



# **IDML File Format Specification**

---

© 2008 Adobe Systems Incorporated. All rights reserved.

If this guide is distributed with software that includes an end user agreement, this guide, as well as the software described in it, is furnished under license and may be used or copied only in accordance with the terms of such license. Except as permitted by any such license, no part of this guide may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior written permission of Adobe Systems Incorporated. Please note that the content in this guide is protected under copyright law even if it is not distributed with software that includes an end user license agreement.

The content of this guide is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Adobe Systems Incorporated. Adobe Systems Incorporated assumes no responsibility or liability for any errors or inaccuracies that may appear in the informational content contained in this guide.

Please remember that existing artwork or images that you may want to include in your project may be protected under copyright law. The unauthorized incorporation of such material into your new work could be a violation of the rights of the copyright owner. Please be sure to obtain any permission required from the copyright owner.

Any references to company names and company logos in sample material are for demonstration purposes only and are not intended to refer to any actual organization.

Adobe, the Adobe logo, Acrobat, PostScript and Reader are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Java and JavaScript are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Windows and OpenType are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

SVG is a trademark of the World Wide Web Consortium; marks of the W3C are registered and held by its host institutions MIT, INRIA and Keto.

All other trademarks are the property of their respective owners.

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

**Notice to U.S. Government End Users.** The Software and Documentation are "Commercial Items," as that term is defined at 48 C.F.R. §2.101, consisting of "Commercial Computer Software" and "Commercial Computer Software Documentation," as such terms are used in 48 C.F.R. §12.212 or 48 C.F.R. §227.7202, as applicable. Consistent with 48 C.F.R. §12.212 or 48 C.F.R. §§227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are being licensed to U.S. Government end users (a) only as Commercial Items and (b) with only those rights as are granted to all other end users pursuant to the terms and conditions herein. Unpublished-rights reserved under the copyright laws of the United States. Adobe Systems Incorporated, 345 Park Avenue, San Jose, CA 95110-2704, USA. For U.S. Government End Users, Adobe agrees to comply with all applicable equal opportunity laws including, if appropriate, the provisions of Executive Order 11246, as amended, Section 402 of the Vietnam Era Veterans Readjustment Assistance Act of 1974 (38 USC 4212), and Section 503 of the Rehabilitation Act of 1973, as amended, and the regulations at 41 CFR Parts 60-1 through 60-60, 60-250, and 60-741. The affirmative action clause and regulations contained in the preceding sentence shall be incorporated by reference.

Portions Copyright © 2006 by International Digital Publishing Forum™.

# Contents

Abstract.....	10
Introduction.....	10
Terminology.....	10
INX, IDML, and InDesign Scripting .....	11
Script (JavaScript): .....	12
IDML: .....	12
Dynamic Object Model.....	12
IDML and Third Party Data .....	12
Uses for IDML.....	13
IDML Design Goals.....	13
IDML Design Approaches .....	14
Separate Content for Efficient Processing .....	14
Maximize Compatibility with InDesign.....	15
Maximize Independence of XML Elements.....	15
Use Attributes Rather Than Elements .....	15
Self-Documenting .....	16
Not a Literal Representation of InDesign Data Structures .....	16
Performance.....	16
Support for Previous Versions .....	16
IDML Document Structure .....	17
IDML Package File Organization .....	17
MIMETYPE .....	18
designmap.xml.....	19
Master Spreads Folder and MasterSpreads.xml .....	19
Resources Folder .....	19
Graphic.xml .....	19
Fonts.xml .....	19
Styles.xml .....	19
Preferences.xml .....	19
Spreads Folder .....	20
Stories Folder .....	20
XML Folder.....	20
BackingStory.xml .....	20
Tags.xml .....	20
Mapping.xml .....	21
META-INF Folder .....	21
IDML Component Names.....	21
Default Name .....	21
User Defined Names.....	21
Reset Component Name on Import.....	21
Scripting Interface.....	22
C++ API.....	22

IDML Syntax .....	22
XML Conventions and Style .....	22
Names in IDML.....	22
Use of Empty Elements .....	22
Generating IDML Schema .....	23
Data Types in IDML.....	24
Scalar Data Types vs. Complex Data Types.....	24
Enumerations .....	25
Key String.....	26
Measurement Units .....	26
Representation of Objects.....	27
Representation of Properties .....	27
Properties Represented as Attributes .....	27
Properties Represented as Elements .....	28
RecordType .....	29
Geometry .....	29
VariableType.....	29
Complex Structures.....	30
Use of the Type Attribute in Property Elements.....	30
Object Reference Format.....	30
Cross-file Reference Examples .....	32
 IDML File Reference.....	37
Common Attributes and Elements .....	37
Self .....	37
Scripting Labels .....	37
Optional Values, Defaults, and Preferences .....	38
designmap.xml .....	41
Documents and Color Management .....	47
Language .....	47
Condition .....	48
ConditionalTextPreference .....	49
TextVariable.....	50
Layer.....	54
Section .....	55
CrossReferenceFormat .....	56
Hyperlinks .....	58
HyperlinkPageDestination .....	59
HyperlinkURLDestination .....	60
HyperlinkExternalPageDestination.....	60
HyperlinkPageItemSource .....	61
Bookmark.....	62
Spreads and Master Spreads.....	63
Minimal Spread Example.....	65
Page Items .....	65
Nested Objects and IDML Structure.....	72
Geometry in IDML.....	73
Coordinates, Transformation Matrices, and the IDML Element Hierarchy .....	73
Pasteboard Coordinates .....	73
Transformations .....	74
Spread Coordinates.....	76
Page Item Geometry.....	77
Geometry Example.....	78
Page Item Examples .....	82

