
virtual void InstallSpecialHandler
                            (AEKeyword functionClass,
                             XMPSpecialHandlerPtr handler);
```

`functionClass`
The keyword for the special handler that is installed.

`handler` A pointer to the special handler.

**DESCRIPTION**

Semantic Events. Public.

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

After this method executes successfully, the function will be installed in the special handler table.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

`paramErr` handler pointer is nil or odd

`memFullErr` Not enough room in heap zone

`errAENotASpecialFunction`
Wrong keyword for a special function

## RemoveCoercionHandler

See AppleEvent Manager documentation.

```
virtual void RemoveCoercionHandler
                                (DescType fromType,
                                 DescType toType,
                                 XMPCoercionHandler handler);
```

`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.

**DESCRIPTION**

Semantic Events. Public.

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

After this method executes successfully, the function is removed from the coercion handler table.

**EXCEPTIONS**

`memFullErr` Not enough room in heap zone

`errAEHandlerNotFound`
No coercion handler found



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## RemoveEventHandler

See AppleEvent Manager documentation.

```
virtual void RemoveEventHandler
                          (AEEventClass theAEEventClass,
                           AEEventID theAEEventID,
                           XMPEventHandlerProcPtr handler);
```

`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.

### DESCRIPTION

Semantic Events. Public.

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

After this method executes successfully, the function is removed from the event handler table.

### EXCEPTIONS

`errAEHandlerNotFound`
No handler found for an Apple event



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## RemoveObjectAccessor

See AppleEvent Manager documentation.

```
virtual void RemoveObjectAccessor
                          (DescType desiredClass,
                           DescType containerType,
                           XMPAccessorProcPtr theAccessor);
```

`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.

### DESCRIPTION

Content Model. Public.

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

After this method executes successfully, the object accessor function is removed from the object accessor table.

### EXCEPTIONS

See AppleEvent Manager documentation.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## RemoveSpecialHandler

See AppleEvent Manager documentation.

```
virtual void RemoveSpecialHandler
                              (AEKeyword functionClass,
                               XMPSpecialHandlerPtr handler);
```

`functionClass`
The keyword for the special handler that is installed.

`handler` A pointer to the special handler.

### DESCRIPTION

Semantic Events. Public.

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

After this method executes successfully, the function is removed from the special handler table.

### EXCEPTIONS

`memFullErr` Not enough room in heap zone

`errAENotASpecialFunction`
Wrong keyword for a special function

## SetOSLSupportFlags

Set the flags indicating the support for the OSL that this XMPSemanticInterface provides.

```
virtual void SetOSLSupportFlags  (XMPShort flags);
```

`flags` The OSL support flags. See the documentation for AEResolve.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Semantic Events. Public.

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

See the documentation for the OSL routine, AEResolve.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPSession

*Superclasses* `XMPObject`
*Subclasses* none

This class encapsulates access to OpenDoc globals as well as initialization and shutdown of the OpenDoc environment.

## Description

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.

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: All member variables always contain valid references.

## Methods

`Close`
`Exists`
`GetArbitrator`
`GetClipboard`
`GetDispatcher`



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

```
GetDragAndDrop
GetLinkManager
GetMessageInterface
GetNameResolver
GetNameSpaceManager
GetSemanticInterface
GetShellSemtInterface
GetStorageSystem
GetTranslation
GetType
GetUndo
GetWindowState
Purge
RegisterObject
RemoveEntry
Tokenize
UniqueChangeID
UnregisterObject
```

Detailed descriptions of these methods follow, in alphabetical order.

## Close

(For embedding application or document shell only.)
Shuts down OpenDoc environment.

```
void Close();
```

**DESCRIPTION**

System. Public. Should only be called once for each OpenXMPSession call. Usually called by the application shell.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

After this method executes successfully, the OpenDoc environment will be shut down.

## Exists

The `Exists` method returns whether the object given, in the collection identified by collection and whose refCon is refCon, exists.

```
void      Exists(
XMPPersistentObject*  object,
XMPISOStr      collection,
XMPSLong      refCon);
```

| | |
|---|---|
| `object` | A reference to the object to unregister. |
| `collection` | The collection in which to find this object. |
| `refCon` | The `refCon` that was originally furnished. |
| *return value* | Whether the object was found. |

**DESCRIPTION**

None.

Public.

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

## GetArbitrator

Return reference to the global object.

```
virtual XMPArbitrator* GetArbitrator();
```

| | |
|---|---|
| *return value* | Reference to the global object |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

System. Public.

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

## GetClipboard

Return reference to the global object.

```
virtual XMPClipboard* GetClipboard();
```

*return value* Reference to the global object

**DESCRIPTION**

System. Public.

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

## GetDispatcher

Return reference to the global object.

```
virtual XMPDispatcher* GetDispatcher();
```

*return value* Reference to the global object

**DESCRIPTION**

System. Public.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetDragAndDrop

Return reference to the global object.

```
virtual XMPDragAndDrop* GetDragAndDrop();
```

*return value* Reference to the global object

**DESCRIPTION**

System. Public.

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

## GetLinkManager

Return reference to the global object.

```
XMPLinkManager* GetLinkManager();
```

**DESCRIPTION**

Data Interchange. Public. Should not be called by parts.

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

## GetMessageInterface

Return reference to the global object.

```
virtual XMPMessageInterface* GetMessageInterface();
```

*return value* Reference to the global object



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

System. Public.

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

## GetNameResolver

```
virtual XMPNameResolver* GetNameResolver();
```

*return value* Reference to the global object

**DESCRIPTION**

System. Public.

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

Return reference to the global object.

## GetNameSpaceManager

Return reference to the global object.

```
XMPNameSpaceManager* GetNameSpaceManager();
```

*return value* Reference to the global object

**DESCRIPTION**

System. Public.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetSemanticInterface

Returns cached reference to system's XMPSemanticInterface.

```
virtual XMPSemanticInterface* GetSemanticInterface();
```

*return value* Reference to system's copy of the XMPSemanticInterface.

**DESCRIPTION**

System. Public.

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

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.

## GetShellSemtInterface

(For embedding application or document shell only.)
Return a reference to an XMPSemanticInterface object that the application shell can use.

```
XMPSemanticInterface* GetShellSemtInterface();
```

*return value* a reference to the shell's XMPSemanticInterface object

**DESCRIPTION**

System. Public. Only the shell should use this method.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetStorageSystem

Return reference to the global object.

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

*return value* Reference to the global object

**DESCRIPTION**

System. Public.

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

## GetTranslation

Return reference to the global object.

```
virtual XMPTranslation* GetTranslation();
```

*return value* Reference to the global object

**DESCRIPTION**

System. Public.

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

## GetType

Get the XMPType corresponding to the given XMPTypeToken.

```
XMPBoolean GetType   (XMPTypeToken token,
                      XMPType* type);
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`token` The XMPTypeToken of interest.

*return value* Whether the type for the given token exists in the token table.

**DESCRIPTION**

Name Binding. Public.

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

If the token does not exist, `kXMPFalse` is returned. `kXMPTrue` is returned otherwise.

## GetUndo

Return reference to the global object.

```
virtual XMPUndo* GetUndo();
```

*return value* Reference to the global object

**DESCRIPTION**

System. Public.

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

## GetWindowState

Return reference to the global object.

```
virtual XMPWindowState* GetWindowState();
```

*return value* Reference to the global object



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

System. Public.

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

## Purge

Make memory available.

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

| | |
|---|---|
| `size` | The size requested to be freed. |
| *return value* | The amount of memory that was able to be freed. |

**DESCRIPTION**

System, XMPObject. Public.

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

After this method executes successfully, some memory may be freed for use.

## RegisterObject

The `RegisterObject` method registers the object given in the collection identified by collection.

```
void      RegisterObject(
XMPPersistentObject*  object,
XMPISOStr      collection,
XMPSLong      refCon);
```

| | |
|---|---|
| `object` | The object to register |
| `collection` | The identifier for the collection in which the object is to be placed. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`refCon` A value for the client to store along with the object.

**DESCRIPTION**

None.

Public.

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

The refCon may be used for extra storage. If the object is already registered, no exception is thrown and the given refCon does not replace the current refCon. After this method executes successfully, if the object is unique, it is added to the collection.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrOutOfMemory` | Not enough memory to create another entry for this object. |

## RemoveEntry

Remove an XMPType from the type/token table.

```
void RemoveEntry  (XMPType type);
```

`type` The XMPType to be removed from the type/token table.

**DESCRIPTION**

Name Binding. Public.

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

Any unique ID previously generated for this type can now be reused. No error is signalled if the type was not previously tokenized.

After this method executes successfully, the type/token pair corresponding to this type will be removed from the token table.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Tokenize

Returns a unique XMPTypeToken for a given XMPType.

```
XMPTypeToken Tokenize   (XMPType type);
```

`type` The XMPType to register
*return value* The unique token for this type.

**DESCRIPTION**

Name Binding. Public.

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

Original token is returned if type has been previously tokenized.

After this method executes successfully, a new unique token is generated if the type has not yet been tokenized.

**EXCEPTIONS**

`kXMPErrOutOfMemory`
out of memory

## UniqueChangeID

Return a change identification unique to this session and unlikely to be repeated on the network.

```
XMPChangeID UniqueChangeID();
```

*return value* A unique change identification.

**DESCRIPTION**

Data Interchange. Public.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

After this method executes successfully, the result is an identifier with a high likelihood of uniqueness.

## UnregisterObject

The `UnregisterObject` method unregisters the object given in the collection identified by collection and refCon.

```
void      UnregisterObject(
XMPPersistentObject*  object,
XMPISOStr      collection,
XMPSLong      refCon);
```

`object` The object to unregister

`collection` The identifier for the collection in which the object is to be found.

`refCon` The refCon that was originally furnished.

### DESCRIPTION

None.

Public.

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

If the object is not already registered, no exception is thrown. After this method executes successfully, the object, if it was found, is removed from the collection.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPShape

*Superclasses* `XMPObject`

*Subclasses* none

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.

## Description

Base class: XMPObject.

Class XMPShape has no friends.

More background information can be found in the XMPFrame and XMPFacet class documentation.

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: 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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Methods

XMPShape
~XMPShape
ContainsPoint
Copy
CopyFrom
CopyPolygon
GetBoundingBox
GetGeometryMode
GetPlatformShape
HasGeometry
Intersect
InverseTransform
IsEmpty
IsRectangular
IsSameAs
ReadShape
SetGeometryMode
SetPlatformShape
SetPolygon
SetRectangle
Subtract
Transform
Union
WriteShape

Detailed descriptions of these methods follow, in alphabetical order.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## XMPShape

Creates an XMPShape object with no shape data.

```
XMPShape();
```

*return value* an XMPShape object.

**DESCRIPTION**

Imaging. Public. No restrictions.

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

This must be followed by a call to SetRectangle, SetPlatformShape, CopyFrom or ReadShape (or ~XMPShape, of course.)

After this method executes successfully, the shape is allocated but uninitialized (fShape==NULL.)

## ~XMPShape

```
~XMPShape();
```

**DESCRIPTION**

Imaging. Public. No restrictions.

Derived class **can** override. Derived class **can** call base class behavior, after derived class behavior.

Deletes an XMPShape object and any private shape data.

After this method executes successfully, Shape, fShape (if any) and fShape's private data (if any) are deleted.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## ContainsPoint

Determines whether the shape contains a point.

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

`point` XMPPoint given in this shape's coordinate space.

*return value* `kXMPTrue` if aPoint is in this shape, `kXMPFalse` otherwise.

**DESCRIPTION**

Imaging. Public. No restrictions.

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

Before you call this method, Shape must be initialized.

## Copy

Returns a copy of this shape.

```
XMPShape* Copy();
```

*return value* A new shape identical to the receiver

**DESCRIPTION**

None. Public.

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

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.

Before you call this method, Shape must be initialized.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

`kXMPErrOutOfMemory`

## CopyFrom

Makes this shape an exact copy of sourceShape.

```
void CopyFrom  (XMPShape* sourceShape);
```

`sourceShape`
Pointer to another XMPShape object. Caller's storage responsibility.

**DESCRIPTION**

Imaging. Public. No restrictions.

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

This is a deep copy: the shape does not share any data with sourceShape and both can be modified independently.

Before you call this method, sourceShape must be initialized. After this method executes successfully, Shape is initialized and identical to sourceShape.

**EXCEPTIONS**

`kXMPErrOutOfMemory`
Not enough memory to copy shape data.

## CopyPolygon

Returns a new XMPPolygon describing this shape.

```
XMPPolygon* CopyPolygon();
```


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

*return value* Pointer to a new XMPPolygon.

**DESCRIPTION**

None. Public.

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

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.

Before you call this method, Shape must be initialized and must have geometry.

**EXCEPTIONS**

`kXMPErrOutOfMemory`

`kXMPErrNoGeometry`
Shape cannot be described as a polygon.

## GetBoundingBox

Returns the bounding box of this shape (the smallest rectangle that contains this shape.)

```
void GetBoundingBox  (XMPRect *bounds );
```

`bounds` Pointer to the XMPRect into which to copy the bounding box.

**DESCRIPTION**

None. Public.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

After the call the XMPRect pointed to by bounds will be set to the bounding box.

Before you call this method, Shape must be initialized. After this method executes successfully, *bounds is set to the shape's bounding box.

## GetGeometryMode

Returns the current geometry mode of this shape. See the discussion of the fMode field.

```
XMPGeometryMode GetGeometryMode();
```

*return value* Shape's geometry mode.

**DESCRIPTION**

None. Public.

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

## GetPlatformShape

Returns the data of this shape object, for a particular graphics system.

```
XMPPlatformShape GetPlatformShape   (XMPGraphicsSystem);
```

`XMPGraphicsSystem`
The graphics system of the data you want to get.

*return value* Graphics-system-specific shape data (see method description)



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Imaging. Public. No restrictions.

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

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.

Before you call this method, Shape must be initialized.

**EXCEPTIONS**

`kXMPErrInvalidGraphicsSystem`
Graphics system unknown or not installed.

## HasGeometry

Returns true if the shape's geometric information (i.e. its polygonal shape) is accessible.

```
XMPBoolean HasGeometry();
```

*return value* `kXMPTrue` if the shape can be represented as a polygon, else `kXMPFalse`.

**DESCRIPTION**

None. Public.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

A call to CopyPolygon is valid only if this is true. What makes a shape non-geometric is implementation specific. See SetGeometryMode.

Before you call this method, Shape must be initialized.

## Intersect

Intersects this shape with sectShape. This shape becomes the intersection; fShape is unchanged.

```
void Intersect (XMPShape* sectShape);
```

`sectShape` another XMPShape object to intersect the receiver with.

**DESCRIPTION**

Imaging. Public. No restrictions.

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

Before you call this method, Shape must be initialized and sectShape must be initialized. After this method executes successfully, this shape becomes the intersection of sectShape and its previous shape. sectShape is unchanged.

**EXCEPTIONS**

`kXMPErrOutOfMemory`
not enough memory to intersect shapes.

`kXMPErrNoGeometry`
Geometry mode is`kXMPNeedsGeometry`, but diffShape has no geometry.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## InverseTransform

Runs this shape through the opposite of the given transform.

```
XMPShape* InverseTransform (XMPTransform* transform);
```

`transform` XMPTransform to apply the inverse of.

*return value* The receiver.

### DESCRIPTION

None. Public.

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

This is the inverse operation of XMPShape::Transform. Shapes with no geometry may not be transformable except by simple offsets.

Before you call this method, Shape must be initialized, and the transform must be non-singular (i.e. invertible). After this method executes successfully, Shape is transformed.

### EXCEPTIONS

`kXMPErrNoShapeGeometry`
Shape does not have enough geometric information to be transformed in this way.

## IsEmpty

Returns true if the shape is empty (has no area).

```
XMPBoolean IsEmpty();
```

*return value* `kXMPTrue` if the shape is empty, otherwise `kXMPFalse`.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Imaging. Public. No restrictions

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

Before you call this method, Shape must be initialized.

## IsRectangular

Returns `kXMPTrue` if the shape is a rectangle, `kXMPFalse` otherwise.

```
XMPBoolean IsRectangular();
```

*return value* `kXMPTrue` if the shape is rectangular, else `kXMPFalse`

**DESCRIPTION**

None. Public.

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

Empty shapes are rectangular.

Before you call this method, Shape must be initialized.

## IsSameAs

Returns true if the two shapes are identical (describe the same area.)

```
XMPBoolean IsSameAs  (XMPShape* compareShape);
```

`compareShape`
shape to compare against.

*return value* `kXMPTrue` if shapes are equivalent or equal, `kXMPFalse` otherwise.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Imaging. Public. No restrictions.

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

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.

Before you call this method, Shape must be initialized.

compareShape is initialized.

**EXCEPTIONS**

`kXMPErrOutOfMemory`
not enough memory to compare shapes.

## ReadShape

Reads shape data from a storage unit into this shape.

```
XMPShape* ReadShape  (XMPStorageUnit *su );
```

| | |
|---|---|
| `su` | Prefocused storage unit to read from. |
| *return value* | The receiver if the data was read successfully; otherwise NULL. |

**DESCRIPTION**

None. Public.

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

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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

The proper usage of this call is: s = s->ReadShape(su);

Before you call this method, Storage unit must be prefocused to a property. After this method executes successfully, Shape is initialized [if data was found in su] otherwise this shape is deleted.

## SetGeometryMode

Sets the geometry mode (fMode) of this shape.

```
void SetGeometryMode (XMPGeometryMode mode );
```

`mode` The geometry mode to use.

### DESCRIPTION

None. Public.

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

See the discussion of the fMode field. Setting the mode to `kXMPNeedsGeometry` will throw a `kXMPErrNoGeometry` exception if the shape has no geometry.

### EXCEPTIONS

`kXMPErrNoGeometry`
If mode is set to `kXMPNeedsGeometry` but shape has no geometry.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## SetPlatformShape

Sets the shape's graphics system dependent data.

```
void SetPlatformShape   (XMPGraphicsSystem graphicsSystem,
                         XMPPlatformShape platformShape);
```

`graphicsSystem`
On Mac, `kXMPQuickDraw` or `kXMPQuickDrawGX`.

`platformShape`
On Mac, this is a RgnHandle or gxShape.

**DESCRIPTION**

Imaging. Public. No restrictions.

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

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.

Before you call this method, graphicsSystem must be a supported graphics system of the platform. platformShape must be a valid platform-specific shape for this graphics system. After this method executes successfully, Shape is initialized.

**EXCEPTIONS**

`kXMPErrInvalidGraphicsSystem`
not a valid graphics system.

## SetPolygon

Sets the shape based on the given polygon.

```
XMPShape* SetPolygon (XMPPolygon *polygon );
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `polygon` | XMPPolygon to use for this shape. |
| *return value* | The receiver. |

**DESCRIPTION**

None. Public.

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

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.

Before you call this method, polygon must point to a valid XMPPolygon. After this method executes successfully, Shape is initialized to a copy of the given polygon.

## SetRectangle

Changes this shape to be a rectangle with the given coordinates.

```
XMPShape* SetRectangle  (XMPRect *rect );
```

| | |
|---|---|
| `rect` | Rectangle to set this shape equal to. |

**DESCRIPTION**

None. Public.

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

This call may be used to initialize a shape right after it's created.

Before you call this method, rect must point to a valid XMPRect structure. After this method executes successfully, Shape is initialized to a rectangle identical to rect.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Subtract

Subtracts diffShape from this shape; result is stored in this shape.

```
XMPShape* Subtract   (XMPShape* diffShape);
```

`diffShape` Shape to subtract from this shape.
*return value* The receiver.

### DESCRIPTION

Imaging. Public. No restrictions.

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

Before you call this method, Shape must be initialized. diffShape must be initialized. After this method executes successfully, this shape is set to the value (this - diffShape). diffShape is unaltered.

### EXCEPTIONS

`kXMPErrOutOfMemory`
not enough memory to subtract shapes.

`kXMPErrNoShapeGeometry`
Geometry mode is`kXMPNeedsGeometry`, but diffShape has no geometry.

## Transform

Runs this shape through a transformation (an offset, scale, rotation, skew, etc.)

```
XMPShape* Transform  (XMPTransform* transform);
```

`transform` XMPTransform to apply to this shape.
*return value* The receiver.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

None. Public.

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

Shapes with no geometry may not be transformable except by simple offsets.

Before you call this method, the shape must be initialized. After this method executes successfully, Shape is transformed.

**EXCEPTIONS**

`kXMPErrNoGeometry`
Shape doesn't have enough geometric information for this transformation.

## Union

Unions this shape with sectShape. This shape becomes the union; unionShape is unchanged.