Nested Page Item.....	83
Rotated Page Item.....	84
Transformations and Nested Page Items.....	85
Page Item Containing an Imported Graphic .....	87
Text Frames .....	90
TextFramePreference .....	94
BaselineFrameGridOptions .....	97
Transparency .....	99
Group.....	100
Buttons .....	102
Behaviors .....	106
Associating Page Items with XML Elements.....	109
Pages.....	110
Columns and Margins .....	112
Guides.....	112
Transparency Flattener Settings .....	113
Stories.....	115
Story vs. XMLStory .....	115
Story vs. AssignedStory.....	116
Referring to a Story.....	116
Adding a Story .....	116
Local Formatting vs. Styles .....	116
Common Text Properties .....	118
Attributes and Elements Related to Japanese Features.....	136
Story Schema .....	137
Text Child Elements .....	147
Text Range Elements .....	147
ParagraphStyleRange Attributes .....	156
ParagraphStyleRange Elements .....	156
Character Style Range .....	156
CharacterStyleRange Attributes.....	164
CharacterStyleRange Elements.....	164
XML Elements in Text.....	168
Rules for Breaking Text Range Elements.....	170
Inline Elements.....	170
Tables.....	170
Cell .....	183
Notes .....	197
Anchored Frames.....	199
Anchored Graphics.....	202
Hyperlink Text Sources.....	204
HyperlinkTextDestination.....	205
Fonts .....	207
Font and Font Family .....	207
Referring to a Font.....	208
Composite Font .....	209
Graphics.....	211
Colors and Swatches .....	211
Swatch .....	212
Color.....	213
Gradient .....	216
GradientStop.....	217
Tint .....	217
Mixed Ink.....	219

MixedInkGroup.....	221
Ink .....	222
Stroke Styles .....	224
Preferences .....	229
DataMerge.....	230
DataMergeOption.....	231
LayoutAdjustmentPreference.....	231
XMLImportPreference.....	232
XMLExportPreference .....	234
XMLPreference.....	235
ExportForWebPreference.....	237
TransparencyPreference .....	238
TransparencyDefaultContainerObject.....	238
StoryPreference .....	239
TextFramePreference.....	239
TextPreference .....	242
TextDefault .....	244
DictionaryPreference.....	252
AnchoredObjectDefault .....	252
AnchoredObjectSetting .....	253
BaselineFrameGridOption .....	254
FootnoteOption.....	255
TextWrapPreference.....	261
DocumentPreference.....	262
GridPreference .....	264
GuidePreference .....	266
MarginPreference .....	267
PasteboardPreference.....	267
ViewPreference .....	268
PrintPreference.....	270
PrintBookletOption.....	278
PrintBookletPrintPreference .....	279
MetadataPacketPreference .....	281
IndexOptions .....	281
IndexHeaderSetting .....	282
PageItemDefault .....	282
FrameFittingOption .....	283
ButtonPreference .....	284
TinDocumentDDDataObject .....	284
LayoutGridDataInformation .....	284
StoryGridDataInformation .....	285
CjkGridPreference .....	285
MojikumiUiPreference .....	286
ChapterNumberPreference .....	287
Styles.....	288
Style Paths and the Self Attribute .....	288
ParagraphStyles.....	288
Nested Styles .....	299
Character Styles.....	300
TableStyles.....	306
CellStyles .....	317
Object Styles.....	319
TOCStyles.....	323

TrapPresets .....	327
XML Elements .....	330
BackingStory.xml.....	330
XMLStory .....	330
XMLElement .....	331
Mapping.....	333
Tags .....	335
 Appendix A. UCF Container Format.....	336
Overview .....	336
Purpose and Scope.....	336
Definitions .....	336
ASCII.....	336
IDML.....	337
IRI .....	337
UCF .....	337
UCF Container .....	337
UCF User Agent.....	337
ODF.....	337
OEBPS .....	337
MIME.....	337
RFC .....	337
Relax NG .....	338
Rootfile .....	338
XML.....	338
Zip .....	338
Relationship of UCF to Other Specifications .....	338
Conformance .....	339
Conforming Containers.....	339
Conforming User Agents .....	339
Future Directions.....	339
UCF Overview.....	340
UCF: A General Container Technology.....	340
“Abstract Container” vs. “Physical Container” .....	340
UCF Container Contents.....	340
File and directory structure .....	340
Relative IRIs for referring to other components .....	341
File Names .....	342
Container media type identification .....	343
META-INF .....	343
Container – META-INF/container.xml (Optional) .....	343
Rootfiles (Optional).....	344
Relationships (Optional) .....	344
Manifest – META-INF/manifest.xml (Optional).....	345
Metadata – META-INF/metadata.xml (Optional).....	345
Digital Signatures – META-INF/signatures.xml (Optional) .....	345
Encryption – META-INF/encryption.xml (Optional) .....	346
Rights Management – META-INF/rights.xml (Optional) .....	346
Zip Container .....	346
RELAX NG UCF Schema .....	347
Comparison of UCF and OCF .....	353
Digital Signatures .....	353
Encryption.....	355

Appendix B. IDML Variants .....	358
InDesign Snippets (.idms files) .....	358
ICML.....	367
InCopy Assignments (ICMA) .....	369
Appendix C. IDML Defaults.....	370
<Document> Attributes .....	370
Languages .....	370
Colors.....	370
Inks .....	371
Swatches.....	371
Gradients .....	371
Stroke Styles .....	371
Font Families.....	371
Character Styles.....	372
[No character style].....	372
Paragraph Styles .....	372
[No paragraph style].....	372
NormalParagraphStyle.....	379
TOC Styles .....	379
Cell Styles.....	380
Table Styles .....	380
[No table style] .....	380
[Basic Table] .....	383
Named Grids.....	383
Object Styles.....	384
[None] .....	384
[Normal Graphics Frame] .....	387
[Normal Text Frame].....	392
Trap Presets.....	401
[No Trap Preset] .....	401
\$ID/kDefaultTrapStyleName .....	402
Text Frame Preferences.....	403
Text Preferences.....	403
Text Defaults.....	404
Anchored Object Defaults.....	411
Anchored Object Settings .....	412
Baseline Frame Grid Options.....	412
Footnote Options .....	412
Text Wrap Preferences.....	414
Document Preferences .....	414
Margin Preferences.....	415
Page Item Defaults .....	416
Frame Fitting Options .....	416
Button Preferences .....	417
Conditional Text Preferences .....	417
XML Tag .....	417
Layer.....	418
Master Spread.....	418
Page.....	420
Spread.....	421
Section .....	423
XmlStory.....	423
IndexingSortOptions .....	424

\$ID/kIndexGroup_Symbol .....	424
\$ID/kIndexGroup_Alphabet .....	424
\$ID/kIndexGroup_Numeric .....	425
\$ID/kWRIndexGroup_GreekAlphabet .....	425
\$ID/kWRIndexGroup_CyrillicAlphabet .....	425
\$ID/kIndexGroup_Kana .....	425
\$ID/kIndexGroup_Chinese .....	426
\$ID/kIndexGroup_Korean .....	426
Bullets .....	426
dABullet0 .....	426
dABullet1 .....	427
dABullet2 .....	427
dABullet3 .....	427
dABullet4 .....	428
Assignment .....	428

# 1 Abstract

IDML is an XML representation of an InDesign document or components. This document describes the structure of IDML files, the XML schema for IDML validation, and provides examples of IDML file content.

# 2 Introduction

IDML is an XML-based format that is capable of fully describing an InDesign document, and is the interchange format for Adobe InDesign CS4 documents. By using an XML representation of text and graphic frames, stories, and other aspects of an InDesign layout, IDML gives you a way to create and modify InDesign documents using tools outside the InDesign application.

Like its precursor INX, IDML is a Document Object Model (DOM) representation of InDesign documents, and is based on the InDesign scripting object model. INX stands for “InDesign Interchange” and was introduced in InDesign CS2 to support the ability to save documents for use in a previous version. INX will continue to be used for backward compatibility for InDesign versions prior to InDesign CS4; IDML will be used for backward compatibility between InDesign CS4 and future versions.

It was quite difficult to write or edit the INX format, as it was designed to be written and consumed by InDesign alone. IDML addresses requests by third party developers and system integrators to make INX more readable and to make it easier to change and assemble InDesign documents using XML tools. By using IDML, you can realize gains in performance, convenience, and flexibility.

IDML is also like INX in that InDesign uses the format for a variety of other purposes, such as library items, InDesign snippets (standalone document fragments saved using IDML markup), and InCopy assignments and stories.

It is important to note that while IDML is an XML format, you do not necessarily need to make use of InDesign’s XML features in an IDML document. In fact, we expect that some users will prefer to keep all XML processing outside of InDesign for a variety of reasons, including performance and the limitations of the implementation of XML in InDesign.

# 3 Terminology

When you open an IDML document in InDesign, the XML elements in the document are converted to objects—both objects in the resulting InDesign document and objects in the InDesign scripting object model. Because of this, we need to be very careful when using certain terms in this specification. When we say “object,” for example, are we referring to the XML element or the InDesign object that the XML element corresponds to after the IDML file has been

opened? The same question applies to the XML attributes and elements that specify properties of InDesign objects.