```
XMPShape* Union   (XMPShape* unionShape);
```

`unionShape` Shape to combine with this shape.

*return value* The receiver.

**DESCRIPTION**

None. Public.

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

Before you call this method, Shape must be initialized. unionShape must be initialized. After this method executes successfully, this shape becomes the intersection of sectShape and its previous shape. sectShape is unchanged.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

`kXMPErrNoGeometry`
Geometry mode is `kXMPNeedsGeometry`, but diffShape has no geometry.

## WriteShape

Writes shape data to a specific value of a storage unit.

```
void WriteShape   (XMPStorageUnit *su );
```

`su` Storage unit (pre-focused) to write polygon data to.

**DESCRIPTION**

None. Public.

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

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.)

Before you call this method, Shape must be initialized. After this method executes successfully, shape data is written to one or more values of a property of the storage unit.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPStorageSystem

*Superclasses* `XMPObject`

*Subclasses* none

The XMPStorageSystem object provides the functionality of the OpenDoc Storage System. It is responsible for guaranteeing that there is only one XMPContainer object associated with each physical Container.

## Description

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.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

only oneXMPStorageUnit objectassociated with each StorageUnit, Part, Frame, LinkSource and Link within it.

Invariants: 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.

## Methods

`CreateContainer`

`GetContainer`

`GetSession`

`NeedSpace`

`ReleaseContain` er

Detailed descriptions of these methods follow, in alphabetical order.

### CreateContainer

(For embedding application or document shell only.)
This is used to create a new container that corresponds to the given `containerType` and `id`.

```
XMPContainer* CreateContainer
                              (XMPContainerType containerType,
                               XMPContainerID id);
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`containerType`
A general type of container.

`id` A pointer to a container brand specific structure in memory.

*return value* A fully functional `XMPContainer` object that refers to the container of the given `containType` and `id`.

**DESCRIPTION**

Container Manipulation. Public. This method can be called byShell or Container App to create a new OpenDoc container.

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

See `GetContainer()` for more information on id and`containerType`.

**EXCEPTIONS**

`kXMPInvalidContainerID`
Invalid Container ID.

`kXMPCannotCreateContainer`
Invalid Container Type.

## GetContainer

(For embedding application or document shell only.)
This is used to return an`XMPContainer` object tha refers to the container of the specified `containerType` and with the specified `id`.

```
XMPContainer* GetContainer (XMPContainerType containerType,
                            XMPContainerID id);
```

`containerType`
Type of the desired container.

`id` The ID of the desired container.

*return value* A fully functional `XMPContainer` object that refers to the container of the given `containType` and `id`.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Object Storage. Public. This method can be called by Shell or Container App when it needs to open an OpenDoc document.

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

`XMPContainerID` is a 32-bit value and is used only by the `XMPContainer` object. Therefore, its format is completely opaque to the Storage System. Typically, it is a pointer to a container brand specific structure in memory (e.g., `FileSpec`).

**EXCEPTIONS**

`kXMPInvalidContainerID`
Invalid Container ID.

`kXMPCannotGetContainer`
Invalid Container Type.

## GetSession

Returns the current XMPSession object.

```
XMPSession*  GetSession();
```

*return value* XMPBaseSession in which this XMPStorageSystem runs.

**DESCRIPTION**

ObjectStorage. Public. Parts can call this method even though they should really be calling XMPStorageUnit::GetSession().

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## NeedSpace

This is called to check for or preflight some memory.

```
void NeedSpace (XMPSize memSize,
                XMPBoolean doPurge);
```

`memSize` size of memory block desired

`doPurge` whether or not to purge if necessary

**DESCRIPTION**

Memory. Public. This method can be called by any class. However, it should be used cautiously as this may be a slow operation.

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

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.

## ReleaseContainer

(For embedding application or document shell only.)
This is used to release the specified `XMPContainer` object.

```
XMPStorageSystem* ReleaseContainer
                                    (XMPContainer* container);
```

`container` The container to be released.

*return value* The container is no longer a valid `XMPContainer` object.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Memory. Public. Private between `XMPStorageSystem` and `XMPContainer`. This method should only be called by `XMPContainer::Release`.

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

Before calling this method, you must ensure that `container` is a valid `XMPContainer` object that was obtained via either `GetContainer` or `CreateContainer`.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPStorageUnit

*Superclasses* `XMPRefCntObject` → `XMPObject`

*Subclasses* none

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.

## Description

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.

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 position codes.

When focussed to a particular value, the StorageUnit provides a stream interface to that value.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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: 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.

## Methods

`XMPStorageUnit`

`~XMPStorageUnit`

`AddProperty`

`AddValue`

`ClearAllPromises`

`CloneInto`

`CloneTo`

`CopyTo`

`CountProperties`

`CountValues`



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

```
CreateCursor
CreateView
DeleteValue
Exists
Exists
Externalize
Focus
Focus
GetDraft
GetGenerationNumber
GetID
GetIDFromStorageUnitRef
GetName
GetOffset
GetPromiseValue
GetProperty
GetSession
GetSize
GetStorageUnitRefIterator
GetStrongStorageUnitRef
GetType
GetValue
GetWeakStorageUnitRef
IncrementGenerationNumber
IncrementRefCount
InitStorageUnit
InsertValue
Internalize
IsPromiseValue
IsStrongStorageUnitRef
IsWeakStorageUnitRef
Lock
```


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

```
Purge
Release
Remove
RemoveStorageUnitRef
SetName
SetOffset
SetPromiseValue
SetType
SetValue
```

Detailed descriptions of these methods follow, in alphabetical order.

## XMPStorageUnit

(For embedding application or document shell only.)
This method is the constructor of the class.

```
XMPStorageUnit();
```

*return value* this

**DESCRIPTION**

Constructor. Public. Private within Container Suite. This method should only be called by XMPDraft::CreateStorageUnit or XMPDraft::GetStorageUnit.

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

After this method executes successfully, the return result is an uninitialized XMPStorageUnit object.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## ~XMPStorageUnit

(For embedding application or document shell only.)
This method is the destructor of the class.

```
~XMPStorageUnit();
```

**DESCRIPTION**

Cleanup. Public. Private within Container Suite. This method should only be called by XMPDraft.

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

Note that it does not externalize the XMPStorageUnit object.

After this method executes successfully, this is no longer a valid XMPStorageUnit object.

## AddProperty

Adds a property to a storage unit.

```
XMPStorageUnit* AddProperty (XMPPropertyName propertyName);
```

`propertyName`
Name of the property to add

*return value* this

**DESCRIPTION**

ObjectStorage. Public.

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

If no property with the same name exist in this storage unit, it is added.

The focus is set to the given property.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Before you call this method, a Property with the given name may or may not exist in the Storage Unit. After this method executes successfully, XMPStorageUnit is focused to the Property of the given name.

**EXCEPTIONS**

`kXMPErrInvalidPropertyName`
propertyName is `kXMPNull`

`kXMPErrCannotAddProperty`
Failure to add the given Property.

## AddValue

Adds a value type to a storage unit.

```
XMPStorageUnit* AddValue   (XMPValueType type);
```

`type` type of value to add
*return value* this

**DESCRIPTION**

ObjectStorage. Public.

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

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.

Before you call this method, this XMPStorageUnit must be focused to a Property. The given type may or may not exist in the focused Property. After this method executes successfully, XMPStorageUnit is focused to the Value with the given type.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

`kXMPErrValueExists`
Value exists already.

`kXMPErrUnfocusedStorageUnit`
This XMPStorageUnit is not focused to a Property.

`kXMPErrInvalidType`
type is `kXMPNull`.

`kXMPErrCannotAddType`
Cannot add Type to Storage Unit.

## ClearAllPromises

Removes all Values containing promises from this Storage Unit.

```
void    ClearAllPromises();
```

**DESCRIPTION**

Cleanup. Public.

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

## CloneInto

Deep-copy all Properties and Values of this Storage Unit to the specified destStorageUnit.

```
void  CloneInto   (XMPDraftKey key,
                   XMPStorageUnit* destStorageUnit,
                   XMPStorageUnit* initiatingFrameSU);
```

`key` XMPDraftKey identifying the Clone transaction.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`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.

**DESCRIPTION**

ObjectStorage. Public.

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

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.

**EXCEPTIONS**

`kXMPErrCloneNotStarted`
: A Clone was started unsuccessfully.

`kXMPErrInvalidDraftKey`
: Invalid Clone.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## CloneTo

Deep-copy all Properties and Values of this Storage Unit to the newly created Storage Unit in the specified destDraft.

```
XMPStorageUnit* CloneTo
                    (XMPDraftKey key,
                     XMPDraft* destDraft,
                     XMPStorageUnit* initiatingFrameSU);
```

`initiatingFrameSU`
All Storage Units cloned from this point on should be within the scope of this XMPFrame.

`destDraft` Destination Draft to where this Storage Unit is copied.

`initiatingFrameSU`
All Storage Units cloned from this point on should be within the scope of this XMPFrame.

*return value* XMPStorageUnit referring to the new Storage Unit created in the destination draft.

**DESCRIPTION**

ObjectStorage. Public.

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

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.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

`kXMPErrCloneNotStarted`
A Clone was started unsuccessfully.

`kXMPErrInvalidDraftKey`
Invalid Clone.

## CopyTo

Shallow Copy (i.e., one-level) properties and values of this storageUnit to the given toSU.

```
void  CopyTo (XMPStorageUnit* toSU);
```

`toSU` the storage unit from which data is to be copied

**DESCRIPTION**

Storage Unit Manipulation. Public.

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

After this method executes successfully, all properties and values in this XMPStorageUnit have their respective values in toSU.

## CountProperties

Returns the number of Properties in this XMPStorageUnit.

```
XMPULong CountProperties();
```

*return value* Number or Properties in this XMPStorageUnit.

**DESCRIPTION**

Storage Unit Manipulation. Public.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

## CountValues

Returns the number of Values in this focused XMPStorageUnit.

```
XMPULong CountValues();
```

*return value* Number or Values in this focused XMPStorageUnit.

**DESCRIPTION**

Storage Unit Manipulation. Public.

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

Before you call this method, this XMPStorageUnit must be focused to a Property or a Value.

**EXCEPTIONS**

`kXMPErrUnfocusedStorageUnit`
This XMPStorageUnit is not focused to a Property.

## CreateCursor

Creates a XMPCursor object from the current focus of this XMPStorageUnit and returns it.

```
XMPStorageUnitCursor* CreateCursor();
```

*return value* a newly created storageUnitCursor


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Initialization. Public.

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

**EXCEPTIONS**

`kXMPErrUnfocusedStorageUnit`
This XMPStorageUnit is not focused to a Property or a Value.

After this method executes successfully, the returned XMPStorageUnitCursor object contains the same context as this XMPStorageUnit object.

## CreateView

Creates a new XMPStorageUnitView object.

```
XMPStorageUnitView* CreateView
                              (XMPStorageUnitCursor* cursor);
```

`cursor` Cursor to create the current view. If cursor is `kXMPNull`, the current focus is used.

*return value* a StorageUnitView of this unit and the given cursor or the focus of this unit

**DESCRIPTION**

Initialization. Public.

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

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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Before you call this method, either cursor is a valid XMPStorageUnitCursor or this XMPStorageUnit is focused to a Property or a Value. After this method executes successfully, the returned XMPStorageUnitView represents the appropriate context of this XMPStorageUnit.

## DeleteValue

Deletes length bytes at the current offset in the current value focus.

```
void  DeleteValue (XMPULong length);
```

`length` the length of bytes to delete from the focussed value

**DESCRIPTION**

ObjectStorage. Public.

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

If the focused Value is a Promise Value, the Promise is resolved first before insertion is done.

Before you call this method, this XMPStorageUnit object must be focused to a particular value. After this method executes successfully, length bytes at the current offset in the current value focus have been deleted.

**EXCEPTIONS**

`kXMPErrUnfocusedStorageUnit`
This XMPStorageUnit is not focused to a Value.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Exists

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.

```
XMPBoolean Exists (XMPPropertyName propertyName,
                   XMPValueType valueType,
                   XMPValueIndex valueIndex);
```

`propertyName`
Name of a Property.

`valueType` Type of a Value.

`valueIndex` Value Index.

*return value* XMPBoolean showing whether the specified Property and Value exists.

### DESCRIPTION

Object Storage. Public.

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

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`.

Before you call this method, Property Name must be specified. After this method executes successfully, the focus of the XMPStorageUnit is not changed.

### EXCEPTIONS

`kXMPErrInvalidName`
Invalid Property Name.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`kXMPErrInvalidType`
Invalid Value Type.

## Exists

Returns `kXMPTrue` if the given context specified by the cursor exists.

```
XMPBoolean Exists (XMPStorageUnitCursor* cursor);
```

`cursor` XMPStorageUnitCursor specifying a context (Property and/or a Value.)

*return value* XMPBoolean showing whether the specified Property and Value exists.

**DESCRIPTION**

Object Storage. Public.

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

Returns `kXMPTrue` if the given context specified by the cursor exists. Otherwise, returns `kXMPFalse`.

After this method executes successfully, the focus of the XMPStorageUnit is not changed.

## Externalize

Makes all changes in the Storage Unit persistent.

```
XMPStorageUnit* Externalize();
```

*return value* this


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

ObjectStorage. Public.

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

Resolves all promises in this XMPStorageUnit and makes all changes in the Storage Unit persistent.

## Focus

Focuses the StorageUnit on the specified property and value.

```
XMPStorageUnit* Focus   (XMPPropertyName propertyName,
                         XMPPositionCode propertyPosCode,
                         XMPValueType valueType,
                         XMPValueIndex valueIndex,
                         XMPPositionCode valuePosCode);
```

`propertyName`
: focus property name

`propertyPosCode`
: focus relative property relation

`valueType` focus value type

`valueIndex` focus value index

`valuePosCode`
: focus relative value relation

*return value* this

**DESCRIPTION**

Storage Unit Manipulation. Public.

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

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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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.

After this method executes successfully, the StorageUnit is focused on the specified property and value.

The current offset is 0.

**EXCEPTIONS**

`kXMPErrInvalidProperty`
No such Property.

`kXMPErrInvalidType`
No Value with this Type.

`kXMPErrInvalidValueIndex`
Invalid value index.

`kXMPErrInvalidPosCode`
Invalid position code.

## Focus

Focus the Storage Unit using the property and value specified by the given XMPStorageUnitCursor object.

```
XMPStorageUnit* Focus (XMPStorageUnitCursor* cursor);
```

`cursor` the cursor to use for focusing

*return value* this

**DESCRIPTION**

Storage Unit Manipulation. Public.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

After this method executes successfully, the StorageUnit is focused on the property and/or value specified by the XMPStorageUnitCursor object.

The current offset is 0.

**EXCEPTIONS**

`kXMPErrInvalidPropertyName`
No such Property.

`kXMPErrInvalidType`
No Value with such Type.

`kXMPErrInvalidValueIndex`
Invalid Value Index.

`kXMPErrInvalidPosCode`
Invalid position code.

## GetDraft

Returns the draft from which this XMPStorageUnit is instantiated.

```
XMPDraft* GetDraft();
```

*return value* XMPDraft from which this XMPStorageUnit object is created.

**DESCRIPTION**

Getter. Public.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetGenerationNumber

Return the generation number of the current value focus.

```
XMPULong  GetGenerationNumber();
```

*return value* the generation number of the current value

**DESCRIPTION**

Storage Unit Manipulation. Public.

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

Before you call this method, this XMPStorageUnit object must be focused to a value.

**EXCEPTIONS**

`kXMPUnfocusedStorageUnit`
This XMPStorageUnit object is not focused to a Value.

## GetID

Returns the ID of this Storage Unit.

```
XMPID GetID();
```

*return value* the id of the focus

**DESCRIPTION**

ObjectStorage. Public.

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


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetIDFromStorageUnitRef

Resolves the given ref to a XMPStorageUnitID and returns the XMPStorageUnitID.

```
XMPStorageUnitID GetIDFromStorageUnitRef(
XMPStorageUnitRef ref);
```

`ref` XMPStorageUnitRef to be resolved.

*return value* the StorageUnitID associated with the given ref in this value

### DESCRIPTION

Persistent Reference. Public.

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

If the specified ref cannot be resolved, an exception is raised.

Before you call this method, this XMPStorageUnit object must be focused to a value.

### EXCEPTIONS

`kXMPUnfocusedStorageUnit`
this XMPStorageUnit object is not focused to a Value.

`kXMPErrInvalidStorageUnitRef`
Invalid Storage Unit Reference.

## GetName

Returns a copy of the name of the Storage Unit.

```
XMPName GetName();
```

*return value* Name of the Storage Unit



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Getter. Public.

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

If the Storage Unit does not have a name, `kXMPNull` is returned.

## GetOffset

Returns the current offset of the currently focused value.

```
XMPULong GetOffset();
```

*return value* the current offset of the currently focused value

**DESCRIPTION**

Getter. Public.

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

Before you call this method, this XMPStorageUnit object must be focused to a value.

**EXCEPTIONS**

`kXMPErrUnfocusedStorageUnit`
This XMPStorageUnit is not focused to a Value.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetPromiseValue

Extracts data placed in the value when the promise was created. This does not fulfill the promise.

```
XMPULong GetPromiseValue   (XMPValueType valueType,
                            XMPULong offset,
                            XMPULong length,
                            XMPValue value,
                            XMPPart **sourcePart);
```

| | |
|---|---|
| `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. **out** Source Part which has put out the Promise. |
| *return value* | the actual number of bytes read |

### DESCRIPTION

Promise Manipulation. Public.

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

Before you call this method, the current storage unit must be focused to a valid property containing a promise of the same type specified in the valueType parameter.

After this method executes successfully, the promise data (not the data the promise represents) is copied into the supplied value parameter.

### EXCEPTIONS

`kXMPErrUnfocusedStorageUnit`
This XMPStorageUnit object is not focused to a Property.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`kXMPErrInvalidType`
valueType is `kXMPNull`.

`kXMPErrCannotAddType`
Invalid valueType.

`kXMPErrInvalidValue`
Cannot focus to specified Value with valueType.

`kXMPErrNotPromise`
The Value does not contain a Promise.

## GetProperty

Returns the name of the currently focused property.

```
XMPPropertyName GetProperty();
```

*return value* Property name of the current focus.

### DESCRIPTION

Getter. Public.

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

Before you call this method, this XMPStorageUnit object must be focused to a property or value.

### EXCEPTIONS

`kXMPErrUnfocusedStorageUnit`
this XMPStorageUnit is not focused to a Property.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetSession

Returns the session in which this XMPStorageUnit object is running.

```
XMPSession* GetSession();
```

*return value* XMPSession object in which this XMPStorageUnit object is running.

**DESCRIPTION**

Getter. Public.

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

This is a utility routine Parts should use to retrieve the current XMPSession object.

## GetSize

Returns the size of the current focus.

```
XMPULong GetSize();
```

*return value* Desired size.

**DESCRIPTION**

Value Manipulation. Public.

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

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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

`kXMPUnfocusedStorageUnit`

## GetStorageUnitRefIterator

Returns an iterator for iterating over all the persistent references created in the currently focused Value.

```
XMPStorageUnitRefIterator* GetStorageUnitRefIterator();
```

*return value* Iterator used to iterate over all persistent references in the focused Value.

**DESCRIPTION**

Iteration. Public.

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

Before you call this method, this XMPStorageUnit must be focused to a Value.

## GetStrongStorageUnitRef

Returns a persistent reference to the Storage Unit refered to by embeddedSU.

```
XMPStorageUnitRef GetStrongStorageUnitRef
                                (XMPStorageUnit* embeddedSU);
```

`embeddedSU` XMPStorageUnit object referring to the Storage Unit whose ref is desired.

*return value* persistent reference to the Storage Unit referred to by embeddedSU.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Persistent Reference. Public.

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

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.

Before you call this method, this XMPStorageUnit object must be focused to a value.

Storage Units referred to by this XMPStorageUnit and embeddedSU must be in the same Draft.

**EXCEPTIONS**

`kXMPUnfocusedStorageUnit`

## GetType

Returns the type of the currently focused value.

```
XMPValueType GetType();
```

*return value* Type of the focused particular value

**DESCRIPTION**

Getter. Public.

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

Before you call this method, this XMPStorageUnit object must be focused to a value.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

`kXMPErrInvalidValue`

`kXMPErrUnfocusedStorageUnit`.

## GetValue

Reads length bytes from the currently focused value at the current offset into the buffer value.

```
XMPULong GetValue (XMPULong length,
                   XMPValue value);
```

| | |
|---|---|
| `length` | Number of bytes to read into value |
| `value` | a buffer of length>= length |
| *return value* | Number of bytes actually read |

**DESCRIPTION**

Value Manipulation. Public.

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

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.

Before you call this method, this XMPStorageUnit object must be focused to a value. value is a buffer of length length or more. After this method executes successfully, value contains length bytes from offset in the currently focused value.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetWeakStorageUnitRef

Returns a persistent reference to the Storage Unit refered to by embeddedSU.

```
XMPStorageUnitRef GetWeakStorageUnitRef
                                    (XMPStorageUnit* embeddedSU);
```

`embeddedSU` Storage Unit for which a ref is desired

*return value* a ref to the given storageUnit

**DESCRIPTION**

Persistent Reference. Public.

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

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.

Before you call this method, this XMPStorageUnit object must be focused to a value.

**EXCEPTIONS**

`kXMPUnfocusedStorageUnit`
this XMPStorageUnit is not focused to a Value.

## IncrementGenerationNumber

Increment and return the generation number of the current value focus.

```
XMPULong  IncrementGenerationNumber();
```

*return value* Generation number of the focused Value.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Generation. Public.

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

Before you call this method, this XMPStorageUnit object must be focused to a value.

**EXCEPTIONS**

`kXMPErrUnfocusedStorageUnit`
this XMPStorageUnit is not focused to a Value.

## IncrementRefCount

Increments the reference count of this object.

```
void IncrementRefCount();
```

**DESCRIPTION**

Reference counting. Public.

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

After this method executes successfully, the ref count of this XMPStorageUnit object is incremented by 1.

## InitStorageUnit

(For embedding application or document shell only.)
Initializes this XMPStorageUnit object.

```
void InitStorageUnit (XMPDraft* draft,
                      XMPStorageUnitID suid);
```


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `draft` | the draft of this storage unit |
| `suid` | the id of this storage unit |

**DESCRIPTION**

Constructor. Public. Private within Storage System. This method should only be called by XMPDraft::CreateStorageUnit.

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

Before you call this method, this must be a valid XMPStorageUnit. After this method executes successfully, this is an initialized XMPStorageUnit object.

this XMPStorageUnit is focused to all Properties.

## InsertValue

Insert length bytes into the currently focused value at the current offset from the buffer value.

```
void InsertValue  (XMPULong length,
                   XMPValue value);
```

| | |
|---|---|
| `length` | Number of bytes to write into the focussed value |
| `value` | Buffer of size>= length |

**DESCRIPTION**

Storage Unit Manipulation. Public.

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

If the focused Value is a Promise Value, the Promise is resolved first before insertion is done.

Before you call this method, this XMPStorageUnit object must be focused to a value. value is a buffer of size length or more. After this method executes successfully, length bytes of value have been inserted at the current offset in the currently focused value.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

`kXMPErrUnfocusedStorageUnit`
this XMPStorageUnit is not focused to a Value.

## Internalize

(For embedding application or document shell only.)
Brings into ephemeral storage from persistent storage, all properties and their values in the current focus.

```
XMPStorageUnit* Internalize();
```

*return value* this

**DESCRIPTION**

ObjectStorage. Public. Private within Storage System. This method should only be called by XMPDraft.

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

After this method executes successfully, all properties and values in the current focus have been brought into ephemeral storage from persistent storage.

## IsPromiseValue

Returns `kXMPTrue` if focused Value contains a promise.

```
XMPBoolean IsPromiseValue();
```

*return value* `kXMPTrue` if the focused value contains a promise. otherwise false.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Promise. Public.

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

Returns `kXMPTrue` if focused Value contains a promise. Otherwise, returns `kXMPFalse`.

Before you call this method, this XMPStorageUnit must be focused to a Value.

## IsStrongStorageUnitRef

Returns `kXMPTrue` if ref is a strong Storage Unit Ref.

```
XMPBoolean IsStrongStorageUnitRef(
XMPStorageUnitRef ref);
```

| | |
|---|---|
| `ref` | XMPStorageUnitRef to be tested. |
| *return value* | Boolean showing whether the ref is a Strong Storage Unit Ref. |

**DESCRIPTION**

Persistent Reference. Public.

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

Returns `kXMPTrue` if ref is a strong Storage Unit Ref. Otherwise, it returns `kXMPFalse`.

Before you call this method, this XMPStorageUnit must be focused to a Value.

## IsWeakStorageUnitRef

Returns `kXMPTrue` if ref is a weak Storage Unit Ref.

```
XMPBoolean IsWeakStorageUnitRef (XMPStorageUnitRef ref);
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `ref` | XMPStorageUnitRef to be tested. |
| *return value* | Boolean showing whether the ref is a Strong Storage Unit Ref. |

**DESCRIPTION**

Persistent Reference. Public.

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

Returns `kXMPTrue` if ref is a weak Storage Unit Ref. Otherwise, it returns `kXMPFalse`.

Before you call this method, this XMPStorageUnit must be focused to a Value.

## Lock

Sets the status of this XMPStorageUnit to locked.

```
XMPStorageUnitKey Lock (XMPStorageUnitKey key);
```

| | |
|---|---|
| `key` | Previous XMPStorageUnitKey acquired. |
| *return value* | XMPStorageUnitKey |

**DESCRIPTION**

Shared access. Public.

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

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.

After this method executes successfully, either an exception is raised or the lock is granted.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

`kXMPErrStorageUnitLocked`
XMPStorageUnit is already locked.

`kXMPErrInvalidStorageUnitKey`
Invalid XMPStorageUnitKey.

## Purge

(For embedding application or document shell only.)
Purge memory from ephemeral store until size bytes (not necessarily contiguous) have been freed up.

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

`size` Number of bytes to purge

*return value* Number of bytes actually purged

**DESCRIPTION**

Memory Management. Public. Private within Storage System. This method should only be called by XMPDraft::Purge.

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

After this method executes successfully, either size bytes (not necessarily contiguous) are free in the default heap or all Properties and Values in memory have been flushed out.

## Release

Releases this XMPStorageUnit object.

```
XMPDraft* Release();
```

*return value* fDraft



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Reference counting. Public.

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

This is called when the reference to this XMPStorageUnit is no longer needed.

After this method executes successfully, this is no longer a valid XMPStorageUnit object.

## Remove

Removes a Value if focused to a Value. Otherwise, removes a property if focused to a property. Otherwise, removes all the properties.

```
XMPStorageUnit* Remove();
```

*return value* this

**DESCRIPTION**

Storage Unit Manipulation; Value Manipulation. Public.

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

After this method executes successfully, the current focus is undefined.

## RemoveStorageUnitRef

Removes a XMPStorageUnitRef from a Value.

```
XMPStorageUnit* RemoveStorageUnitRef
                                    (XMPStorageUnitRef ref);
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Persistent Reference. Public.

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

Before you call this method, this XMPStorageUnit object must be focused to a value. ref is a valid XMPStorageUnitRef (either strong or weak). After this method executes successfully, ref is no longer a valid XMPStorageUnitRef in that Value.

**EXCEPTIONS**

`kXMPErrUnfocusedStorageUnit`
this XMPStorageUnit is not focused to a Value.

## SetName

Set the name of the Storage Unit.

```
void SetName (XMPName name);
```

`name` the name of the focus

**DESCRIPTION**

Setter. Public.

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

After this method executes successfully, the name of the Storage Unit is name.

## SetOffset

Sets the current offset of the currently focused value to offset.

```
void SetOffset (XMPULong offset);
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`offset` the offset to use with the currently focused value

**DESCRIPTION**

Setter. Public.

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

Before you call this method, this XMPStorageUnit must be focused to a Value.

0≤offset<size of the focused Value.

After this method executes successfully, the current offset for value operations on that value is offset.

**EXCEPTIONS**

`kXMPErrUnfocusedStorageUnit`
this XMPStorageUnit is not focused to a Value.

## SetPromiseValue

Puts the given data into the focused value and make the focused value a promise.

```
void SetPromiseValue (XMPValueType valueType,
                      XMPULong offset,
                      XMPULong length,
                      XMPValue value
                     XMPPart *sourcePart);
```

`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.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Promise. Public.

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

Before you call this method, value must be of length bytes long or less. this XMPStorageUnit must be focused to a Property. After this method executes successfully, 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.

**EXCEPTIONS**

`kXMPErrUnfocusedStorageUnit`
this XMPStorageUnit is not focused to a Property.

`kXMPErrInvalidType`
valueType is invalid ( `kXMPNull`).

`kXMPErrCannotAddType`
Invalid Type.

## SetType

Sets the type of the currently focused value.

```
void SetType (XMPValueType valueType);
```

`valueType` New type of the focused particular value

**DESCRIPTION**

Setter. Public.

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

Before you call this method, this XMPStorageUnit object must be focused to a Value. After this method executes successfully, The type of the focused value is valueType.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

`kXMPErrUnfocusedStorageUnit`
this XMPStorageUnit is not focused to a Value.

`kXMPErrInvalidType`
valueType is invalid (i.e., `kXMPNull`).

## SetValue

Writes length bytes to the currently focused value at the current offset from the buffer value.

```
void SetValue  (XMPULong length,
                XMPValue value);
```

`length` Number of bytes to write into the focussed value

`value` Buffer of size>= length

**DESCRIPTION**

Value Manipulation. Public.

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

If length + offset is greater than the current size of the Value, the Value should be grown to accommodate the extra bytes.

Before you call this method, this XMPStorageUnit object must be focused to a Value. value is a buffer of size length or more. After this method executes successfully, length bytes of value have been written into offset in the currently focused value.

**EXCEPTIONS**

`kXMPErrUnfocusedStorageUnit`
this XMPStorageUnit is not focused to a Value.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPStorageUnitCursor

*Superclasses* none

*Subclasses* none

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.

## Description

XMPStorageUnitCursor has no base class.

Part developers or Platform developers can subclass this class.

Invariants: The focus context (i.e., Property Name and Value Type OR Property Name and Value Index) cannot be changed once the object is initialized.

## Methods

`XMPStorageUnitCursor`

`~XMPStorageUnitCursor`

`GetCursor`

`InitStorageUnitCursor`

Detailed descriptions of these methods follow, in alphabetical order.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## XMPStorageUnitCursor

This is the constructor of the class.

```
XMPStorageUnitCursor();
```

**DESCRIPTION**

Constructor. Public.

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

## ~XMPStorageUnitCursor

This is the destructor for the class.

```
~XMPStorageUnitCursor();
```

**DESCRIPTION**

Destructor. Public.

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

## GetCursor

Returns the focus context of this cursor.

```
void GetCursor (XMPPropertyName* propertyName,
                XMPValueType* valueType,
                XMPValueIndex* valueIndex);
```


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`propertyName`
Property Name of the focus context. On return, Property Name for the focus context.

`valueType` Value Type for the focus context. On return, ValueType for the focus context.

`valueIndex` Pointer to buffer for Value index for the focus context. On return, Value index for the focus context.

**DESCRIPTION**

Getter. Public.

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

**EXCEPTIONS**

`valueIndex` Value index for the focus context.

## InitStorageUnitCursor

Initializes this object with the focus context.

```
void InitStorageUnitCursor (XMPPropertyName propertyName,
                            XMPValueType valueType,
                            XMPValueIndex valueIndex);
```

`propertyName`
Property Name for the focus context.

`valueType` Value Type for the focus context.

`valueIndex` Value index for the focus context.

**DESCRIPTION**

Constructor. Public.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

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.

After this method executes successfully, this object is a fully functional XMPStorageUnitCursor (i.e., it can be used for XMPStorageUnit::Focus).

**EXCEPTIONS**

`kXMPErrInvalidProperty`
Property Name is `kXMPNull`.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPStorageUnitRefIterator

*Superclasses* none

*Subclasses* none

XMPStorageUnitRefIterator is a companion of XMPStorageUnit , and is used to iterate over the references in a Value.

## Description

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.

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: 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.

## Methods

`XMPStorageUnitRefIterator`

`~XMPStorageUnitRefIterator`



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

```
First
InitStorageUnitRefIterator
IsNotComplete
Next
```

Detailed descriptions of these methods follow, in alphabetical order.

## XMPStorageUnitRefIterator

Constructs the iterator.

```
XMPStorageUnitRefIterator();
```

**DESCRIPTION**

Constructor. Public.

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

After this method executes successfully, the result is a properly constructed iterator.

## ~XMPStorageUnitRefIterator

This is the destructor of the class.

```
~XMPStorageUnitRefIterator();
```

**DESCRIPTION**

Destructor. Public.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

Before you call this method, this must be a valid XMPStorageUnitRefIterator object. After this method executes successfully, this is no longer a valid XMPStorageUnitRefIterator object.

## First

Returns the first reference in the Value if such a reference exists.

```
XMPStorageUnitRef* First();
```

*return value* The first XMPStorageUnitRef in the focused Value of the XMPStorageUnit.

**DESCRIPTION**

Iteration. Public.

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

If no such reference exists, the return value is undefined.

## InitStorageUnitRefIterator

Initializes this XMPStorageUnitRefIterator object.

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

`storageUnit***`

**DESCRIPTION**

Constructor. Public.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

## IsNotComplete

Returns `kXMPTrue` if the iteration is not complete, `kXMPFalse` otherwise.

```
XMPBoolean IsNotComplete();
```

*return value* Boolean showing whether the iteration is incomplete.

**DESCRIPTION**

Iteration. Public.

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

Before you call this method, a constructed object of this class must exist.

## Next

Returns the next reference in the Value if such a reference exists.

```
XMPStorageUnitRef* Next();
```

*return value* The next reference in the Value.

**DESCRIPTION**

Iteration. Public.

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

If no such reference exists, the return value is undefined.

Before you call this method, First() must have been called.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPStorageUnitView

*Superclasses* `XMPObject`
*Subclasses* none

A XMPStorageUnitView provides an easy and thread-safe way to access a specific Property or a Value of a Storage Unit.

## Description

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 overridden.

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 directly by clients.

Parts can instantiate XMPStorageUnitView.

Invariants: The XMPStorageUnit and XMPStorageUnitCursor of a StorageUnitView are specified in the InitStorageUnitView method, and these values never change.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Methods

XMPStorageUnitView
~XMPStorageUnitView
AddProperty
AddValue
CloneInto
CloneTo
CopyTo
DeleteValue
Externalize
GetCursor
GetGenerationNumber
GetID
GetIDFromStorageUnitRef
GetName
GetOffset
GetPromiseValue
GetProperty
GetSize
GetStorageUnit
GetStorageUnitRefIterator
GetStrongStorageUnitRef
GetType
GetValue
GetWeakStorageUnitRef
IncrementGenerationNumber
InitStorageUnitView
InsertValue
Internalize
IsPromiseValue
IsStrongStorageUnitRef
IsWeakStorageUnitRef


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

```
Purge
Remove
RemoveStorageUnitRef
SetName
SetOffset
SetPromiseValue
SetType
SetValue
```

Detailed descriptions of these methods follow, in alphabetical order.

## XMPStorageUnitView

This function is the constructor of the XMPStorageUnitView object.

```
XMPStorageUnitView();
```

*return value* this

**DESCRIPTION**

Initialization. Public.

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

After this method executes successfully, this is an uninitialized XMPStorageUnitView object.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## ~XMPStorageUnitView

This method releases the XMPStorageUnit with which this XMPStorageUnitView object is associated and deletes the cursor which provides the context.

```
~XMPStorageUnitView();
```

**DESCRIPTION**

Cleanup. Public.

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

This is the destructor of the XMPStorageUnitView object.

Note that this method does not externalize the XMPStorageUnit.

After this method executes successfully, this is no longer a valid XMPStorageUnitView object.

## AddProperty

Adds a Property to the Storage Unit this XMPStorageUnitView is viewing.

```
XMPStorageUnitView* AddProperty
                              (XMPPropertyName propertyName);
```

`propertyName`***

**DESCRIPTION**

Storage Unit Manipulation. Public.

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

The resulting XMPStorageUnit will have the new Property as the focus even though the XMPStorageUnitView will maintain the cursor.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Before you call this method, its XMPStorageUnitCursor must correspond to a particular Property of the XMPStorageUnit. After this method executes successfully, the viewed XMPStorageUnit has a focus on the new Property.

## AddValue

Adds a Value to the XMPStorageUnit it is viewing.

```
XMPStorageUnitView* AddValue (XMPValueType type);
```

**DESCRIPTION**

Storage Unit Manipulation. Public.

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

The resulting XMPStorageUnit will have a new Value with the focus on it. This XMPStorageUnitView will maintain its cursor.

Before you call this method, its XMPStorageUnitCursor must correspond to a particular value of the XMPStorageUnit. After this method executes successfully, the viewed XMPStorageUnit has a view on the added Value.

## CloneInto

Deep-copy all Properties and Values of the Storage Unit this XMPStorageUnitView is viewing to the specified destination Storage Unit.

```
void CloneInto (XMPDraftKey key,
                XMPStorageUnit* destSU,
                XMPStorageUnit* initiatingFrameSU);
```

`key` XMPDraftKey identifying the Clone transaction.

`destSU` Destination Storage Unit to where this Storage Unit is copied.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`initiatingFrameSU`
: All Storage Units cloned from this point on should be within the scope of this XMPFrame.

**DESCRIPTION**

Storage Unit Management. Public.

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

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.

## CloneTo

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.

```
XMPStorageUnit* CloneTo
                      (XMPDraftKey key,
                       XMPDraft* toDraft,
                       XMPStorageUnit* initiatingFrameSU);
```

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.

*return value*



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Object Storage. Public.

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

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.

After this method executes successfully, the focus of the viewed XMPStorageUnit is undefined.

## CopyTo

Shallow copies properties and values from the XMPStorageUnit this XMPStorageUnitView is viewing to toSU.

```
void CopyTo (XMPStorageUnit* toSU);
```

`toSU` the storage unit to which data is to be copied

**DESCRIPTION**

Object Storage. Public.

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

After this method executes successfully, 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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## DeleteValue

Delete length bytes at the current offset (inclusive) in the focused Value.

```
void DeleteValue (XMPULong length);
```

| | |
|---|---|
| `offset` | the offset into the focussed value |
| `length` | the length of bytes to delete from the focussed value |

**DESCRIPTION**

Object Storage. Public.

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

If the focused Value is a Promise Value, the Promise is resolved first before insertion is done.

Before you call this method, its XMPStorageUnitCursor must correspond to a particular value of the XMPStorageUnit. After this method executes successfully, 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

Makes all changes of the XMPStorageUnit this XMPStorageUnitView object is viewing persistent.

```
XMPStorageUnitView* Externalize();
```

| | |
|---|---|
| *return value* | this |

**DESCRIPTION**

ObjectStorage. Public.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

After this method executes successfully, 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

Returns the XMPStorageUnitCursor used to focus the XMPStorageUnit.

```
XMPStorageUnitCursor* GetCursor();
```

*return value* XMPStorageUnitCursor which provides the context for this XMPStorageUnitView object.

**DESCRIPTION**

Object Storage. Public.

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

## GetGenerationNumber

Returns the generation number of the Value this XMPStorageUnitView object is viewing.

```
XMPULong GetGenerationNumber();
```

*return value* Generation number of the focused value

**DESCRIPTION**

Generation. Public.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

Before you call this method, its XMPStorageUnitCursor must correspond to a particular value of the XMPStorageUnit. After this method executes successfully, the focus of the viewed XMPStorageUnit is on the context specified by the cursor of this XMPStorageUnitView object.

## GetID

Returns the persistent ID of the XMPStorageUnit object this XMPStorageUnitView object is viewing.

```
XMPID GetID();
```

*return value* ID of the XMPStorageUnit object this object is viewing.

**DESCRIPTION**

ObjectStorage. Public.

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

## GetIDFromStorageUnitRef

Resolves the given ref to a XMPStorageUnitID and returns the XMPStorageUnitID.

```
XMPStorageUnitID GetIDFromStorageUnitRef
                                    (XMPStorageUnitRef ref);
```

`ref` persistent reference to be resolved.

*return value* ID of the embedded StorageUnit.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Storage Unit Reference. Public.

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

If the specified ref cannot be resolved, an exception is raised.

Before you call this method, its XMPStorageUnitCursor must correspond to a particular value of the XMPStorageUnit. ref must be generated by GetStrongStorageUnitRef or GetWeakStorageUnitRef. After this method executes successfully, the focus of the viewed XMPStorageUnit is on the context specified by the cursor of this XMPStorageUnitView object.

**EXCEPTIONS**

`kXMPNotValueFocussed`

## GetName

Returns the name of the Storage Unit this XMPStorageUnitView is viewing.