In this specification, we'll use the phrase *XML element* when we're specifically referring to an element in an IDML file, and we'll use *XML attribute* to refer to an attribute of an XML element. When we use the terms *object* and *property*, we're referring to an entity in an InDesign document (and in the InDesign scripting document object model).

## 4 INX, IDML, and InDesign Scripting

INX, the precursor to IDML, was introduced to give InDesign the ability to save a document for use in a previous version. Rather than write a complicated and error-prone binary file conversion plug-in, we decided to use InDesign's scripting features. When InDesign exports to INX or IDML, we serialize objects and properties from the scripting DOM of an InDesign document to an XML file. When we open an INX or IDML document, InDesign constructs layout objects, sets preferences, enters text, and applies formatting to produce an InDesign document.

IDML will continue to be built on and reflect the scripting DOM view of the InDesign object model. The scripting system provides several vital benefits:

- A high-level isolation from the details and changes in the low-level InDesign object model.
- Well-documented and easily understood architecture for extending the object model and the file format to add third-party data. There is also an existing policy and procedure for developers to register their objects and attributes to ensure that the object names they add are unique within the scripting DOM.
- Significant level of object and property parameter validation and error checking.
- High level versioning support and a well-defined mechanism for adding, deleting, or modifying items.

IDML differs from the scripting DOM in the following ways:

- Scripting events (methods) are not included in IDML document.
- IDML requires that some elements and attributes appear in certain order, while scripting DOM can be viewed as random access.
- Some objects and properties that are included in IDML may not be available in scripting object models for AppleScript, JavaScript, VBScript, and vice versa.
- In some cases, property values that are the same as default values are not written out to IDML.

The following example shows the close connection between scripting and IDML:

**Script (JavaScript):**

```
//Where myRectangle is a reference to a rectangle:  
myRectangle.strokeWeight = .5;  
myRectangle.cornerOption = CornerOptions.bevelCorner;
```

**IDML:**

```
<Rectangle StrokeWeight=".5" CornerOption="BevelCorner" .../>
```

Given the close connection of IDML to InDesign scripting, the InDesign scripting documentation and example scripts provide a great way to learn about the format. The scripting object model is also exposed to scripting clients, and can be viewed from script editors (such as the Adobe ExtendScript Toolkit) for any of the supported scripting languages.

## 4.1 Dynamic Object Model

It is important to note that InDesign's scripting object model is dynamic, and changes based on the active set of plug-ins. If you add plug-ins that support InDesign scripting, objects, properties, enumerations, and methods can appear in the scripting object model, and new objects and properties may appear in the exported IDML. In addition, removing or disabling plug-ins can change the scripting object model and, therefore, the XML elements written to an IDML file.

## 4.2 IDML and Third Party Data

IDML supports the inclusion of new scripting objects and properties added by InDesign plug-ins. Because IDML is built on the InDesign scripting object model, any features added by plug-ins that support InDesign scripting can be included in the IDML package.

That said, IDML does not guarantee round-trip of data added to an InDesign document by third party plug-ins. If the third party plug-in does not implement scripting at all, or if the scripting support for the plug-in is not complete, the data will not be included when you export as IDML.

If the IDML file calls for a plug-in that is not currently installed, InDesign will ignore the plug-in data (i.e., omit it from the InDesign file created by opening the IDML document) and will not include it in a subsequent IDML export of the file. Note that this differs from InDesign's behavior when opening InDesign binary files, where third-party data is maintained when the plug-ins are missing (refer to the InDesign documentation for a more complete description of this behavior).

Since IDML is an XML format, you can add your own XML elements the XML files in the IDML package. Note, however, that InDesign will ignore XML elements that do not exist in the IDML schema when opening the IDML file, and that the data will not be preserved in the converted InDesign binary document (and will therefore not be included in future IDML export from that document).

## 5 Uses for IDML

IDML provides a way for InDesign layouts to interoperate with XML based formats and technologies. Documents can be created and manipulated in IDML format and can then be opened and interpreted using InDesign.

We've designed IDML to make it a key part of automated workflows. Using IDML, you can:

- Generate or modify IDML documents or document elements using data from databases or other data sources (programmatic assembly).
- Reuse parts of IDML documents, or break a document into components that can be used in a development environment (programmatic disassembly).
- Transform document elements using XSLT.
- Find data in InDesign documents using XPath or XQuery.
- Use source control to manage creative content, or to compare two versions of a design.

IDML is intended for consumption by InDesign-family applications, including InDesign, InCopy, and InDesign Server. IDML is not intended as an interchange format for use with applications outside the InDesign family of products, and does not attempt to write or structure InDesign content in a manner that is compatible with other XML layout formats (such as Mars, XSL-FO, or SVG). It is, however, possible to transform IDML content into these formats by applying XSLT transformations to the IDML structures. Writing such a transformation is a very large task, and is beyond the scope of this specification.