```
XMPName* GetName();
```

*return value* the name of the focus

**DESCRIPTION**

ObjectStorage. Public.

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

## GetOffset

Return the offset of the Value this XMPStorageUnitVeiw is viewing.

```
XMPULong  GetOffset();
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

*return value* Offset of the current Value.

**DESCRIPTION**

ObjectStorage. Public.

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

Before you call this method, its XMPStorageUnitCursor must correspond to a particular Value of the Storage Unit.

**EXCEPTIONS**

`kXMPNotValueFocussed`

## GetPromiseValue

Get the promise data in the Value this XMPStorageUnitView object is viewing. This does not resolve the promise.

```
XMPULong GetPromiseValue   (XMPValueType valueType,
                            XMPULong offset,
                            XMPULong length,
                            XMPValue value,
                            XMPPart **sourcePart);
```

| | |
|---|---|
| `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, and returns the Buffer containing the promise data. |
| *return value* | Number of bytes read. |

**DESCRIPTION**

ObjectStorage. Public.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

Before you call this method, its XMPStorageUnitCursor must correspond to a particular value of the XMPStorageUnit. After this method executes successfully, 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

Returns the Property Name in the cursor of this XMPStorageUnitView object.

```
XMPPropertyName     GetProperty();
```

*return value* Property name.

**DESCRIPTION**

ObjectStorage. Public.

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

Before you call this method, its XMPStorageUnitCursor must correspond to a particular property or value of the XMPStorageUnit. After this method executes successfully, the focus of the viewed XMPStorageUnit is on the context specified by the cursor of this XMPStorageUnitView object.

## GetSize

Returns the size of the focused Property or Value this XMPStorageUnitView object is viewing.

```
XMPULong GetSize();
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

*return value* Size of the focused Property or Value this XMPStorageUnitView object is viewing.

**DESCRIPTION**

Storage Unit Manipulation. Public.

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

## GetStorageUnit

Returns the XMPStorageUnit that this XMPStorageUnitView is viewing.

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

*return value* XMPStorageUnit this XMPStorageUnitView is viewing.

**DESCRIPTION**

Object Storage. Public.

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

The returned XMPStorageUnit should be focused using the XMPStorageUnitCursor stored in this XMPStorageUnitView object.

## GetStorageUnitRefIterator

```
XMPStorageUnitRefIterator* GetStorageUnitRefIterator();
```

*return value* Iterator used to iterate over all persistent references in the focused Value.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Iteration. Public.

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

Returns an iterator for iterating over all the persistent references created in the Value this XMPStorageUnitRefIterator is viewing.

## GetStrongStorageUnitRef

Returns a persistent reference to the Storage Unit referred to by embeddedSU.

```
XMPStorageUnitRef GetStrongStorageUnitRef
                                    (XMPStorageUnit* embeddedSU);
```

`embeddedSU` XMPStorageUnit referring to the Storage Unit whose persistent reference is desired.

*return value* persistent reference to the Storage Unit referred to by embeddedSU.

**DESCRIPTION**

Storage Unit Reference. Public.

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

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.

Before you call this method, its XMPStorageUnitCursor must correspond to a particular value of the XMPStorageUnit. After this method executes successfully, the focus of the viewed XMPStorageUnit is on the context specified by the cursor of this XMPStorageUnitView object.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetType

Returns the type of the Value in the cursor of this XMPStorageUnitView object.

```
XMPValueType GetType();
```

*return value* the type of the focused particular value

**DESCRIPTION**

ObjectStorage. Public.

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

Before you call this method, its XMPStorageUnitCursor must correspond to a particular value of the XMPStorageUnit. After this method executes successfully, The focus of the viewed XMPStorageUnit is on the context specified by the cursor of this XMPStorageUnitView object.

## GetValue

Reads length bytes from current offset of the Value this XMPStorageUnitView object is viewing into value.

```
XMPULong GetValue (XMPULong length,
                   XMPValue value);
```

`offset` the offset into the focussed value

`length` the length of bytes to read into value

`value` a buffer of size >= length and returns the Buffer containing the read data.

*return value* Number of bytes actually read.

**DESCRIPTION**

ObjectStorage. Public.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

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.

Before you call this method, its XMPStorageUnitCursor corresponds to a particular value of the XMPStorageUnit. value is a buffer of length length or more. After this method executes successfully, 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

Returns a persistent reference to the Storage Unit refered to by embeddedSU.

```
XMPStorageUnitRef GetWeakStorageUnitRef
                                (XMPStorageUnit* embeddedSU);
```

`embeddedSU` XMPStorageUnit object referring to the Storage Unit whose persistent reference is desired.

*return value* Persistent reference to the Storage Unit referred to by embeddedSU.

### DESCRIPTION

Storage Unit Reference. Public.

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

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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Before you call this method, its XMPStorageUnitCursor must correspond to a particular value of the XMPStorageUnit. After this method executes successfully, the focus of the viewed XMPStorageUnit is on the context specified by the cursor of this XMPStorageUnitView object.

## IncrementGenerationNumber

Increment and returns the generation number of the Value this XMPStorageUnitView object is viewing.

```
XMPULong IncrementGenerationNumber();
```

*return value* Generation number of the Storage Unit

**DESCRIPTION**

ObjectStorage. Public.

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

Before you call this method, its XMPStorageUnitCursor must correspond to a particular value of the XMPStorageUnit. After this method executes successfully, the focus of the viewed XMPStorageUnit is on the context specified by the cursor of this XMPStorageUnitView object.

## InitStorageUnitView

Initializes this XMPStorageUnitView by storing the parameters.

```
void InitStorageUnitView   (XMPStorageUnit* storageUnit,
                            XMPStorageUnitCursor* cursor);
```

`storageUnit`
XMPStorageUnit this XMPStorageUnitView is associated with.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`cursor` Context (Property and/or Value) in which the XMPStorageUnitView methods should work.

**DESCRIPTION**

None. Public.

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

After this method executes successfully, this is an initialized XMPStorageUnitView object.

## InsertValue

Inserts length bytes at the current offset in the value this XMPStorageUnitView object is viewing.

```
void InsertValue  (XMPULong length,
                   XMPValue value);
```

`length` the length of bytes to write into the focussed value

`value` a buffer of length>= length

**DESCRIPTION**

ObjectStorage. Public.

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

If the focused Value contains a Promise, the Promise is resolved first before insertion is done.

Before you call this method, its XMPStorageUnitCursor corresponds to a particular value of the XMPStorageUnit. value is a buffer of length length or more. After this method executes successfully, length bytes of value have been inserted at offset in the currently focused value



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

The focus of the viewed XMPStorageUnit is on the context specified by the cursor of this XMPStorageUnitView object.

## Internalize

(For embedding application or document shell only.)
Internalizes the XMPStorageUnit object this XMPStorageUnitView object is viewing.

```
XMPStorageUnitView* Internalize();
```

*return value* this

**DESCRIPTION**

ObjectStorage. Public. Private within Storage System.

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

After this method executes successfully, all properties and values in the focus have been brought into ephemeral storage from persistent storage

## IsPromiseValue

Returns `kXMPTrue` if the Value this XMPStorageUnitView object is viewing contains a Promise. Otherwise, returns `kXMPFalse`.

```
XMPBoolean IsPromiseValue();
```

*return value* XMPBoolean showing whether the focused Value contains a Promise.

**DESCRIPTION**

Promise. Public.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

If the focused Value is a Promise, the Promise will not be resolved by this call.

Before you call this method, its XMPStorageUnitCursor must correspond to a particular value of the XMPStorageUnit. After this method executes successfully, The focus of the viewed XMPStorageUnit is on the context specified by the cursor of this XMPStorageUnitView object.

## IsStrongStorageUnitRef

Returns `kXMPTrue` if ref is a strong XMPStorageUnitRef. Otherwise, returns `kXMPFalse`.

```
XMPBoolean IsStrongStorageUnitRef (XMPStorageUnitRef ref);
```

| | |
|---|---|
| `ref` | XMPStorageUnitRef to be tested. |
| *return value* | Boolean showing whether ref is a strong XMPStorageUnitRef. |

**DESCRIPTION**

Storage Unit Reference. Public.

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

Before you call this method, its XMPStorageUnitCursor must correspond to a particular value of the XMPStorageUnit. After this method executes successfully, the focus of the viewed XMPStorageUnit is on the context specified by the cursor of this XMPStorageUnitView object.

## IsWeakStorageUnitRef

Returns `kXMPTrue` if ref is a weak XMPStorageUnitRef. Otherwise, returns `kXMPFalse`.

```
XMPBoolean IsWeakStorageUnitRef (XMPStorageUnitRef ref);
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `ref` | XMPStorageUnitRef to be tested. |
| *return value* | Boolean showing whether the ref is a weak XMPStorageUnitRef. |

**DESCRIPTION**

Storage Unit Reference. Public.

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

Before you call this method, its XMPStorageUnitCursor corresponds to a particular value of the XMPStorageUnit. After this method executes successfully, the focus of the viewed XMPStorageUnit is on the context specified by the cursor of this XMPStorageUnitView object.

## Purge

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

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

| | |
|---|---|
| `size` | the number of bytes to purge |
| *return value* | the number of bytes actually purged |

**DESCRIPTION**

ObjectStorage. Public.

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

After this method executes successfully, 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.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Remove

Removes the value or property this XMPStorageUnitView object is viewing.

```
XMPStorageUnitView* Remove();
```

*return value* this

**DESCRIPTION**

ObjectStorage. Public.

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

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.

Before you call this method, its XMPStorageUnitCursor corresponds to a particular property or value of the XMPStorageUnit. After this method executes successfully, the property or value at the cursor focus is removed

The focus of the viewed XMPStorageUnit is undefined.

## RemoveStorageUnitRef

Removes a XMPStorageUnitRef from a Value.

```
XMPStorageUnitView* RemoveStorageUnitRef
                                    (XMPStorageUnitRef ref);
```

`ref` persistent reference to a Storage Unit.
*return value* this XMPStorageUnitView object.

**DESCRIPTION**

Storage Unit Reference. Public.

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


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Before you call this method, its XMPStorageUnitCursor must correspond to a particular value of the XMPStorageUnit. ref is a valid XMPStorageUnitRef (either strong or weak). After this method executes successfully, ref is no longer a valid XMPStorageUnitRef in that Value.

## SetName

Set the name of the Storage Unit this XMPStorageUnitView is viewing.

```
void SetName (XMPName name);
```

`name` the name of the focus

**DESCRIPTION**

ObjectStorage. Public.

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

After this method executes successfully, the name of the focus = name

## SetOffset

Sets the offset of the value this XMPStorageUnitView object is viewing.

```
void SetOffset (XMPULong offset);
```

`offset` Offset to use with the focused value

**DESCRIPTION**

Value Manipulation. Public.

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


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Before you call this method, its XMPStorageUnitCursor must correspond to a particular value of the Storage Unit. After this method executes successfully, the current offset for value operations is offset.

## SetPromiseValue

Put the given data into the value this XMPStorageUnitView is viewing and make the viewed value a promise.

```
void SetPromiseValue (XMPValueType valueType,
                      XMPULong offset,
                      XMPULong length,
                      XMPValue value);
```

| | |
|---|---|
| `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. |

**DESCRIPTION**

Promise. Public.

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

Before you call this method, its XMPStorageUnitCursor must correspond to a particular value of the XMPStorageUnit. After this method executes successfully, 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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## SetType

Sets the type of the Value of the Storage Unit this XMPStorageUnitView is viewing.

```
void SetType (XMPValueType valueType);
```

| | |
|---|---|
| `valueType` | Type of the focused Value. |
| *return value* | Exceptions Signalled |

**DESCRIPTION**

Value manipulation. Public.

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

Before you call this method, its XMPStorageUnitCursor must correspond to a particular value of the XMPStorageUnit. After this method executes successfully, 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

Write length bytes to the current offset of the Value this XMPStorageUnitView is viewing.

```
void SetValue  (XMPULong length,
                XMPValue value);
```

| | |
|---|---|
| `length` | the length of bytes to write into the focussed value |
| `value` | a buffer of length>= length |

**DESCRIPTION**

ObjectStorage. Public.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

If length + offset is greater than the current size of the Value, the Value should be grown to accommodate the extra bytes.

Before you call this method, Its XMPStorageUnitCursor corresponds to a particular value of the XMPStorageUnit.

value is a buffer of length length or more

After this method executes successfully, 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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPTransform

*Superclasses* `XMPObject`

*Subclasses* none

XMPTransform objects are used to define transformations between coordinate spaces in different XMPFrames and XMPFacets. More background information can be found in the XMPFrame and XMPFacet class documentation.

## Description

Base class: XMPObject.

Class XMPTransform has no friends.

More background information can be found in the XMPFrame class documentation.

There are different flavors of Transforms for different graphics models. For example, 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 matrixes, resulting in an "aggregate transform". See any standard computer graphics text for gory details. It is possible for subclasses of XMPTransform to be defined that do not use transformation matrices. These might perform more complex operations like morphing or wrapping around a 3D surface. The public API is general enough to support these transforms, but the implementation will have to be modified to make them possible.

Invariants: fMatrix holds a transformation matrix.

fInverseMatrix points to a cached inverse-transformation matrix (the inverse of fMatrix). It is generated when needed, and cleared when fMatrix changes or a memory-purge happens.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Methods

XMPTransform
~XMPTransform
ConvertTransform
Copy
CopyFrom
GetGraphicsSystem
GetMatrix
GetOffset
GetPlatformTransform

GetPreScaleOffset
GetScale
GetType
HasMatrix
Invert
InvertPoint
InvertShape
IsSameAs
MoveBy
PostCompose
PreCompose
ReadFrom
Reset
ScaleBy
ScaleDownBy
SetMatrix
SetOffset
SetPlatformTransform
TransformPoint
TransformShape
WriteTo


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Detailed descriptions of these methods follow, in alphabetical order.

## XMPTransform

Constructs a new XMPTransform object representing the identity transform.

```
XMPTransform();
```

*return value* an `XMPTransform` object.

**DESCRIPTION**

Imaging. Public. No restrictions.

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

Before you call this method, you must ensure that enough memory is available to allocate the `XMPTransform` object. After this method executes successfully, the object is allocated and initialized to a state where it can safely be deleted.

**EXCEPTIONS**

`kXMPErrOutOfMemory`

## ~XMPTransform

Deletes an `XMPTransform` object.

```
~XMPTransform();
```

**DESCRIPTION**

Imaging. Public. No restrictions.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

Before you call this method, you must ensure that an object has been created.

## ConvertTransform

Return a copy of `fPlatformShape` converted into the specified graphics system's representation.

```
XMPPlatformTransform ConvertTransform
                          (XMPGraphicsSystem graphicsSystem);
```

graphicsSystem
: The graphics system format to transform the transform's representation into.

*return value* The `platformTransform` in the appropriate format.

### DESCRIPTION

Imaging. Public. No restrictions.

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

Always returns a copy of the shape data, even if the requested graphics system is the same as the internal representation of `fPlatformShape`. Caller assumes responsibility for returned storage. Before you call this method, you must ensure that the transform data has been initialized (see `SetPlatformTransform`)

### EXCEPTIONS

`kXMPErrInvalidGraphicsSystem`
: Not a supported graphics system.

`kXMPErrOutOfMemory` Not enough memory to allocate converted shape.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Copy

Creates and returns a new XMPTransform object identical to the receiver.

```
XMPTransform* Copy();
```

*return value* A new XMPTransform object identical to the receiver.

**DESCRIPTION**

Imaging. Public.

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

## CopyFrom

Updates this `XMPTransform` object's platform transform data so that it represents an equivalent transform as referred to by the parameter.

```
void CopyFrom (XMPTransform* sourceTransform);
```

`sourceTransform`
Pointer to another `XMPTransform` object. Caller's storage responsibility.

**DESCRIPTION**

Imaging. Public. No restrictions.

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

Before you call this method, you must ensure that `sourceTransform` is a valid instance of `XMPTransform`. After this method executes successfully, this object is now equivalent to the `sourceTransform` object.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrInvalidXMPTransform` | Bad input parameter. |
| `kXMPErrOutOfMemory` | Not enough memory to copy transform data. |

## GetGraphicsSystem

Returns the cached value.

```
XMPGraphicsSystem GetGraphicsSystem();
```

*return value* of type `XMPGraphicsSystem`: `= kXMPQuickDraw`, if Macintosh' QuickDraw, `= kXMPQuickDrawGX`, if QuickDraw GX.

**DESCRIPTION**

Imaging. Public. No restrictions.

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

## GetMatrix

Copies the entire transformation matrix as an XMPMatrix into the 'm' parameter.

```
void GetMatrix(
XMPMatrix *m );
```

**DESCRIPTION**

Imaging. Public.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

## GetOffset

Returns the offset portion of the transformation in the 'offset' parameter.

```
void GetOffset(
XMPPoint *offset );
```

**DESCRIPTION**

Imaging. Public.

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

This corresponds to the first two elements of the bottom row of the matrix.

## GetPlatformTransform

Returns the platform- or graphics-system-specific transform data structure.

```
XMPPlatformTransform GetPlatformTransform();
```

*return value* a value of type `XMPPlatformTransform`. On the Macintosh, this is a pointer to a 3 by 3 array of Fixed values for `kXMPQuickDraw`. Caller must not de-allocate this data.

**DESCRIPTION**

Imaging. Public. No restrictions.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

This may or may not be allocated from the heap; it depends on the particular graphics system. Also graphics-system-dependent and implementation-dependent is whether the data is a copy or the original. In the Mac implementation, it is the original data and the caller must not dispose or otherwise consume it. (In particular, in the QuickDraw graphics system, the data is a 32-bit Point structure, not a pointer to anything.) Before calling this method, you must ensure that its storage member points to allocated transform data. After executing this method successfully, its storage member points to allocated transform data.

#### EXCEPTIONS

`kXMPErrInvalidPlatformTransform`

`kXMPErrUnavailablePlatformTransform`
: Translation of the transform data for this graphics system is not available.

## GetPreScaleOffset

Returns the offset scaled down by the scale portion of the transform.

```
void GetPreScaleOffset(
XMPPoint *offset );
```

#### DESCRIPTION

None. Public.

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

This is useful for clients that want to transform their data by offsetting first and then scaling: this is the offset they should use. It doesn't make much sense if the transformation involves rotation, skewing, or perspective.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetScale

Returns the scaling factor of this transformation in the 'scale' parameter. This is equivalent to the [0,0] and [1,1] elements of the matrix.

```
void GetScale(
XMPPoint *scale );
```

**DESCRIPTION**

Imaging. Public.

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

## GetType

Returns the type of transformation, as one of the values of the XMPTransformType enumeration.

```
XMPTransformType GetType();
```

*return value* The type of this transformation.

**DESCRIPTION**

Imaging. Public.

Derived class **can** override. Derived class **can** call base class behavior, after derived class behavior.

## HasMatrix

Use HasMatrix to find out if an XMPTransform is based on a transformation ma-


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

trix. (If not, the GetMatrix call will raise an exception.)

```
XMPBoolean HasMatrix();
```

*return value* kXMPTrue if transform has a matrix, else kXMPFalse.

**DESCRIPTION**

Imaging. Public.

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

Currently all transforms are matrix-based and this call always returns true. However, in the future there may be complex transforms that cannot be represented as matrices.

## Invert

Turns this `XMPTransform` object into its inverse.

```
XMPTransform* Invert();
```

*return value* Pointer to this `XMPTransform` object which is turned into its inverse.

**DESCRIPTION**

Imaging. Public. No restrictions.

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

Returns the receiver so you can easily append another call to this one.

After calling this method successfully, the receiver now represents the inverse transformation.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

`kXMPErrOutOfMemory`
Not enough memory to allocate inverse matrix.

## InvertPoint

The opposite of `TransformPoint`: transforms a point by the inverse of this transform.

```
XMPPoint InvertPoint (XMPPoint point);
```

`point` `XMPPoint` data structure.

*return value* Transformed point.

**DESCRIPTION**

Imaging. Public. No restrictions.

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

The transform itself is left unchanged, unlike Invert. Before calling this method, you must ensure that the transform is invertible. After executing this method successfully, the returned `Point` is the input point transformed by the inverse of the transform.

**EXCEPTIONS**

`kXMPErrOutOfMemory`
Not enough memory to allocate inverse matrix.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## InvertShape

Transforms the shape through the inverse of this transformation.

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

**DESCRIPTION**

Imaging. Public.

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

This is equivalent to the InverseTransform method in XMPShape.

## IsSameAs

Comparison of two`XMPTransform` objects.

```
XMPBoolean IsSameAs (XMPTransform* compareTransform);
```

`compareTransform`
Pointer to an `XMPTransform` object. Caller 's storage responsibility.

*return value* `XMPBoolean`; `kXMPTrue`, if different, `kXMPFalse` otherwise.

**DESCRIPTION**

Imaging. Public. No restrictions.

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

Before calling this method, you must ensure that the `fPlatformTransform` points to a platform-specific transform data structure, and that the `compareTransform` object caches a pointer to a valid platform-specific transform data structure.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## MoveBy

Offsets the transformation by the given amount.

```
XMPTransform* MoveBy(
XMPPoint offset );
```

`offset` Amount to offset by.

*return value* Returns the receiver.

### DESCRIPTION

Imaging. Public.

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

This is just like post-composing the transform with an offset matrix.

## PostCompose

Update this transform using matrix postmultiplication.

```
XMPTransform* PostCompose (XMPTransform* transform);
```

`transform` Pointer to `XMPTransform`. Caller 's storage responsibility.

*return value* Pointer to this `XMPTransform` object.

### DESCRIPTION

Imaging. Public. No restrictions.

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

Update this transform according to the matrix multiplication:

`this = this * transform`.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Returns a pointer to the transform object ('this') as a convenience to the caller.

## PreCompose

Update this transform using matrix premultiplication:

```
XMPTransform* PreCompose (XMPTransform* transform);
```

`transform` Pointer to `XMPTransform`. Caller 's storage responsibility.

*return value* Pointer to this `XMPTransform` object.

### DESCRIPTION

Imaging. Public. No restrictions.

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

Update this transform according to the matrix multiplication:

`this = transform * this.`

Returns a pointer to the transform object ('this') as a convenience to the caller.

### EXCEPTIONS

`kXMPErrNumericError`

## ReadFrom

Reads the transformation matrix from a storage unit, which must be pre-focused on a particular property.

```
void ReadFrom(
XMPStorageUnit* );
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Imaging. Public.

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

The matrix will be read from the `kXMPTransform` value of that property.

## Reset

Turns this `XMPTransform` object into the Identity transform.

```
XMPTransform* Reset();
```

*return value* Pointer to this `XMPTransform` object.

**DESCRIPTION**

Imaging. Public. No restrictions.

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

Returns the receiver for easy concatenation with other operations.

## ScaleBy

Scales the transformation.

```
XMPTransform* ScaleBy(
XMPPoint scale ) ;
```

scale Scaling factor (x and y)

*return value* Returns the receiver.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Imaging. Public.

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

## ScaleDownBy

This is the inverse of Scale: it scales by the reciprocal of the input.

```
XMPTransform* ScaleDownBy(
XMPPoint scale );
```

| | |
|---|---|
| scale | Inverse scaling factor |
| *return value* | Returns the receiver. |

**DESCRIPTION**

Imaging. Public.

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

## SetMatrix

Makes the receiver's transformation equivalent to the supplied transformation matrix.

```
XMPTransform* SetMatrix(
XMPMatrix *matrix );
```

| | |
|---|---|
| `matrix` | Points to an input XMPMatrix specifying the new transform. |
| *return value* | Returns the receiver. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Imaging. Public.

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

If the matrix is NULL this is equivalent to calling Reset.

## SetOffset

Makes the receiver a pure offset (or translation) with the given coordinates.

```
XMPTransform* SetOffset(
XMPPoint offset );
```

`offset` Coordinates of the offset

*return value* Returns the receiver.

**DESCRIPTION**

Imaging. Public.

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

## SetPlatformTransform

Sets this object's transform based on graphics-system-dependent data.

```
void SetPlatformTransform
                  (XMPGraphicsSystem graphicsSystem,
                   XMPPlatformTransform platformTransform);
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`graphicsSystem`
of type `XMPGraphicsSystem`: `= kXMPQuickDraw`, if Macintosh' QuickDraw, `= kXMPQuickDrawGX`, if QuickDraw GX.

`platformTransform`
of type `XMPPlatformTransform`. Actual data depends on graphics system.

**DESCRIPTION**

Imaging. Public. No restrictions.

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

Sets this object's transform based on graphics-system-dependent data. (On the Mac, for QD the data is a point [NOT a pointer to a point!] while for GX the data is a pointer to a `gxMatrix`.) If the data is pointer based, it may or may not be copied, depending on the implementation. On the Macintosh, the gx matrix is not copied, and the caller must give up control and not dispose it afterwards. Before calling this method, you must ensure that the transform data the parameter is pointing to are allocated, and the transform data the storage member is pointing to are allocated. After executing this method successfully, `fPlatformTransform` has the data of the `platformTransform` parameter.

**EXCEPTIONS**

`kXMPErrInvalidPlatformTransform`

## TransformPoint

Maps a point from a source coordinate space to a destination coordinate space by applying this transform.

```
XMPPoint TransformPoint (XMPPoint point);
```

`point` `XMPPoint` data structure;



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

*return value* transformed point.

**DESCRIPTION**

Imaging. Public. No restrictions.

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

## TransformShape

Transforms the shape through this transformation.

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

**DESCRIPTION**

None. Public.

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

This is equivalent to the Transform method in XMPShape.

## WriteTo

Writes the transform (as a matrix) to the storage unit.

```
void WriteTo(
XMPStorageUnit* );
```

**DESCRIPTION**

Imaging. Public.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

The storage unit should be pre-focused to a particular property; the matrix will be written in the `kXMPTransform` value of that property.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPTranslation

*Superclasses* `XMPObject`

*Subclasses* none

XMPTranslation provides data translation capability to OpenDoc documents and their parts.

## Description

XMPTranslation 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.

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: The translation methods provided by XMPTranslation should include all those provided by the system translation service.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Methods

```
XMPTranslation
~XMPTranslation
GetISOTypeFromPlatformType
GetPlatformTypeFromISOType
GetTranslateMethod
GetTranslationOf
InitTranslation
Purge
Translate
Translate
```

Detailed descriptions of these methods follow, in alphabetical order.

### XMPTranslation

(For embedding application or document shell only.)
Creates an XMPTranslation object.

```
XMPTranslation();
```

**DESCRIPTION**

Constructor. Public. Private between XMPSession and XMPTranslation. This method should only be called by XMPSession::InitSession.

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

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

After this method executes successfully, this is an uninitialized XMPTranslation object.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## ~XMPTranslation

(For embedding application or document shell only.)

```
~XMPTranslation();
```

**DESCRIPTION**

Destructor. Public. Private between XMPSession and XMPTranslation. This method should only be called by XMPSession::~XMPSession or the equivalent shutdown method.

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

This function is the destructor of the XMPTranslation object. It is called once when the process quits. It closes the system translation service.

## GetISOTypeFromPlatformType

Returns the ISO type corresponding to the argument platform type in the context of the argument type space.

```
XMPType GetISOTypeFromPlatformType
                        (XMPPlatformType platformType,
                         XMPPlatformTypeSpace typeSpace);
```

`platformType` The platform specific type

`typeSpace` The context in which to interprete the type

*return value* The ISO type

**DESCRIPTION**

Translation. Public.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetPlatformTypeFromISOType

This function returns the platform type corresponding to the argument ISO type.

```
XMPPlatformType GetPlatformTypeFromISOType (XMPType type);
```

`type` The ISO type.

*return value* The platform type.

**DESCRIPTION**

Translation. Public.

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

## GetTranslateMethod

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.)

```
XMPBoolean GetTranslateMethod
                    (XMPTypeSet* givenTypes,
                     XMPType nativeType,
                     XMPTranslateResult* result,
                     XMPTranslateMethod* howToTranslate);
```

`givenTypes` Types of the source data.

`nativeType` Type that the part can handle.

`result` `kCanTranslate` kCannotTranslate ???

`howToTranslate`
Hints used in XMPTranslation::Translate for the actual translation. Callers should use this as a blackbox.

*return value* true if translation to nativeType is possible



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Translation. Public.

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

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.

Before you call this method, givenTypes must be a non-empty set.

**EXCEPTIONS**

`kXMPErrEmptySet`
: givenTypes is empty.

## GetTranslationOf

This function returns a XMPTypeList that contains all the types to which `fromType` can be translated.

```
XMPTypeList* GetTranslationOf (XMPType fromType);
```

`fromType` Type of the source data.

*return value* A list of types that can be translated to from the source data.

**DESCRIPTION**

Translation. Public.

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

If translation cannot be done on fromType, an empty set is returned.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

After this method executes successfully, `fromType` is unchanged.

#### EXCEPTIONS

`kXMPErrOutOfMemory`
: Cannot create a XMPTypeSet because the system is running out of memory.

## InitTranslation

(For embedding application or document shell only.)
Initializes the system translation service.

```
void InitTranslation();
```

#### DESCRIPTION

Constructor. Public. Private between XMPSession and XMPTranslation. This method should only be called by XMPSession::InitSession.

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

It is called once at the process' startup time.

After this method executes successfully, this is an initialized XMPTranslation object.

## Purge

(For embedding application or document shell only.)
Purges memory from ephemeral store until size bytes (not necessarily contiguous) have been freed up.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `size` | Number of bytes to purge. |
| *return value* | Number of bytes actually purged. |

**DESCRIPTION**

Memory Management. Public. Private between XMPSession and XMPTranslation. This should only be called by XMPSession.

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

## Translate

Translates the Value viewed by fromView and stores the translated data in the Value viewed by toView.

```
XMPTranslateResult Translate(
XMPStorageUnitView* fromView,
XMPStorageUnitView* toView);
```

| | |
|---|---|
| `fromView` | A reference to a value that contains the data to be translated. |
| *return value* | Result of translation. |
| | `toView`<br>A reference to a value where the translated data should be stored. |

**DESCRIPTION**

Translation. Public.

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

**EXCEPTIONS**

`kXMPErrBadTranslationMethod`
Corrupted XMPTranslationMethod.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`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.

## Translate

Translates the data in the buffer pointed to by fromData (of length fromDataSize) and stores the translated data in the buffer pointed by toData.

```
XMPVMethod void Translate
                        (XMPPtr fromData,
                         XMPULong fromDataSize,
                         XMPType fromType,
                         XMPType    toType,
                         XMPPtr* toData,
                         XMPULong* toDataSize);
```

`fromData` Source data to be translated.

`fromDataSize`
Size of data to be translated.

`fromType` Type of source data.

`toType` Type of data desired.

*return value* Result of translation (i.e., whether the translation succeeds).

`toData`
Pointer to buffer used to contain translated data.

`toDataSize`
Size of translated data.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Translation. Public.

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

After this method executes successfully, the content referred to by fromData is unchanged.

**EXCEPTIONS**

`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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPTypeList

*Superclasses* none

*Subclasses* none

XMPTypeList is an ordered collection class for XMPType objects.

## Description

XMPTypeList is an ordered 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 list.

Parts may create instances of this class. Subclassing of XMPTypeList is not anticipated.

The main clients of this class are XMPTranslation and parts. XMPTranslation class uses XMPTypeList as parameters to its functions. Parts can use XMPTypeList to store a set of Types for repeated uses.

XMPTypeList uses OpenDoc's private collection class.

Invariants: Every item in a XMPTypeList object is unique. Therefore, a XMPTypeList object contains a set of different XMPTypes.

A XMPTypeList object can contain zero or more XMPTypes. In other words, the type set can either be empty or non-empty.

## Methods

`XMPTypeList`

`~XMPTypeList`

`Add`

`AddLast`

`Contains`

`Count`



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

```
CreateTypeListIterator

InitTypeList
Remove
```

Detailed descriptions of these methods follow, in alphabetical order.

## XMPTypeList

This method is the constructor of the class.

```
XMPTypeList();
```

**DESCRIPTION**

Creation and Deletion. Public.

Derived class **cannot** override. Derived class **must** call base class behavior, before derived class behavior. After calling this method successfully, memebers are initialized to a known state.

## ~XMPTypeList

This method is the destructor of the class.

```
~XMPTypeList();
```

**DESCRIPTION**

Creation and Deletion. Public.

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

Before you call this method, you must ensure that the list has been created.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Add

Adds the argument type to the set.

```
void Add (XMPType type);
```

`type` The type to add to the set.

**DESCRIPTION**

None.. Public.

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

If the type is already present in the set, no action is taken.

**EXCEPTIONS**

`kXMPErrOutOfMemory`
Could not add the argument type to the set.

`kXMPErrInvalidType`
Argument type is not a valid XMPType.

## AddLast

Adds a copy of the argument type to the end of the list.

```
void AddLast(
XMPType type);
```

`type` The type to add to the list.

**DESCRIPTION**

None. Public.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

If the type is already present in the list, no action is taken.

Before calling this method, you must ensure that the list has been initialized. After calling this method successfully, the list contains the argument type.

**EXCEPTIONS**

`kXMPErrOutOfMemory`
Could not add the argument type to the list.

`kXMPErrInvalidType`
Argument type is not a valid XMPType.

## Contains

Test this list for the presence of the argument type.

```
XMPBoolean Contains (XMPType type);
```

`type` The type to test for inclusion in the list.

*return value* `kXMPTrue` if the argument type is in the set and `kXMPFalse` otherwise.

**DESCRIPTION**

None. Public.

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

Before calling this method, you must ebnsurew that the list has been initialized.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Count

This function returns the number of types in the list. If the list is empty, zero is returned.

```
XMPULong Count();
```

*return value* Number of types in the list.

**DESCRIPTION**

None. Public.

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

Before calling this method, you must ebnsurew that the list has been initialized.

## CreateTypeListIterator

Return an iterator for this list.

```
XMPTypeListIterator* CreateTypeListIterator();
```

*return value* An iterator for this list.

**DESCRIPTION**

Iteration. Public.

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

The client should dispose of the iterator using the delete operator.

Before calling this method, you must ebnsurew that the list has been initialized.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

`kXMPErrOutOfMemory`
Could not create the iterator.

## InitTypeList

Initializes data members and sets the list to a copy of 'initList'.

```
void InitTypeList(
XMPTypeList* initList);
```

`initList` A list of XMPTypes from which to initialize this object. A `kXMPNull` value results in an intially empty list.

**DESCRIPTION**

Initialization. Public.

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

Before calling this method, you must ensure that the list has been created. After calling this method successfully, the list has been initialized to a copy of 'initList'.

**EXCEPTIONS**

`kXMPErrOutOfMemory` Could not initialize the list.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## InitTypeList

Initializes data members and sets the list to a copy of 'initList'.

```
void InitTypeList(
XMPTypeList* initList,
XMPHeap where);
```

| | |
|---|---|
| `initList` | A list of XMPTypes from which to initialize this object. A `kXMPNull` value results in an intially empty list. |
| `where` | The heap to use for memory allocation. |

**DESCRIPTION**

Initialization. Public.

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

Before calling this method, you must ensure that the list has been created. After calling this method successfully, the list has been initialized to a copy of 'initList'.

**EXCEPTIONS**

| | |
|---|---|
| kXMPErrOutOfMemory | Could not initialize the list. |

## Remove

Removes the argument type from the list.

```
void Remove (XMPType type);
```

| | |
|---|---|
| `type` | Type to be removed. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

None. Public.

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

If the type is not in the list, no action is taken.

Before calling this method, you must ensure that the list has been initialized. After this method executes successfully, the argument type is not in the list.



**Draft. Preliminary. © Apple Computer, Inc. 6/20/94**

# XMPTypeListIterator

*Superclasses* none
*Subclasses* none

XMPTypeListIterator is an iterator for the class XMPTypeList.

## Description

XMPTypeListIterator has no base class.

This class is used to iterate over the elements of a XMPTypeList instance in order. Iteration is forward only, from first to last. The iteration can be restarted at any time by calling the First method. If the XMPTypeList 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 XMPTypeListIterator by calling the CreateTypeListIterator method of the XMPTypeList instance that is to be iterated over. The iterator should be disposed of using the delete operator.

Subclassing of XMPTypeListIterator is not anticipated.

The implementation of XMPTypeListIterator uses OpenDoc's private collection class.

## Methods

```
~XMPTypeListIterator
First
IsNotComplete
Next
```

Detailed descriptions of these methods follow, in alphabetical order.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## ~XMPTypeListIterator

Class instance destructor.

```
~XMPTypeListIterator();
```

**DESCRIPTION**

Creation and Deletion. Public.

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

After this method executes successfully, the iterator has been created.

## First

Initiates iteration and returns the first XMPTypeObject in the list.

```
XMPType First();
```

| | |
|---|---|
| *return value* | The first XMPType object in the list. If the list is empty kXMPNULL is returned, although the client should test for completion by calling IsNotComplete. |

**DESCRIPTION**

Iteration. Public.

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

This method does not return a copy of the item, so the client must not dispose the returned object. After creating an iterator using XMPTypeList::CreateTypeListIterator, call this method once, then call Next repeatedly to iterate through all items in the list. First can be called at any time to restart the iteration.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Before calling this method, you must ensure that the iterator was created. After executing this method successfully, the first item in the list is returned, or `kXMPNull` if the list is empty.

**EXCEPTIONS**

`kXMPErrIteratorOutOfSync`
The XMPTypeList instance was changed during the iteration.

## IsNotComplete

Test if iteration is complete.

```
XMPBoolean IsNotComplete();
```

*return value* Returns `kXMPTrue` if the last call to First or Next returned a valid XMPType element, and `kXMPFalse` otherwise.

**DESCRIPTION**

Iteration. Public.

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

Before calling this method, you must ensure that the list has been created and that `First()` has been called to start the iteration.

## Next

Returns the next element from the ordered list.

```
XMPType Next();
```

*return value* ***



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Iteration. Public.

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

This method does not return a copy of the item, so the client must not dispose the returned object. First() should be called prior to this routine to start the iteration. Next can be called repeatedly to iterate through the remaining items in the list. IsNotComplete returns kXMPFalse if the preceeding call to Next did not return an item from the list.

Before calling this method, you must ensure that First() has been called to start the iteration. Next can be called repeatedly until IsNotComplete returns `kXMPTrue`.

**EXCEPTIONS**

`kXMPErrIteratorOutOfSync`
: The XMPTypeList instance was changed during the iteration.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPUndo

*Superclasses* `XMPObject`
*Subclasses* none

XMPUndo is used to store command history.

## Description

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.

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 from XMPArbitrator first. As soon as the part is done with the XMPUndo object, it should relinquish 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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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: XMPUndo must contain two stacks (an undo stack and a redo stack) at all times.

## Methods

`AddActionToHistory`

`ClearActionHistory`

`ClearRedoHistory`

`MarkActionHistory`

`PeekRedoHistory`

`PeekUndoHistory`

`Purge`

`Redo`

`Undo`

Detailed descriptions of these methods follow, in alphabetical order.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## AddActionToHistory

This function pushes actionData together with its associated part onto the Undo Stack.

```
void AddActionToHistory (XMPPart* whichPart,
                         XMPActionData actionData,
                         XMPActionType actionType,
                         XMPName* actionLabel);
```

`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".

### DESCRIPTION

Undo. Public. Parts can call, but shell apps must not call.

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

Before you call this method, Undo Stack must have been created. After this method executes successfully, whichPart and actionData are on the top of the Undo Stack.

### EXCEPTIONS

`kXMPErrCannotAddAction`
Currently already doing an undo or redo; trying to add a nested begin action.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`kXMPErrOutOfMemory`
Initialization of XMPUndo object failed; Couldn't allocate memory for action info.

## ClearActionHistory

This function clears the Stacks.

```
void ClearActionHistory
                    (XMPRespectMarksChoices respectMarks);
```

`respectMarks`
respect or don't respect marks.

**DESCRIPTION**

Undo. Public.

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

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.

Before you call this method, Undo Stack and Redo Stack must have been created. After this method executes successfully, Undo Stack and Redo Stack should be empty or cleared to the marks depending on respectMarks.

## ClearRedoHistory

This function clears the Redo history.

```
void ClearRedoHistory();
```


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Undo. Public.

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

Marks are always respected.

Before you call this method, Redo Stack must have been created. After this method executes successfully, Redo Stack should be empty.

## MarkActionHistory

This function places a mark on top of the Undo Stack and the Redo Stack.

```
void MarkActionHistory();
```

**DESCRIPTION**

Undo. Public.

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

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.

Before you call this method, Both Undo Stack and Redo Stack must have been created. After this method executes successfully, marks are on top of both Undo Stack and Redo Stack.