## 6 IDML Design Goals

The primary design goals of IDML are as follows:

- Completeness: Any object, property, or preference that can appear in an InDesign document can be represented in IDML. Complete “round trip” compatibility is expected of IDML files.
- Readability: The IDML format is human-readable, and should be easily understood by a user with basic knowledge of the page layout. Representations of InDesign elements—spreads, text frames, stories, and colors, for example—are found inside the XML structure in the same location as they appear in a document. A text frame inside a group, for example, appears inside the group element, which, in turn is inside a spread element.
- For ease of programmatic assembly and disassembly, IDML is designed to be read and written by virtually any program or tool capable of reading and writing XML. Relative to INX, IDML files are easier to process using external tools, while maintaining the same full coverage of the InDesign object model.
- Robustness: The InDesign mechanisms for reading and interpreting IDML will be robust in the face of mistakes and eliminate fatal consequences. A Relax NG schema definition for

IDML will be provided for validation to help customers and workflow developers discover potential errors in their programmatically assembled or edited files. Since even validated files can contain unknown objects, errors will be reported during the interpretation process to assist in debugging.

- Backward compatibility: A user will be able to take an IDML file generated for version X and open it in version X-1 (provided X-1 supports IDML). InDesign CS4 is the first version to provide IDML support. InDesign CS4 will support the INX format for compatibility with InDesign CS3.
- Performance: IDML aims to maintain or exceed the performance of INX in InDesign CS4.
- Improvements over INX: IDML offers the following advantages over INX (this is not an exhaustive list):
  - Provides a published format specification.
  - Avoids obscure element and attribute ordering, relationships, and processing constraints contained in the InDesign export/import code.
  - Provides an XML schema (Relax NG) for validation.
  - Eliminates the use of processing instructions in text content for text object placement

## 7 IDML Design Approaches

To recognize our design goals for IDML, we have adopted a series of design philosophies, which we'll discuss in the following sections.

### 7.1 Separate Content for Efficient Processing

InDesign is often part of workflows where documents are broken into pieces so that they might be worked on in parallel. In a magazine production process, for example, sections, articles, or individual spreads might be given to different graphic artists or layout staff; stories can be given to different writers and editors. The same thing can be true for the process of assembling an IDML document. Stories might be populated with text from RSS feeds; informational graphics on a given spread might be generated from spreadsheet data.

In addition, InDesign documents contain a large number of entities that might be standardized across an organization or publication. Document preferences, styles, fonts, and colors, for example, might be common to all production processes for a particular job. These preferences can be stored in separate XML files and added to an IDML package.

All of the parts of an IDML file are put together in a Zip-compressed archive to represent one InDesign document. Inside this container, a specific “master” file defines the relationships between the component files included in the container.

## 7.2 Maximize Compatibility with InDesign

IDML reflects the scripting DOM view of an InDesign document. It was designed to maximize compatibility and consistency with the InDesign binary file and represent InDesign document as accurately as possible.

IDML files do not retain any view or history data of the document; the intent is to represent the appearance and content of the document in XML.

Being able to convert back and forth between the InDesign binary file format and IDML is a requirement. In most cases, we must convert between IDML and InDesign binary representations, and each representation should be able to carry extensions added to the other.

There are many workflows involving conversions between IDML format and InDesign binary format. We also want to encourage a variety of uses that we may not have envisioned. Some general classes of workflows include following:

- IDML documents can be constructed from information in a database or from a wire feed using XML tools, and then opened in InDesign for further processing.
- An InDesign document can be exported to IDML for use as a template outside of InDesign. The IDML template can be modified using XML tools, then converted back to InDesign.
- An InDesign document can be exported to an IDML format file, which can be opened and saved in previous version of InDesign.

When converting between formats, it is not necessary to maintain binary equivalence. We only need to guarantee a high percentage of visual fidelity.

## 7.3 Maximize Independence of XML Elements

InDesign objects represented as XML elements in IDML are intended to be independent whenever possible. For example, to add, change, or delete a rectangle, you should be able to manipulate the rectangle element in a specific spread file. Other object types are more complicated, however. A text frame element, for example, must include a reference to the story containing the text shown in the text frame. The story itself, including all text formatting attributes, is in a separate file inside the IDML package.

## 7.4 Use Attributes Rather Than Elements

Wherever possible, IDML uses XML attributes, rather than XML elements, for storing most object properties. XML attributes are more compact, and offer performance advantages over XML elements (XML attributes are parsed along with the opening of the element which contains them, rather than as child XML elements).

## 7.5 Self-Documenting

The structure of IDML should be as self-documenting as possible. While the parent-child relationships of the elements do not need to reflect the object relationship in the InDesign binary object model, they generally reflect the InDesign scripting model.

Object and properties from the scripting model are represented in IDML as XML elements and attributes. The tag names of the elements and attributes are the “long name” identifiers for the objects and properties defined in the scripting DOM.

## 7.6 Not a Literal Representation of InDesign Data Structures

IDML is not intended to represent the internal data structure of InDesign, nor is it intended as a replacement for the InDesign SDK. The InDesign API and binary file format will continue to evolve, and plug-in developers will continue to use the existing interfaces. By using the scripting DOM, IDML is insulated from changes to the InDesign API and low level changes to the file system.