**EXCEPTIONS**

`kXMPErrCannotMarkAction`
XMPUndo was never initialized properly.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## PeekRedoHistory

This function returns information about the redoable action on the top of the Redo Stack.

```
XMPBoolean PeekRedoHistory (XMPPart** part,
                            XMPActionData* actionData,
                            XMPActionType* actionType,
                            XMPName** actionLabel);
```

`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 value* Returns whether there was anything on the stack.

### DESCRIPTION

Undo. Public.

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

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 respects marks.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## PeekUndoHistory

This function returns information about the undoable action on the top of the Undo Stack.

```
XMPBoolean PeekUndoHistory (XMPPart** part,
                            XMPActionData* actionData,
                            XMPActionType* actionType,
                            XMPName** actionLabel);
```

`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 value* Returns whether there was anything on the stack.

**DESCRIPTION**

Undo. Public.

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

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 respects marks.

## Purge

Frees up some memory, if possible.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `size` | The amount of memory requested |
| *return value* | The amount of memory actually freed. |

**DESCRIPTION**

None. Public.

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

After this method executes successfully, some memory may be freed.

## Redo

(For embedding application or document shell only.)
This function redoes the last undone action in the Redo history.

```
void Redo();
```

**DESCRIPTION**

Undo. Public. Usually the container is responsible for calling this.

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

Before you call this method, Both Undo Stack and Redo Stack must have been created. Redo Stack must not be empty. After this method executes successfully, what was on top of the Redo Stack is now on top of the Undo Stack.

**EXCEPTIONS**

| | |
|---|---|
| Others… | AppleEvent errors and any error returned from the part. |
| `kXMPErrEmptyStack` | Redo stack is empty; Undo object was never initialized. |


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Undo

(For embedding application or document shell only.)
This function undoes the last action in the Undo history.

```
void Undo();
```

**DESCRIPTION**

Undo. Public. Usually only called by the container app or document shell.

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

Before you call this method, both Undo Stack and Redo Stack must have been created. Undo Stack must not be empty. After this method executes successfully, what was on top of the Undo Stack is now on top of the Redo Stack.

**EXCEPTIONS**

Other… AppleEvent errors or any errors returned by the part.

kXMPErrEmptyStack
Undo stack is empty; Undo object was never initialized.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPValueIterator

*Superclasses* none
*Subclasses* none

This class is an iterator for use with the XMPNameSpace class.

## Description

It has no superclass. The platform vendor will implement.

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: It is only meant to be used in the context of a for statement. Any other use is not guaranteed to be correct.

## Methods

```
XMPValueIterator
~XMPValueIterator
First
IsNotComplete
Next
```

Detailed descriptions of these methods follow, in alphabetical order.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## XMPValueIterator

Initialize the private member variables.

```
XMPValueIterator (XMPNameSpace* nameSpace);
```

`nameSpace` The XMPNameSpace over which to iterate.

**DESCRIPTION**

Name Binding. Public.

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

Before you call this method, a valid XMPNameSpace must exist.

## ~XMPValueIterator

```
~XMPValueIterator();
```

**DESCRIPTION**

Name Binding. Public.

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

Deletes private storage.

## First

Return the first key/value pair.

```
void First  (XMPISOStr* key,
             XMPPtr* value);
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`key` a pointer to the key to be returned and returns the key placed in the storage pointed to by key.

`value` a pointer to the value to be returned and returns the value placed in the storage pointed to by value.

**DESCRIPTION**

Name Binding. Public.

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

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.

**EXCEPTIONS**

`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.

## IsNotComplete

Return whether we have tried to go beyond the end of the XMPNameSpace entries.

```
XMPBoolean IsNotComplete();
```

*return value* `kXMPFalse` if we have tried to get an entry that was beyond the end of the table, `kXMPTrue` otherwise.

**DESCRIPTION**

Name Binding. Public.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

## Next

Return the next key/value pair.

```
void Next   (XMPISOStr* key,
             XMPPtr* value);
```

key — a pointer to the key to be returned, and returnsthe key placed in the storage pointed to by key.

value — a pointer to the value to be returned, and returns the value placed in the storage pointed to by value .

### DESCRIPTION

Name Binding. Public.

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

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.

Before you call this method, First must be have been called.

### EXCEPTIONS

`kXMPErrWrongType`
this is thrown if the type of the key does not match the type of the XMPNameSpace with which the iterator was initialized.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPWindow

*Superclasses* `XMPRefCntObject` → `XMPObject`

*Subclasses* none

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.

## Description

XMPWindow is a subclass of XMPRefCntObject.

Related classes are XMPWindowState and XMPWindowIterator, XMPFrame and XMPFacet.

XMPWindow is platform-specific.

Each XMPWindow contains a pointer to a platform-specific window structure. The facilities 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: 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`



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

kXMPPropWindowIsVisible

kXMPPropWindowHasCloseBox

kXMPPropWindowHasZoomBox

kXMPPropWindowIsResizable

kXMPPropWindowIsRootWindow

kXMPPropWindowIsFloating

kXMPPropWindowHasMailer

kXMPPropWindowIsMailerAware

kXMPPropWindowRefCon

kXMPPropRootFrame

kXMPPropSourceFrame

kXMPPropShouldShowLinks

## Methods

Close

CloseAndRemove

GetFacetUnderPoint

GetID

GetPlatformWindow

GetRootFrame

GetSourceFrame

Hide

IsActive

IsResizable

IsRootWindow

IsShown

Open

Purge

Resized



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

```
SetShouldSave
SetShouldShowLinks
ShouldSave
ShouldShowLinks
Show
```

Detailed descriptions of these methods follow, in alphabetical order.

## Close

(For embedding application or document shell only.)
Closes this window.

```
void Close();
```

**DESCRIPTION**

Display. Public. Called by the shell and window state. Parts typically call CloseAndRemove.

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

The window is hidden, removed from the window state, and the root frame is closed. The facet hierarchy is deleted.

Before you call this method, a valid initialized instance must exist. After this method executes successfully, the window is closed, and no longer usable.

## CloseAndRemove

Closes this window and removes the root frame from the draft.

```
void CloseAndRemove();
```


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Display. Public. Called by parts to remove auxiliary windows like palettes, which are not stored persistently.

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

The window is hidden, removed from the window state, and the root frame is removed from the draft. The facet hierarchy is deleted.

Before you call this method, a valid initialized instance must exist. After this method executes successfully, the window is closed, and no longer usable.

## GetFacetUnderPoint

Returns the facet under the given point.

```
XMPFrame* GetFacetUnderPoint (XMPPoint aPoint);
```

| | |
|---|---|
| `aPoint` | A point in window coordinates. |
| *return value* | The facet under the point. |

**DESCRIPTION**

Hit Detection. Public. Part Editors typically won't call this method.

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

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.

Before you call this method, a valid initialized instance must exist. After this method executes successfully, no effect on this instance.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetID

Returns the ID of the window.

```
XMPID GetID();
```

*return value* The ID of this window.

**DESCRIPTION**

Invariant Properties. Public.

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

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.

Before you call this method, a valid initialized instance must exist. After this method executes successfully, no effect on this instance.

## GetPlatformWindow

Returns the platform-specific window data structure

```
XMPPlatformWindow GetPlatformWindow();
```

*return value* The platform-specific window stored in fPlatformWindow.

**DESCRIPTION**

Invariant Properties. Public.

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

Before you call this method, a valid initialized instance must exist. After this method executes successfully, no effect on this instance.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetRootFrame

Returns the root frame of the window.

```
XMPFrame* GetRootFrame();
```

*return value* The root frame of the window.

**DESCRIPTION**

Invariant Properties. Public.

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

Before you call this method, a valid initialized instance must exist. After this method executes successfully, no effect on this instance.

## GetSourceFrame

Returns the frame from which this window was opened.

```
XMPFrame* GetSourceFrame();
```

*return value* The frame from which this window was opened, or `kXMPNull`

**DESCRIPTION**

Invariant Properties. Public.

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

Result may be `kXMPNull`.

Before you call this method, a valid initialized instance must exist. After this method executes successfully, no effect on this instance.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Hide

Makes the window invisible.

```
void Hide();
```

**DESCRIPTION**

Display. Public.

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

Before you call this method, a valid initialized instance must exist. After this method executes successfully, the window is hidden.

## IsActive

Returns `kXMPTrue` if this window is a floating window, or the frontmost non-floating window.

```
XMPBoolean IsActive();
```

*return value* `kXMPTrue`, if this is an active window.

**DESCRIPTION**

Layers. Public.

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

Before you call this method, a valid initialized instance must exist. After this method executes successfully, no effect on this instance.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## IsResizable

Returns `kXMPTrue` if the window is resizable by users, `kXMPFalse` otherwise.

```
XMPBoolean IsResizable();
```

*return value* `kXMPTrue` if the window is resizable, `kXMPFalse` otherwise

**DESCRIPTION**

Invariant Properties. Public.

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

Before you call this method, a valid initialized instance must exist. After this method executes successfully, no effect on this instance.

## IsRootWindow

Returns `kXMPTrue` if the window is a root window, `kXMPFalse` otherwise.

```
XMPBoolean IsRootWindow();
```

*return value* `kXMPTrue` if the window is a root window, `kXMPFalse` otherwise

**DESCRIPTION**

Invariant Properties. Public.

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

Before you call this method, a valid initialized instance must exist. After this method executes successfully, no effect on this instance.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## IsShown

Returns `kXMPTrue` if the window is currently shown, `kXMPFalse` otherwise.

```
XMPBoolean IsShown();
```

*return value* `kXMPTrue` if the window is currently shown, `kXMPFalse` otherwise

**DESCRIPTION**

Display. Public.

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

Before you call this method, a valid initialized instance must exist. After this method executes successfully, no effect on this instance.

## Open

Creates a facet hierarchy in this window.

```
void Open();
```

**DESCRIPTION**

Display. Public.

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

Does not make the window visible or change window ordering. Show() and Select() must be called for those operations.

Before you call this method, a valid initialized instance must exist. After this method executes successfully, the window and facet structure is build, and the window can be shown using Show().



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Purge

(For embedding application or document shell only.)
Frees up some memory, by deleting cached values which can be recomputed or restored from persistent storage.

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

| | |
|---|---|
| `size` | The amount of memory requested. |
| *return value* | The amount of memory freed. |

**DESCRIPTION**

Low Memory. Public. Called by OpenDoc.

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

Before you call this method, a valid initialized instance must exist. After this method executes successfully, some memory has been freed up.

## Resized

Reshapes the root frame to match the window size.

```
void Resized();
```

**DESCRIPTION**

Display. Public.

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

Should be called by parts if they resize a window programmatically.

Before you call this method, a valid initialized instance must exist. After this method executes successfully, the root frame matches the window shape.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## SetShouldSave

Changes the value of the "shouldSave" property

```
void SetShouldSave (XMPBoolean shouldSave);
```

`shouldSave` `kXMPTrue` if the window should be saved in the draft.

**DESCRIPTION**

Reading and Writing. Public.

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

Before you call this method, a valid initialized instance. must exist. After this method executes successfully, fShouldSave is updated.

## SetShouldShowLinks

Changes the value of the "shouldShowLinks" property

```
void SetShouldShowLinks (XMPBoolean shouldShowLinks);
```

`shouldShowLinks`
`kXMPTrue` if links should be shown, `kXMPFalse` otherwise.

**DESCRIPTION**

Display. Public.

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

Before you call this method, a valid initialized instance must exist. After this method executes successfully, fShouldShowLinks is updated.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## ShouldSave

Returns `kXMPTrue` if the window is saved in the draft, `kXMPFalse` otherwise.

```
XMPBoolean ShouldSave();
```

*return value* `kXMPTrue` if the window is saved in the draft, `kXMPFalse` otherwise

**DESCRIPTION**

Reading and Writing. Public.

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

Before you call this method, A valid initialized instance must exist. After this method executes successfully, no effect on this instance.

## ShouldShowLinks

Returns `kXMPTrue` if the user has requested that links be highlighted in the window.

```
XMPBoolean ShouldShowLinks();
```

*return value* `kXMPTrue` if links should be shown, `kXMPFalse` otherwise

**DESCRIPTION**

Display. Public.

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

Before you call this method, a valid initialized instance must exist. After this method executes successfully, no effect on this instance.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Show

Makes the window visible. Does not change the ordering of windows.

```
void Show();
```

**DESCRIPTION**

`kXMPTrue` if links should be shown, `kXMPFalse` otherwise. Public.

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

Before you call this method, a valid initialized instance must exist. After this method executes successfully, the window is visible.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPWindowIterator

*Superclasses* none

*Subclasses* none

XMPWindowIterator is a companion class of XMPWindowState , used to iterate over the windows in the window state.

## Description

XMPWindowIterator has no base class.

Related classes are XMPWindowState and XMPWindow.

XMPWindowIterator is platform-independent.

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: XMPWindowIterator maintains a reference to the window state, and to the current window (or to a lower-level iterator).

## Methods

`~XMPWindowIterator`

`First`

`IsNotComplete`

`Last`

`Next`



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Previous

Detailed descriptions of these methods follow, in alphabetical order.

## ~XMPWindowIterator

Frees the memory allocated by this class.

```
~XMPWindowIterator();
```

**DESCRIPTION**

Creation and Deletion. Public.

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

Before you call this method, a constructed instance must exist. After this method executes successfully, the memory used by this instance is freed.

## First

Returns the first window in the window state.

```
XMPWindow* First();
```

*return value* The first window in the window state

**DESCRIPTION**

Begin Iteration. Public.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Advance iteration with Next(). Not related to front-to-back ordering of windows.

Before you call this method, a valid initialized instance must exist. After this method executes successfully, current window is advanced.

## IsNotComplete

Returns `kXMPTrue` if there are more windows to iterate over, `kXMPFalse` otherwise.

```
XMPBoolean IsNotComplete();
```

*return value* `kXMPTrue`, if there are more windows to iterate over.

### DESCRIPTION

Completion Test. Public.

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

Before you call this method, a valid initialized instance must exist. First() or Last() must have been called. After this method executes successfully, Result contains `kXMPTrue` if there are more windows to iterate over, `kXMPFalse` otherwise.

## Last

Returns the last window in the window state.

```
XMPWindow* Last();
```

*return value* The last window in the window state



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Begin Iteration. Public.

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

Advance iteration with Previous().

Before you call this method, a valid initialized instance must exist. After this method executes successfully, Current window is advanced.

## Next

Returns the next window in the window state.

```
XMPWindow* Next();
```

*return value* The next window in the window state

**DESCRIPTION**

Advance Iteration. Public.

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

Begin iteration with First().

Before you call this method, First() must have been called. After this method executes successfully, Current window is advanced.

## Previous

```
XMPWindow* Previous();
```

*return value* The previous window in the window state


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Advance Iteration. Public.

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

Returns the previous window in the window state. Begin iteration with Last().

Before you call this method, a valid initialized instance must exist. After this method executes successfully, Current window is advanced.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPWindowState

*Superclasses* `XMPObject`

*Subclasses* none

XMPWindowState maintains a list of windows, and some menu bar information.

## Description

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.

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: XMPWindowState contains a list of XMPWindow objects.

XMPWindowState also contains the base menu bar object.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

### Other Persistent Properties

Although XMPWindow is not an XMPPersistentObject, the WindowState stored windows persistently in a special draft property.

## Methods

```
AdjustPartMenus
CloseWindows
CopyBaseMenuBar
CreateWindowIterator
Externalize
GetActiveWindow
GetRootWindowCount
GetTotalRootWindowCount
GetWindow
GetWindowCount
GetXMPWindow
Internalize
IsXMPWindow
OpenWindows
Purge
SetBaseMenuBar
SetDefaultWindowTitles
```

Detailed descriptions of these methods follow, in alphabetical order.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## AdjustPartMenus

Calls Part::AdjustMenus for the part with the menu focus, so that it can enable/disable menu items as necessary.

```
void AdjustPartMenus();
```

**DESCRIPTION**

Menu Bar. Public. Called when the user clicks in the menu bar.

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

Before you call this method, a valid initialized instance must exist. After this method executes successfully, menus are set up by part with menu focus.

## CloseWindows

(For embedding application or document shell only.)
Closes all windows belonging to the specified draft.

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

`draft` An open OpenDoc draft

**DESCRIPTION**

Reading and Writing. Public. Called by the OpenDoc shell when closing a draft.

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

Before you call this method, A valid open draft object must exist. After this method executes successfully, All windows of the specified draft are closed.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**EXCEPTIONS**

`kXMPErrInvalidDraft`
The specified draft is not valid.

## CopyBaseMenuBar

Returns a copy of the base menu bar.

```
XMPMenuBar* CopyBaseMenuBar();
```

*return value* Copy of base menu bar

**DESCRIPTION**

Menu Bar. Public. Called by Part Editors.

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

Called by part editors to create a menu bar to which they add their own menus.

Before you call this method, A valid initialized instance must exist. After this method executes successfully, No effect on this instance.

## CreateWindowIterator

Returns an iterator for all the windows (of all drafts) in the window state.

```
XMPWindowIterator* CreateWindowIterator();
```

*return value* The iterator

**DESCRIPTION**

Window Access. Public.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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

Before you call this method, A valid initialized instance must exist. After this method executes successfully, No effect on this instance.

## Externalize

(For embedding application or document shell only.)
Saves all windows belonging to the specified draft in the draft.

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

`draft` An open OpenDoc draft

#### DESCRIPTION

Reading and Writing. Public. Called by the OpenDoc shell when saving a draft.

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

Before you call this method, A valid open draft object must exist. After this method executes successfully, The draft contains the IDs of the windows that were saved in the draft.

#### EXCEPTIONS

`kXMPErrInvalidDraft`
The specified draft is not valid.

## GetActiveWindow

Returns the front-most window which is not a “floating” window.

```
XMPWindow* GetActiveWindow();
```


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

*return value* The active window

**DESCRIPTION**

Window Access. Public. Used by the dispatcher.

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

Before you call this method, A valid initialized instance must exist. After this method executes successfully, No effect on this instance.

## GetRootWindowCount

Returns the number of root windows belonging to the specified draft.

```
XMPUShort GetRootWindowCount (XMPDraft* draft);
```

*return value* The number of root windows belonging to the specified draft.

**DESCRIPTION**

Counting Windows. Public. Used by the document shell when closing a window, to determine when to close the draft.

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

Before you call this method, A valid initialized instance must exist. After this method executes successfully, No effect on this instance.

## GetTotalRootWindowCount

Returns the number of root windows of all drafts.

```
XMPUShort GetTotalRootWindowCount();
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

*return value* The number of root windows of all drafts

**DESCRIPTION**

Counting Windows. Public.

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

Before you call this method, A valid initialized instance must exist. After this method executes successfully, No effect on this instance.

## GetWindow

Returns the window object with the given ID.

```
XMPWindow* GetWindow (XMPID id);
```

`id` A window ID, obtained from a window using XMPWindow::GetID.

*return value* The window matching the specified ID, or `kXMPNull`.

**DESCRIPTION**

Window Access. Public.

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

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.

Before you call this method, A valid initialized instance must exist. After this method executes successfully, No effect on this instance.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## GetWindowCount

Returns the number of windows in the window state (regardless of draft)

```
XMPULong GetWindowCount();
```

*return value* The number of OpenDoc windows in the WindowState.

**DESCRIPTION**

Counting Windows. Public.

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

Before you call this method, A valid initialized instance must exist. After this method executes successfully, No effect on this instance.

## GetXMPWindow

Returns the XMPWindow object corresponding to the specified platform-specific window object.

```
XMPWindow* GetXMPWindow (XMPPlatformWindow window);
```

`window` A platform-specific window object

*return value* The corresponding OpenDoc window, or `kXMPNull` if the specified window is not an OpenDoc window

**DESCRIPTION**

Window Access. Public. Called mostly by the dispatcher.

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

Returns the XMPWindow object corresponding to the specified platform-specific window object, or `kXMPNull`.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Before you call this method, A valid initialized instance must exist. After this method executes successfully, No effect on this instance.

## Internalize

(For embedding application or document shell only.)
Adds to the window state all windows persistently stored in the specified draft.

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

`draft` An open OpenDoc draft.

**DESCRIPTION**

Reading and Writing. Public. Called by the OpenDoc shell when a draft is opened.

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

Before you call this method, A valid open draft object must exist. After this method executes successfully, The window state contains all the windows that were saved in the draft.

**EXCEPTIONS**

`kXMPErrInvalidDraft`
The specified draft is not valid.

`kXMPErrOutOfMemory`
Out of memory


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## IsXMPWindow

Returns true if the specified platform-specific window is an OpenDoc window.

```
XMPBoolean IsXMPWindow (XMPPlatformWindow window);
```

`window` A platform-specific window object

*return value* `kXMPTrue`, if the window is an OpenDoc window

### DESCRIPTION

Window Access. Public. Called mostly by the dispatcher.

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

Before you call this method, A valid initialized instance must exist. After this method executes successfully, No effect on this instance.

## OpenWindows

(For embedding application or document shell only.)
Opens all windows stored in the specified draft, and shows those that were visible when saved.

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

`draft` An open OpenDoc draft.

### DESCRIPTION

Reading and Writing. Public. Called by the OpenDoc shell.

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

Before you call this method, A valid open draft object must exist. The WindowState must have already Internalized the windows of the specified



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

draft. After this method executes successfully, The windows of the specified draft are open.

**EXCEPTIONS**

`kXMPErrInvalidDraft`
The specified draft is not valid.

`kXMPErrOutOfMemory`
Out of memory

## Purge

(For embedding application or document shell only.)
Frees up some memory, by deleting cached values which can be recomputed or restored from persistent storage.

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

| | |
|---|---|
| `size` | The amount of memory requested. |
| *return value* | The amount of memory freed. |

**DESCRIPTION**

Low Memory. Public. Called by OpenDoc.

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

Before you call this method, A valid initialized instance must exist. After this method executes successfully, Some memory has been freed up.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## SetBaseMenuBar

(For embedding application or document shell only.)
Stores the base menu bar created by the shell.

```
void SetBaseMenuBar (XMPMenuBar* menuBar);
```

`menuBar` Reference to base menu bar object

**DESCRIPTION**

Menu Bar. Public. Called by the OpenDoc shell application

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

Parts call CopyBaseMenuBar to create a menu bar to which they add their own menus.

Before you call this method, A valid initialized instance must exist. After this method executes successfully, Field is set.

## SetDefaultWindowTitles

(For embedding application or document shell only.)
Synchronizes window titles with the file name.

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

`draft` An open OpenDoc draft

**DESCRIPTION**

Reading and Writing. Public.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Before you call this method, A valid open draft object must exist. After this method executes successfully, The window titles are synchronized with the file name.

**EXCEPTIONS**

`kXMPErrInvalidDraft`
The specified draft is not valid.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

CHAPTER 3

# OpenDoc Enhancement Classes

## Contents


XMPExtension 568
- XMPExtension 569
- ~XMPExtension 569
- GetBase 570
- InitExtension 570
- Release 571

XMPDispatchModule 572
- XMPDispatchModule 573
- ~XMPDispatchModule 573
- Dispatch 574
- InitDispatchModule 574

XMPFocusModule 576
- XMPFocusModule 577
- ~XMPFocusModule 577
- CreateOwnerIterator 578
- GetFocusOwner 578
- InitFocusModule 579
- IsFocusExclusive 579
- SetFocusOwnership 580
- UnsetFocusOwnership 581





Draft. Preliminary. © Apple Computer, Inc. 6/20/94
Tucson 1.0–Chapter TOC Seq

CHAPTER 3



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# OpenDoc Enhancement Classes

This chapter describes the classes that OpenDoc provides specifically for extending its features.

- The class `XMPExtension` is the abstract superclass from which extensions to OpenDoc are defined; the class `XMPSemanticInterface` (described in Chapter 3 of this book) is an example of a subclass of `XMPExtension`.
- The class `XMPDispatchModule` is not abstract; OpenDoc uses instances of that class to dispatch certain types of events (such as keystroke events) to part handlers. You can, however, provide for dispatching of new types of events or messages to your part handler by subclassing `XMPDispatchModule`.
- The class `XMPFocusModule` is not abstract; OpenDoc uses instances of that class to describe a particular type of focus (such as the selection focus). You can, however, provide new types of focus for your part handler by subclassing `XMPFocusModule`.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPExtension

*Superclasses* none

*Subclasses* `XMPSemanticInterface`

XMPExtension is the abstract base class from which object extension interfaces are derived.

## Description

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.

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: fBase contains a valid base object for this extension.

## Methods

`XMPExtension`

`~XMPExtension`

`GetBase`

`InitExtension`

`Release`



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Detailed descriptions of these methods follow, in alphabetical order.

## XMPExtension

Construct a new extension instance.

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

`base` The base object for this extension.

**DESCRIPTION**

Extensions

Protected.

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

After this method executes successfully, fBase holds base object for this extension.

## ~XMPExtension

Deallocate this instance and its storage.

```
~XMPExtension();
```

**DESCRIPTION**

Extensions

Protected.

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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Before calling this method, you must ensure that this instance must have been previously "released". After this method executes successfully, this instance is no longer usable by clients.

## GetBase

Return the object of which this instance is an extension.

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

*return value* The base of this extension.

**DESCRIPTION**

Extensions. Public.

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

## InitExtension

Prepare this object for use.

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

**DESCRIPTION**

Extensions

Protected.

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

Before calling this method, you must ensure that fBase is a valid base object. After this method executes successfully, this object is ready for use by clients.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Release

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

```
void Release();
```

### DESCRIPTION

Extensions. Public.

Derived class **must** override. Derived class **cannot** call base class behavior.

Before calling this method, you must ensure that fBase is a valid base object. After this method executes successfully, this extension is no longer usable.

### EXCEPTIONS

`kXMPErrInvalidBase`
Not a valid base object.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPDispatchModule

*Superclasses* none
*Subclasses* none

XMPDispatchModule is an abstract class. The dispatcher can be extended by adding subclasses of XMPDispatchModule.

## Description

Related classes are XMPDispatcher and XMPStandardDispatchModule, which is private to part developers.

XMPDispatchModule has no base class.

See XMPDispatcher

Invariants: The field fSession points to the global system interface.

## Methods

```
XMPDispatchModule
~XMPDispatchModule
Dispatch
InitDispatchModule
```

Detailed descriptions of these methods follow, in alphabetical order.


Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## XMPDispatchModule

Constructs a dispatch module.

```
XMPDispatchModule();
```

**DESCRIPTION**

Creation and Deletion

Public.

Derived class **cannot** override. Derived class **cannot** call base class behavior.

InitDispatchModule must be called.

After this method executes successfully, a constructed dispatch module exists. Not yet usable - InitDispatchModule must be called.

## ~XMPDispatchModule

Frees the memory allocated by this object

```
~XMPDispatchModule();
```

**DESCRIPTION**

Creation and Deletion

Public.

Derived class **cannot** override. Derived class **cannot** call base class behavior.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## Dispatch

Classifies the event and dispatches it to the appropriate part.

```
XMPBoolean Dispatch(
XMPEventData event);
```

| | |
|---|---|
| `event` | A platform-specific event data structure. |
| *return value* | `kXMPTrue` if the event was handled by a part, `kXMPFalse` otherwise. |

**DESCRIPTION**

Event Handling

Public. Called by the dispatcher after it has located this module in its table.

Derived class **must** override. Derived class **can** call base class behavior.

Before calling this method, you must ensure that a valid event exists. After this method executes successfully, the event is dispatched.

## InitDispatchModule

Saves the session reference.

```
void InitDispatchModule(
XMPSession* session);
```

`session`

**DESCRIPTION**

Creation and Deletion

Public. Called only by concrete derived class's Init method.

Derived class **cannot** override. Derived class **cannot** call base class behavior.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

After this method executes successfully, the instance is usable.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# XMPFocusModule

*Superclasses* none

*Subclasses* none

The `XMPFocusModule` class is used to store the owner frame for one or more foci.

## Description

`XMPFocusModule` is the abstract superclass from which focus modules are instantiated. This class has no superclass. It is used to register newly defined foci. This is done by calling the `RegisterFocus` method maintained by `XMPArbitrator`. This method takes `focusModule` as a parameter. It may be called several times to register successive foci within one or more focus modules. Several foci can be stored within one instance of `XMPFocusModule` and are then referenced by the `XMPArbitrator` as sharing a common owner frame.

For more information, see `XMPArbitrator`, `XMPFocusSet` and `XMPFocusSetIterator`.

## Methods

XMPFocusModule is used primarily for determining the owner frames for specified foci. Its methods provide for the creation and deletion of focus modules, and for testing the exclusivity of foci.

### Creation and Deletion

| | |
|---|---|
| `XMPFocusModule` | Constructs a focus module. |
| `~XMPFocusModule` | Frees the memory that is allocated by this object. |
| `InitFocusModule` | Initializes the focus module. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**Focus Ownership**

| | |
|---|---|
| `CreateOwnerIterator` | Returns a focus-owner iterator that iterates over the parts that own the specified non-exclusive focus. |
| `GetFocusOwner` | Returns the frame that owns the specified focus. |
| `SetFocusOwnership` | Stores the frame as the owner of the specified focus. |
| `UnsetFocusOwnership` | `Stores kXMPNull` as the owner of the specified focus. |

**Focus Testing**

| | |
|---|---|
| `IsFocusExclusive` | States whether the specified focus is exclusive. |

Detailed descriptions of these methods follow, in alphabetical order.

## XMPFocusModule

The `XMPFocusModule` method constructs a focus module.

```
XMPFocusModule();
```

**DESCRIPTION**

After this method executes successfully, you must ensure that though the object is constructed, it is not ready for use until the `InitFocusModule` method is called.

## ~XMPFocusModule

The `~XMPFocusModule` method frees the memory that is allocated by this object.

```
~XMPFocusModule();
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Before calling this method, you must ensure that the object is a constructed and initialized object of this class. After it executes successfully, the memory used by this object is freed.

## CreateOwnerIterator

The `CreateOwnerIterator` method returns a focus-owner iterator that iterates over the parts that own the specified non-exclusive focus.

```
XMPFocusOwnerIterator* CreateOwnerIterator(
XMPTypeToken focus);
```

| | |
|---|---|
| `focus` | A tokenized string of type `XMPTypeToken`, representing a registered focus type. |
| *return value* | An iterator that specifies the current owner of the specified focus. |

**DESCRIPTION**

Called by the arbitrator object. Part editors call `XMPArbitrator` methods.

The caller must delete the iterator

## GetFocusOwner

The `GetFocusOwner` method returns the frame that owns the specified focus.

```
XMPFrame* GetFocusOwner(
XMPTypeToken focusType);
```

| | |
|---|---|
| `focusType` | A tokenized string of type `XMPTypeToken`, representing a registered focus type. |
| *return value* | The frame that owns the focus. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

Called by the arbitrator object. Part editors call `XMPArbitrator::GetFocusOwner`. You must override this method.

Before calling this method, you must ensure that the specified focus is a constructed and initialized object of this class.

## InitFocusModule

The `InitFocusModule` method initializes the focus module.

```
void InitFocusModule(
XMPSession* session);
```

`session` The OpenDoc session object.

**DESCRIPTION**

Before calling this method, you must ensure that the object is a constructed and initialized object of this class. After it executes successfully, it returns an initialized and usable instance.

## IsFocusExclusive

The `IsFocusExclusive` method states whether the specified focus is exclusive.

```
XMPBoolean IsFocusExclusive(
XMPTypeToken focus);
```

`focus` A tokenized string of type `XMPTypeToken` representing a registered focus type.

*return value* This method returns a Boolean: `kXMPTrue` if the specified focus is exclusive, and `kXMPFalse` otherwise.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

**DESCRIPTION**

This method is called by instances of `XMPArbitrator`. It can be called by instances of `XMPPart`.

Before calling this method, you must ensure that the object is a constructed and initialized object of this class.

## SetFocusOwnership

The `SetFocusOwnership` method stores the frame as the owner of the specified focus.

```
XMPFocusOwnerIterator* CreateOwnerIterator(
XMPTypeToken focus
XMPFrame* frame);
```

| | |
|---|---|
| `focus` | A tokenized string of type `XMPTypeToken`, representing a registered focus type. |
| `frame` | A reference to the new owner frame. |
| *return value* | |

**DESCRIPTION**

Called by the arbitrator object. Part editors call XMPArbitrator methods.

If this focus module manages more than one focus, a table must be maintained.

Before calling this method, you must ensure that the specified token represents a valid, non-exclusive focus, and that the frame is registered. After it executes successfully, ownership of the focus will be stored.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPErrInvalidFrame` | The specified frame is invalid. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## UnsetFocusOwnership

The `UnsetFocusOwnership` method stores `kXMPNull` as the owner of the specified focus.

```
XMPFrame* GetFocusOwner(
XMPTypeToken focus
XMPFrame* frame);
```

| | |
|---|---|
| `focus` | A tokenized string of type `XMPTypeToken`, representing a registered focus. |
| `frame` | A reference to the current owner of the specified focus. |
| *return value* | |

### DESCRIPTION

Called by the arbitrator object. Part editors call XMPArbitrator methods.

Before calling this method, you must ensure that the specified token represents a valid, non-exclusive focus, and that the frame is registered. After it executes successfully, ownership of the focus will be removed.

### EXCEPTIONS

| | |
|---|---|
| `kXMPErrInvalidFrame` | The specified frame is invalid. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Draft. Preliminary. © Apple Computer, Inc. 6/20/94

CHAPTER 4

# Global Functions, Types, and Constants

## Contents


Global Functions 586
- `OpenXMPSession` 586
- `ShowLinkDestinationInfo` 586
- `ShowLinkSourceInfo` 587
- `ShowPartFrameInfo` 588
- `ShowPasteAsDialog` 589

Types 591
- Data Interchange 591
- Storage System 593
- General 593
- Imaging 594
- Linking 598
- UI Subsystem 598
- Messaging 599

Constants 601
- Name Spaces 601
- General 601
- Category Constants 602
- Parts 602
- Frame Groups 602
- UI Subsystem 603
  - Foci 603
  - Menu IDs 603
  - OpenDoc Events 604
  - Command IDs 605
  - Persistent Properties 607




Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Persistent Value Types 608
Undo 608
Messaging subsystem 609
General 609
Properties for Part Info Dialog 609
Data Interchange 610
General 610
Storage System 611
Persistent Properties 612
Persistent Value Types 613



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# Global Functions, Types, and Constants

This chapter describes three sets of items:

- The global functions provided by OpenDoc. These few functions are available from the OpenDoc environment itself, and are not methods of any object class.
- The data types defined by OpenDoc. The types are grouped into functional categories.
- The constants defined by OpenDoc. The constants are grouped into functional categories.

**Note**
All information described in this chapter is platform-independent. For descriptions of any platform-specific OpenDoc global functions, types, and constants, see the *OpenDoc Cookbook* for the platform in question. [For the Macintosh platform, this information is currently in a separate document entitled *OpenDoc Class Reference: Macintosh Supplement*..] ◆



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# Global Functions

## OpenXMPSession

(For embedding application or document shell only.)
Creates new `XMPSession` object; returns reference.

```
XMPSession* OpenXMPSession ();
```

*return value* A reference to the `XMPSession` object created by this call.

**DESCRIPTION**

System. Public.

This should only be called once per process. After this function executes successfully, The OpenDoc environment will be initialized.

**EXCEPTIONS**

| | |
|---|---|
| Various exceptions | OpenDoc could not be initialized. |

## ShowLinkDestinationInfo

Show the Destination Link Info dialog and return the user's choices.

```
XMPBoolean ShowLinkDestinationInfo(
   XMPLink* link,
   XMPLinkInfo* info,
   XMPLinkInfoResult* result);
```



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `link` | the XMPLink object to show info for. |
| `info` | a read-only structure providing info about the content at the destination of the link. |
| `result` | the settings and action chosen by the user (see the documentation on Data Interchange Types). |
| *return value* | kXMPFalse if the user pressed Cancel, and kXMPTrue if any other action was requested. |

**DESCRIPTION**

Linking. Public.

If `kXMPTrue` is returned, the caller should inspect the 'result' structure and perform the action requested by the user.

Before calling this method, you must ensure that 'link' is a valid XMPLink object, 'info' is a structure created and initialized by the caller, and 'result' is a structure created by the caller.After executing this method successfully, 'result' has been updated to reflect what the user chose in the dialog.

**EXCEPTIONS**

| | |
|---|---|
| `kXMPOutOfMemory` | could not complete user interaction. |

## ShowLinkSourceInfo

Show the Source Link Info dialog and return the user's choices.

```
XMPBoolean ShowLinkSourceInfo(
   XMPLinkSource* linkSource,
   XMPChangeID change,
   XMPLinkInfoResult* result);
```

| | |
|---|---|
| `linkSource` | the XMPLinkSource object to show info for. |
| `change` | the change id of the source content, which may be more recent than the link's content if the link has not yet been updated. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `result` | the settings and action chosen by the user (see the documentation on Data Interchange Types). |
| *return value* | kXMPFalse if the user pressed Cancel, and kXMPTrue if any other action was requested. |

**DESCRIPTION**

Linking. Public.

If `kXMPTrue` is returned, the caller should inspect the 'result' structure and perform the action requested by the user.

Before calling this method, 'linkSource' is a valid XMPLinkSource object, and 'result' is a structure created by the caller. After executing this method successfully, 'result' has been updated to reflect what the user chose in the dialog.

**EXCEPTIONS**

`kXMPOutOfMemory`could not complete user interaction.

## ShowPartFrameInfo

Show the part/frame info dialog and permits users to change info on them.

```
XMPBoolean ShowPartFrameInfo(XMPFrame* frame);
```

| | |
|---|---|
| `frame` | A reference to the selected frame |
| *return value* | A Boolean specifying whether or not the user clicked OK. |

**DESCRIPTION**

Part. Public.

Before you call this function, 'frame' must be an `XMPFrame` object selected in the content of the caller or an `XMPFrame` object of the caller itself. After this



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

function executes successfully, Info settings have been updated to reflect what the user chose in the dialog.

## ShowPasteAsDialog

Show the Paste As dialog and return the user's choices via the 'result' parameter.