## 7.7 Performance

IDML is designed with performance in mind. There are several design decisions that should result in performance improvement of IDML export and import:

- Object information is organized to optimize for import/export.
- At the C++ coding level, IDML core processing code and specific script providers have been tuned for better performance.
- IDML documents have been split into functional components, which will give us the ability to take advantage of multi-tasking during export in a future version.

## 7.8 Support for Previous Versions

It must be possible to open an IDML in the previous version of InDesign (for example opening an IDML file from InDesign CS5 in InDesign CS4). Since IDML is being introduced in InDesign CS4, it is not possible for IDML files from InDesign CS4 to be opened in InDesign CS3 or earlier. Instead, InDesign CS4 will be able to export to INX, which will be able to be opened in earlier versions of InDesign.

When a user opens an IDML file from an incompatible (due to plug-in configuration) version of InDesign/InCopy, the application will display an error message. The user can choose to attempt to open the IDML file.

# 8 IDML Document Structure

When you export a document as IDML, InDesign creates a Zip archive containing multiple XML files. These files use IDML markup to represent the significant parts of the InDesign document.

We split the content of the InDesign document into separate files so that you can work on specific parts of the document—the stories containing the text in the document, for example—without disturbing other document content, such as colors or imported graphics. This compartmentalization makes it easier for automated processes to work on the parts of a document in parallel. It also makes it easier for you to store and re-use fragments of InDesign documents.

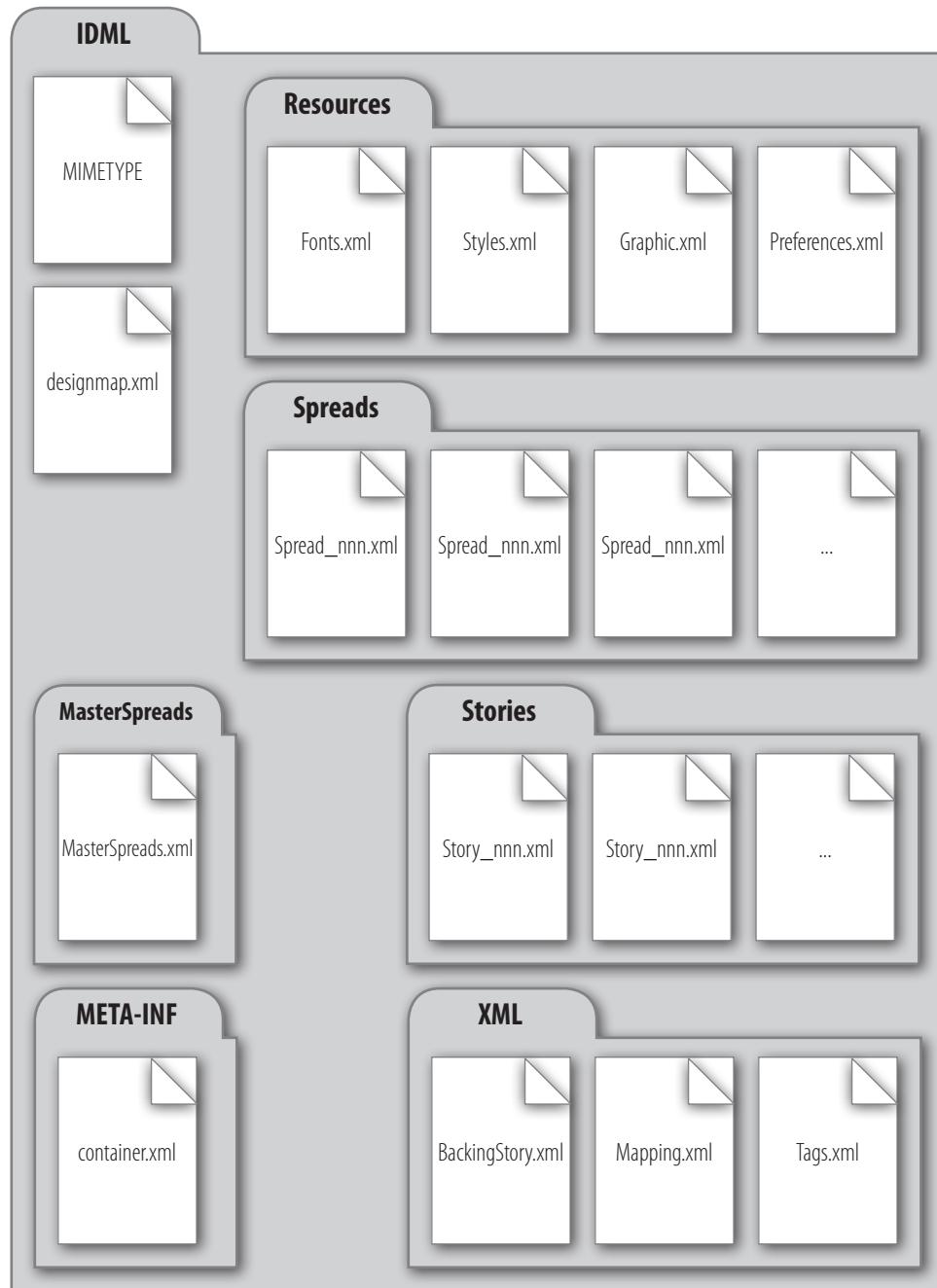
In addition, this approach means that a future version of InDesign will be able to take advantage of multithreading when exporting or importing IDML, which will result in better performance.

## 8.1 IDML Package File Organization

An IDML document exported from InDesign is a Zip archive. For more specific details of the file format, refer to Appendix A, “Universal Container Format.” This is not the only form IDML can take, however: a single-file version of IDML is also used for InDesign Snippet (.idms) and InCopy stories (.icml). In some situations, it might be more practical to work with a single XML file in the IDML format, rather than using the UCF package.

The XML documents saved in an IDML package are arranged to make it easier to find specific content. Files representing spreads, stories, and XML structure, and preferences, for example, are grouped inside folders inside the archive. The following block diagram shows the arrangement of XML files within the IDML archive.

Figure 1. IDML Package Contents



### 8.1.1 MIMETYPE

MIME stands for Multipurpose Internet Mail Extensions, and provides a standard methodology for specifying the content type of files within an IDML package. For more on MIME media types in general, see <hyperlink><http://www.ietf.org/rfc/rfc2045.txt></hyperlink>. For more information on the content of the MIMETYPE file in an IDML package, refer to <hyperlink>"Appendix A: Universal Container Format."</hyperlink>

### **8.1.2 designmap.xml**

The `designmap.xml` file is the key to all of the other files that appear within the IDML package. This file specifies the order in which the spreads appear in the document, maintains the cross references between the resources and content of the file, and defines a variety of document-level attributes not supported by other files. For more on the contents of the `designmap.xml` file , see “Designmap.xml.”

### **8.1.3 Master Spreads Folder and MasterSpreads.xml**

The file `MasterSpreads.xml` contains the master spreads of the document, stored as `<MasterSpread>` elements. Each `<MasterSpread>` element within this file contains all of the page items (rectangles, ellipses, graphic lines, polygons, groups, buttons, and text frames) that appear on the pages of the master spread. For more on the contents of the `MasterSpreads.xml` file , see “Spreads and Master Spreads”.

For information on the naming of MasterSpread files, see “IDML Component Names.”

### **8.1.4 Resources Folder**

The Resources folder in an IDML package contains elements that are commonly used by other files within the document, such as colors, fonts, and paragraph styles. In addition, most of the preferences for the document are stored in this folder. The Resources folder contains the following files:

#### ***Graphic.xml***

The `Graphic.xml` file contains the inks, colors, swatches, gradients, mixed inks, mixed ink groups, tints, and stroke styles contained in the document. For more on the contents of the `Graphic.xml` file , see “Graphics.”

#### ***Fonts.xml***

The `Fonts.xml` file contains the fonts used in the document (including composite fonts, if any). For more on the contents of the `Fonts.xml` file , see “Fonts.”

#### ***Styles.xml***

The `Styles.xml` file contains all of the paragraph, character, object, cell, table, and table of contents (TOC) styles used in the document. For more on the contents of the `Styles.xml` file , see “Styles.”

#### ***Preferences.xml***

The `Preferences.xml` file contains representations of all of the document preferences. For more on the contents of the `Preferences.xml` file , see “Preferences.”

### 8.1.5 Spreads Folder

The Spreads folder contains the XML files representing the spreads in the document. Each spread contains all of the page items (rectangles, ellipses, graphic lines, polygons, groups, buttons, and text frames) that appear on the pages of the spread. The `<Spread>` element also contains `<Page>` elements, which contain attributes and elements that relate to the pages of the spread. Note that `<Page>` elements do not contain page items.

Spreads do not contain text stream content—the `<TextFrame>` XML elements in the spread refer to `<Story>` elements contained in the Stories folder.

For information on the naming of Spread files, see “IDML Component Names.”

### 8.1.6 Stories Folder

The Stories folder contains all of the stories in the InDesign document. Each XML file in the Stories folder of an IDML archive exported from InDesign represents the contents of a single story (as a `<Story>` element) and all of the formatting attributes applied to the text in the story. Stories can also contain other objects, such as inline or anchored frames within the text, or XML elements that have been associated with the text.

Paragraph, character, and object styles used to format the text of the story are not defined within a `<Story>` element. Instead, the story contains cross references (using the unique `self` attribute) to the corresponding styles in the `Styles.xml` file stored in the Resources folder of the archive.

For information on the naming of Story files, see “IDML Component Names.”

### 8.1.7 XML Folder

The XML folder contains XML elements and settings used in the InDesign document.

The XML elements referred to here are the XML elements that actually appear in the InDesign document (i.e., what you see in the Structure view in the InDesign user interface); not the contents of the XML files in the IDML archive. Though an IDML file is made up of XML, the InDesign document it describes does not necessarily contain XML elements.