```
XMPBoolean ShowPasteAsDialog(
   XMPBoolean canPasteLink,
   XMPBoolean canEmbed,
   XMPStorageUnit* contentSU,
   XMPFrame* activeFrame,
   XMPTypeToken viewType,
   XMPPasteAsResult* result);
```

`canPasteLink`
: `kXMPTrue` if the part allows a link to be pasted, and `kXMPFalse` otherwise.

`canEmbed`
: `kXMPTrue` if the part allows embedding, and `kXMPFalse` otherwise.

`contentSU`
: A refernce to the storage unit containing the content to paste.

`activeFrame`
: A reference to the context in which pasting is being performed

`viewType`
: the preferred view type for an embedded part; ignored if canEmbed is `kXMPFalse`.

`result`
: A reference to the settings chosen by the user (see the documentation on Data Interchange Types).

*return value*
: whether or not the user clicked OK

**DESCRIPTION**

Data Interchange. Public.

Before calling this method, you must ensure that 'contentSU' is an XMPStorageUnit containing a contents property with at least one value,



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

'activeFrame' is an XMPFrame object displaying the part calling this function, and 'result' is an object created by the caller. After executing this method successfully, 'result' has been updated to reflect what the user chose in the dialog.

**EXCEPTIONS**

`kXMPOutOfMemory`
could not complete interaction with the user.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# Types

## 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. |
| `XMPClipboardKey` | `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. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `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. |
| `XMPPasteAsResult` | Captures settings from the Paste As dialog. An instance of this class is an output parameter to the global function `ShowPasteAsDialog` (see the documentation for global functions). Since the data members are public, parts can access the fields directly, but should treat the fields as read-only data. Management of the two `XMPType` fields is handled automatically. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## 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. |

## 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. Multiplication or division are trickier. `XMPFixed` is identical to the Macintosh "Fixed" type, and the Macintosh has several Toolbox |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| | 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. |

## 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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

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. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| 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 |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| | 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. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `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. |

## Linking

These scalar types are used by classes providing linking in OpenDoc.

| | |
|---|---|
| `XMPLinkInfo` | Captures information about the content at the destination of a link. An instance of this structure is an input parameter to the global function `ShowLinkDestinationInfo`, which treats this structure as a read-only parameter (see the documentation for global functions). Parts must initialize all fields prior to calling `ShowLinkDestinationInfo`. |
| `XMPLinkInfoAction` | Enumerates all actions the user can request in a link info dialog. |
| `XMPLinkInfoResult` | Contains the setting and action requested by the use in a link info dialog. An instance of this structure is an output parameter to the global functions ShowLinkSourceInfo and ShowLinkDestinationInfo (see the documentation for global functions). Since this structure is set by ShowLinkDestinationInfo, parts should treat the fields as read-only data. |

## UI Subsystem

Types used in the interfaces of the UI subsystem.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `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. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# Constants

## Name Spaces

| | |
|---|---|
| `kNameMappings` | The resource type for NMAP resources. |
| `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. |
| `kXMPNSTypeSLong` | An indicator of a name space with keys that are of type signed long. |

## 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. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## 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. |

## Frame Groups

| | |
|---|---|
| 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. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# UI Subsystem

## Foci

Constants for the standard foci managed by the Arbitrator. These constants are ISO strings that 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. |

## 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`] |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## 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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## 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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`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.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`kXMPCommandViewAsWin` The command ID for the View As Window item in the Edit menu.

## 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`.



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

`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.

## 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`.

## 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



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `kXMPRedone` | Value passed to Part::DisposeActionState() if the action was on the Undo stack |

## Messaging subsystem

Constants of the Messaging subsystem

### General

| | |
|---|---|
| `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. |

### 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. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `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

These constants are used during data interchange.

### General

| | |
|---|---|
| `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. |
| `kXMPCloneDropCopy` | An `XMPCloneKind` value specifying *** |
| `kXMPCloneDropMove` | An `XMPCloneKind` value specifying *** |
| `kXMPCloneDuplicate` | An `XMPCloneKind` value specifying duplicate semantics directly from the source draft to into the destination draft. |
| `kXMPClonePaste` | An `XMPCloneKind` value specifying paste semantics into an intermediate 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. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `kXMPDropSucceed` | An `XMPDropResult` value signifying a successful synchronous drop. |
| `kXMPDropCopy` | An `XMPDropResult` value signifying a successful synchronous drop with copy semantics. |
| `kXMPDropFail` | An `XMPDropResult` value signifying an unsuccessful synchronous drop. |
| `kXMPDropMove` | An `XMPDropResult` value signifying a successful synchronous drop with move semantics. |
| `kXMPDropUnfinished` | An `XMPDropResult` value signifying an asynchronous drag was started. |
| `kXMPNoWait` | An `XMPULong` value specifying not to wait in acquiring a busy lock on the clipboard or on a link. |
| `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. |

## 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. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| 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. |

## 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. |
| 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. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `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. |

## 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. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

| | |
|---|---|
| `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. |



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

# Index

## Symbols


~XMPCanvas method 92
~XMPContainer method 110
~XMPDispatchModule method 573
~XMPDocument method 128
~XMPDraft method 146
~XMPDragAndDrop method 174
~XMPExtension method 569
~XMPFacetIterator method 210
~XMPFacet method 189
~XMPFocusModule method 577
~XMPFocusOwnerIterator method 214
~XMPFocusSetIterator method 224
~XMPFocusSet method 219
~XMPFrameFacetIterator method 252
~XMPLinkSpec method 281
~XMPMenuBar method 286
~XMPObject method 320
~XMPPart method 11
~XMPPersistentObject method 327
~XMPPlatformTypeListIterator method 341
~XMPPlatformTypeList method 333
~XMPRefCntObject method 345
~XMPSemanticInterface method 351
~XMPShape method 384
~XMPStorageUnitCursor method 446
~XMPStorageUnit method 410
~XMPStorageUnitRefIterator method 450
~XMPStorageUnitView method 456
~XMPTransform method 482
~XMPTranslation method 502
~XMPTypeListIterator method 518
~XMPTypeList method 510
~XMPValueIterator method 531
~XMPWindowIterator method 548


## A


AbortClone method 146
AbortRelinquishFocus method 11
ActiveBorderContainsPoint method 189
AddActionToHistory method 523
AddDispatchModule method 120
AddDisplayFrame method 12
AddLast method 334, 511
AddMenuBefore method 286
AddMenuLast method 287
Add method 220, 333, 511
AddMonitor method 121
AddProperty method 410, 456
AddValue method 411, 457
AdjustBorderShape method 13
AdjustMenus method 14
AdjustPartMenus method 554
AnyLinkImported method 265
AttachSourceFrame method 14


## B


BeginClone method 147
BeginRelinquishFocus method 15


## C


CallObjectAccessor method 300
CanvasChanged method 16
CanvasUpdated method 17
ChangeActiveShape method 190
ChangeClipShape method 190
ChangedFromPrev method 147




Draft. Preliminary. © Apple Computer, Inc. 6/20/94

ChangeExternalTransform method 191
ChangeFrameShape method 231
ChangeGeometry method 191
ChangeHighlight method 192
ChangeInternalTransform method 232
ChangeKind method 17
ChangeLinkStatus method 232
ChangePresentation method 233
ChangeUsedShape method 234
ChangeViewType method 234
ClearActionHistory method 524
ClearAllPromises method 412
Clear method 101, 174, 271
ClearRedoHistory method 524
CloneInto method 18, 412, 457
CloneTo method 235, 255, 271, 414, 458
CloseAndRemove method 536
CloseDisplayFrame method 19
Close method 111, 235, 370, 536
CloseWindows method 554
CollapseDrafts method 129
CommitRelinquishFocus method 19
Constants 601–614
ContainingPartPropertiesChanged method 20
Contains method 220, 335, 512
ContainsPoint method 192, 385
ContentChanged method 236, 272
ConvertTransform method 483
CopyBaseMenuBar method 555
CopyFrom method 386, 484
Copy method 288, 385, 484
CopyPolygon method 386
CopyTo method 415, 459
Count method 335, 513
CountProperties method 415
CountValues method 416
CreateContainer method 401
CreateCursor method 416
CreateDraft method 130
CreateEmbeddedFacet method 192
CreateEmbeddedFramesIterator method 21
CreateEvent method 293
CreateFacetIterator method 194, 236
CreateFrame method 148
CreateIterator method 221
CreateLink method 22
CreateLinkSource method 149
CreateLinkSpec method 150
Create method 111
CreateNameSpace method 316
CreateOwnerIterator method 79, 578
CreatePartAddrDesc method 294
CreatePart method 151
CreatePartObjSpec method 295
CreatePlatformTypeListIterator method 336
CreateStorageUnit method 151
CreateSwapToken method 301
CreateToken method 302
CreateTypeListIterator method 513
CreateView method 417
CreateWindowIterator method 555

## D

DeleteNameSpace method 317
DeleteValue method 418, 460
Difference method 336
Dispatch method 122, 574
Display method 288
DisposeActionState method 23
DisposeToken method 302
DoesPropagateEvents method 237
DraftClosing method 265
DraftOpened method 266
DraftSaved method 266
DragEnter method 24
DragLeave method 24
DragWithin method 25
DrawActiveBorder method 195, 237
DrawChildrenAlways method 195
DrawChildren method 195
Draw method 26, 194
DrawnIn method 196
DropCompleted method 27



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

Drop method 27

## E

EmbeddedFrameChanged method 28
EmbeddedFrameSpec method 29
EndClone method 152
Exists method 308, 371, 419, 420
ExportClipboard method 102
ExternalizeKinds method 30
Externalize method 153, 237, 256, 273, 327, 420, 460, 556

## F

FacetAdded method 30, 238
FacetRemoved method 31, 233, 238
First method 179, 182, 211, 215, 224, 252, 341, 451, 518, 531, 548
FocusAcquired method 31
FocusLost method 32
Focus method 421, 422
FrameShapeChanged method 33
FulfillPromise method 33

## G

GeometryChanged method 34
GetActiveShape method 196
GetActiveWindow method 556
GetAggregateClipShape method 197
GetArbitrator method 371
GetBaseDraft method 131
GetBase method 570
GetBoundingBox method 387
GetCanvas method 197
GetChangeID method 103, 257, 273
GetChangeTime method 257
GetClipboard method 372
GetClipShape method 197
GetCoercionHandler method 351
GetContainer method 132, 402
GetContainingFacet method 198
GetContainingFrame method 238, 303
GetContainingPartProperties method 35
GetContentStorageUnit method 103, 175, 258, 274
GetContentTransform method 198
GetContextInfo method 304
GetCursor method 446, 461
GetDispatcher method 372
GetDocument method 112, 153
GetDocumentProperties method 132
GetDraft method 134, 423
GetDraftProperties method 154
GetDragAndDrop method 373
GetEventHandler method 352
GetExtension method 321
GetExternalTransform method 199
GetFacet method 93
GetFacetUnderPoint method 537
GetFocusOwner method 80, 578
GetFrameGroup method 239
GetFrame method 154, 199
GetFrameShape method 239
GetFrameTransform method 199
GetGenerationNumber method 424, 461
GetGeometryMode method 388
GetGraphicsSystem method 93, 485
GetHighlight method 200
GetIDFromStorageUnitRef method 425, 462
GetID method 113, 135, 155, 328, 424, 462, 538
GetInternalTransform method 239
GetISOTypeFromPlatformType method 502
GetLinkManager method 373
GetLink method 155, 274
GetLinkSource method 156
GetLinkStatus method 240
GetMatrix method 485
GetMenu method 289
GetMessageInterface method 373
GetName method 113, 136, 157, 308, 425, 463



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

GetNameResolver method 374
GetNameSpaceManager method 374
GetObjectAccessor method 353
GetOffset method 426, 463, 486
GetOSLSupportFlags method 354
GetOwner method 94
GetPartInfo method 200, 240
GetPart method 157, 240
GetPermissions method 158
GetPlatformCanvas method 94
GetPlatformShape method 388
GetPlatformTransform method 486
GetPlatformTypeFromISOType method 503
GetPlatformWindow method 538
GetPreScaleOffset method 487
GetPresentation method 241
GetPrintResolution method 36
GetPromiseValue method 427, 464
GetProperty method 428, 465
GetRootFrame method 539
GetRootWindowCount method 557
GetScale method 488
GetSemanticInterface method 375
GetSequenceNumber method 241
GetSession method 403, 429
GetShellSemtInterface method 375
GetSize method 429, 465
GetSourceFrame method 539
GetSpecialHandler method 355
GetStatus method 258
GetStorageSystem method 114, 376
GetStorageUnit method 159, 328, 466
GetStorageUnitRefIterator method 430, 467
GetStrongStorageUnitRef method 430, 467
GetTotalRootWindowCount method 557
GetTranslateMethod method 503
GetTranslation method 376
GetTranslationOf method 504
GetType method 376, 431, 468, 488
GetUndo method 377
GetUpdateShape method 95
GetUsedShape method 241
GetValue method 309, 432, 468
GetViewType method 242
GetWeakStorageUnitRef method 433, 469
GetWindowAggregateClipShape method 201
GetWindowContentTransform method 201
GetWindowCount method 559
GetWindowFrameTransform method 201
GetWindow method 200, 558
GetWindowState method 377
GetXMPWindow method 559
Global functions 586–590

## H

HandleEvent method 36
HasCanvas method 202
HasExtension method 321
HasGeometry method 389
HasMatrix method 489
HasNameSpace method 317
Hide method 540
HighlightChanged method 37

## I

IncrementGenerationNumber method 433, 470
IncrementRefCount method 114, 136, 159, 346, 434
InitCanvas method 95
InitContainer method 115
InitDispatchModule method 574
InitDocument method 137
InitDraft method 160
InitDragAndDrop method 175
InitExtension method 570
InitFacet method 202
InitFocusModule method 579
InitFocusOwnerIterator method 215
InitFocusSet method 221
InitMenuBar method 289
InitObject method 322
InitPartFromStorage method 38



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

InitPart method 38
InitPersistentObjectFromStorage method 329
InitPersistentObject method 329
InitPlatformTypeList method 337
InitRefCntObject method 346
InitSemanticInterface method 355
InitStorageUnitCursor method 447
InitStorageUnit method 434
InitStorageUnitRefIterator method 451
InitStorageUnitView method 470
InitTranslation method 505
InitTypeList method 514, 515
InsertValue method 435, 471
InstallAdjustMarksProc method 356
InstallCoercionHandler method 357
InstallCompareProc method 358
InstallCountProc method 358
InstallDisposeTokenProc method 359
InstallEventHandler method 359
InstallGetErrDescProc method 360
InstallGetMarkTokenProc method 361
InstallMarkProc method 361
InstallObjectAccessor method 362
InstallSpecialHandler method 363
Internalize method 436, 472, 560
Intersect method 390
InvalidateActiveBorder method 203, 242
Invalidate method 96, 203, 242
InverseTransform method 391
Invert method 489
InvertPoint method 490
InvertShape method 491
IsActive method 540
IsAutoUpdate method 275
IsDragging method 243
IsDroppable method 243
IsDynamic method 96
IsEmpty method 391
IsFocusExclusive method 579
IsFocusRegistered method 81
IsFrozen method 243
IsNotComplete method 179, 182, 211, 216, 225, 253, 342, 452, 519, 532, 549
IsOffscreen method 97
IsOverlaid method 244
IsPromiseValue method 436, 472
IsRectangular method 392
IsResizable method 541
IsRoot method 244
IsRootWindow method 541
IsSameAs method 392, 491
IsSelected method 204
IsShown method 542
IsStrongStorageUnitRef method 437, 473
IsSubframe method 244
IsWeakStorageUnitRef method 437, 473
IsXMPWindow method 561

## L

Last method 549
LinkStatusChanged method 38
LinkUpdated method 39
Lock method 104, 259, 276, 438

## M

MarkActionHistory method 525
MouseEnter method 39
MouseLeave method 40
MouseWithin method 41
MoveBefore method 204
MoveBehind method 205
MoveBy method 492

## N

NeedSpace method 404
Next method 180, 182, 212, 216, 225, 253, 342, 452, 519, 533, 550



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## O

Open method 41, 115, 542
OpenWindows method 561
OpenXMPSession global function 586

## P

PeekRedoHistory method 526
PeekUndoHistory method 527
PostCompose method 492
PreCompose method 493
PresentationChanged method 42
Previous method 551
ProcessSemanticEvent method 296
Purge method 97, 116, 123, 137, 160, 176, 245, 290, 296, 305, 309, 318, 323, 378, 439, 474, 505, 527, 543, 562

## R

ReadActionState method 43
ReadFromFile method 310
ReadFrom method 493
ReadFromStorage method 310
ReadLinkSpec method 281
ReadPartInfo method 43
ReadShape method 393
RedoAction method 44
Redo method 528
RegisterDependent method 260
Register method 311
RegisterObject method 378
ReleaseContainer method 404
ReleaseDocument method 117
ReleaseDraft method 138
ReleaseExtension method 323
ReleaseFrame method 161
ReleaseLink method 162
Release method 117, 138, 161, 245, 290, 347, 439, 571
ReleasePart method 163
ReleaseStorageUnit method 163
RemoveChanges method 164
RemoveCoercionHandler method 364
RemoveDispatchModule method 123
RemoveDisplayFrame method 45
RemoveEmbeddedFrame method 45
RemoveEntry method 379
RemoveEventHandler method 365
RemoveFacet method 205
RemoveFrame method 165
RemoveFromDocument method 166
RemoveLink method 166
RemoveLinkSource method 167
RemoveMenu method 291
Remove method 222, 245, 338, 440, 475, 515
RemoveMonitor method 124
RemoveObjectAccessor method 366
RemovePart method 168
RemoveSpecialHandler method 367
RemoveStorageUnit method 169
RemoveStorageUnitRef method 440, 475
RequestEmbeddedFrame method 46
RequestFrameShape method 47, 246
Reset method 494
ResetUpdateShape method 98
Resized method 543
Resolve method 305
RevealFrame method 48
RevealLink method 49

## S

SaveToAPrevDraft method 139
SaveToAPrevious method 170
ScaleBy method 494
ScaleDownBy method 495
Send method 297
SequenceChanged method 49
SetAutoUpdate method 276



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

SetBaseDraftFromForeignDraft method 140
SetBaseMenuBar method 563
SetCanvas method 206
SetChangedFromPrev method 170
SetContainingFrame method 246
SetDefaultWindowTitles method 563
SetDragging method 247
SetDroppable method 247
SetFacet method 98
SetFocusOwnership method 580
SetFrameGroup method 247
SetFrozen method 248
SetGeometryMode method 394
SetMatrix method 495
SetName method 118, 141, 171, 441, 476
SetOffset method 441, 476, 496
SetOSLSupportFlags method 367
SetOwner method 99
SetPartInfo method 207, 248
SetPlatformClipboard method 105
SetPlatformShape method 395
SetPlatformTransform method 496
SetPolygon method 395
SetPresentation method 248
SetPromiseValue method 442, 477
SetPropagateEvents method 249
SetRectangle method 396
SetSelected method 207
SetSequenceNumber method 249
SetShouldSave method 544
SetShouldShowLinks method 544
SetSourcePart method 277
SetSubframe method 250
SetType method 443, 478
SetValue method 444, 478
SetViewType method 250
SetWindow method 207
ShouldSave method 545
ShouldShowLinks method 545
ShouldTerminateEventLoop method 124
ShowLinkDestinationInfo global function 586
ShowLinkSourceInfo global function 587
Show method 546
ShowPartFrameInfo global function 588
ShowPasteAsDialog global function 589
ShowSourceContent method 260
SkipChildren method 212
StartDrag method 176
Subtract method 397

## T

TerminateEventLoop method 125
Tokenize method 380
Transform method 397
TransformPoint method 497
TransformShape method 498
Translate method 506, 507
Types 591–600

## U

UndoAction method 50
Undo method 529
Union method 338, 398
UniqueChangeID method 380
Unlock method 106, 261, 277
UnregisterDependent method 262
Unregister method 312
UnregisterObject method 381
UnsavedExportedLinks method 267
UnsetFocusOwnership method 581
Update method 208
UsedShapeChanged method 51

## V

Validate method 99, 208, 250
ViewTypeChanged method 51



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

## W

WriteActionState method 52
WriteLinkSpec method 282
WritePartInfo method 53
WriteShape method 399
WriteToFile method 313
WriteTo method 498
WriteToStorage method 313

## X

XMPArbitrator class 76–89
XMPCanvas class 90–99
XMPCanvas method 92
XMPClipboard class 100–107
XMPContainer class 108–118
XMPContainer method 110
XMPDispatcher class 119–125
XMPDispatchModule class 572–575
XMPDispatchModule method 573
XMPDocument class 126–141
XMPDocument method 128
XMPDraft class 142–171
XMPDraft method 145
XMPDragAndDrop class 172–177
XMPDragAndDrop method 173
XMPDragItemIterator class 178–180
XMPEmbeddedFramesIterator class 181–183
XMPExtension class 568–571
XMPExtension method 569
XMPFacet class 184–208
XMPFacetIterator class 209–212
XMPFacetIterator method 210
XMPFacet method 189
XMPFocusModule class 576–581
XMPFocusModule method 577
XMPFocusOwnerIterator class 213–217
XMPFocusOwnerIterator method 214
XMPFocusSet class 218–222
XMPFocusSetIterator class 223–225
XMPFocusSet method 219
XMPFrame class 226–250
XMPFrameFacetIterator class 251–253
XMPFrameFacetIterator method 251
XMPLink class 254–262
XMPLinkManager class 263–267
XMPLinkSource class 268–278
XMPLinkSpec class 279–283
XMPLinkSpec method 280
XMPMenuBar class 284–291
XMPMenuBar method 285
XMPMessageInterface class 292–298
XMPNameResolver class 299–306
XMPNameSpace class 307–314
XMPNameSpaceManager class 315–318
XMPObject class 319–324
XMPObject method 320
XMPPart class 4–53
XMPPart method 10
XMPPersistentObject class 325–330
XMPPersistentObject method 326
XMPPlatformTypeList class 331–339
XMPPlatformTypeListIterator class 340–343
XMPPlatformTypeList method 332
XMPRefCntObject class 344–347
XMPRefCntObject method 345
XMPSemanticInterface class 348–368
XMPSemanticInterface method 350
XMPSession class 369–381
XMPShape class 382–399
XMPShape method 384
XMPStorageSystem class 400–405
XMPStorageUnit class 406–444
XMPStorageUnitCursor class 445–448
XMPStorageUnitCursor method 446
XMPStorageUnit method 409
XMPStorageUnitRefIterator class 449–452
XMPStorageUnitRefIterator method 450
XMPStorageUnitView class 453–479
XMPStorageUnitView method 455
XMPTransform class 480–499
XMPTransform method 482
XMPTranslation class 500–508
XMPTranslation method 501
XMPTypeList class 509–516



Draft. Preliminary. © Apple Computer, Inc. 6/20/94

XMPTypeListIterator class 517–520
XMPTypeList method 510
XMPUndo class 521–529
XMPValueIterator class 530–533
XMPValueIterator method 531
XMPWindow class 534–546
XMPWindowIterator class 547–551
XMPWindowState class 552–564



Draft. Preliminary. © Apple Computer, Inc. 6/20/94