#### ***BackingStory.xml***

The `BackingStory.xml` file contains the unplaced XML content of the InDesign document (i.e., XML content that has not yet been associated with an element in the layout).

#### ***Tags.xml***

The `Tags.xml` file contains the XML tag definitions stored in the InDesign document, including unused tags.

### **Mapping.xml**

The `Mapping.xml` file contains the style to tag and tag to style mappings defined in the InDesign document.

#### **8.1.8 META-INF Folder**

The `container.xml` file is a standard part of a UCF package and describes the file encoding used by the files in the package. `container.xml` also includes a reference to the root document of the IDML package (usually `designmap.xml`).

## **8.2 IDML Component Names**

An IDML component name can only be defined for a spread, story, or master spread. In an IDML package, each instance of these elements appears as a separate file. The component name is used as the file name for an object instance and it's also used to refer to the file in `designmap.xml`. Note the component name does not change the value of the `self` attribute of the element.

#### **8.2.1 Default Name**

A default name is used if an object in an InDesign document does not have a user-defined component name. The default name is generated by appending the object's UID (an InDesign internal value) to its script object type. For example, the default name for a story with UID 0x1c8 is `Story_uic8`.

#### **8.2.2 User Defined Names**

A user defined name must meet all of the following requirements:

- It must be unique within the IDML package.
- It must not conflict with any existing names for the same type of element.
- It must not conflict with default names. To avoid conflict with default names, a name with the following pattern is *not* valid: starts with the script object name, followed by “\_”, followed by a “u” or “U”, followed by 1 or more hex digits, followed by nothing else. For example, `Story_u123` is not valid (because it matches the default name pattern), but `Story123`, `Story_123`, `Story_u123_Transformed` are all valid user-defined component names.
- It must be a valid file name (it may not contain invalid characters for a file name, and cannot include reserved file name characters).

#### **8.2.3 Reset Component Name on Import**

When an IDML package is opened by InDesign, user-defined component names will be saved in the document so they can be used when the document is exported (that is, they are preserved for round-trip).

Default names are not saved with the document because the object created during import may have a different UID. A new default name is generated when the document is exported.

#### 8.2.4 Scripting Interface

Component names can be accessed through the `IDMLComponentName` property on spread, story, and master spread objects.

#### 8.2.5 C++ API

Component names can be added using the InDesign C++ API. For more on this topic, refer to the *InDesign SDK Programming Guide*.

## 9 IDML Syntax

The following sections provide an overview of IDML syntax, starting with the conventions used in IDML and the way that data types are expressed, and then moving on to the details of the ways that InDesign objects and properties are represented.

### 9.1 XML Conventions and Style

The following sections describe a number of conventions we have adopted for IDML.

#### 9.1.1 Names in IDML

XML elements and attributes in an IDML file will use the “long name” of the corresponding objects in the InDesign scripting object model. For example, the `Rectangle` object in scripting will become a `<Rectangle>` element in IDML; the `StrokeWeight` property will become a `strokeWeight` attribute (shown here in a `<Rectangle>` element):

```
<Rectangle StrokeWeight = "6"/>
```

Note that the elements representing text objects in IDML differ from the text objects used in the scripting object model. In general, the `CharacterStyleRange` element in IDML corresponds to the `TextStyleRange` object in the scripting object model. Both represent a continuous run of identical formatting, and both have roughly the same set of formatting values (represented by properties in scripting and by attributes in IDML).

#### 9.1.2 Use of Empty Elements

Whenever possible, IDML uses empty elements to represent objects or properties. For example, a `<MarginPreference>` element in a `<Page>` element usually looks something like this:

```
<MarginPreference ColumnCount="1" ColumnGutter="12" Top="36" Bottom="36" Left="36"
Right="36" ColumnDirection="Horizontal" ColumnsPositions="0 540"/>
```

As you can see, the `<MarginPreference>` element does not contain any child elements—all of the properties of the element are represented as attributes. Empty elements can be processed more rapidly than elements containing child elements.

## 9.2 Generating IDML Schema

The complete IDML schema can be written as Relax NG Compact Syntax files. You can generate these schema files using the following script:

```
//Fill in the location of a folder on your system in the following line.
app.generateIDMLSchema(Folder("/c/IDMLSchema"), false);
```

When you generate an IDML schema with the second parameter set to false, all of the schema elements are contained in two files: `IDMarkupLanguage.rnc` and `datatype.rnc` (the latter file is included by reference in the former). If you want to view only the schema elements for a specific type of file in the IDML package (e.g., a `Story_nnn.xml` file), you generate separate schema files corresponding to each XML file included in the IDML package (`MasterSpreads.xml`, `Spread.xml`, `Text.xml`, etc.). To do this, run the following script:

```
//Fill in the location of a folder on your system in the following line.
app.generateIDMLSchema(Folder("/c/IDMLSchema"), true);
```

The output folder you specified will include the following schema files (in the same folder organization as in an IDML package):

```
datatype.rnc
designmap.rnc
MasterSpreads.rnc
Fonts.rnc
Graphic.rnc
Preferences.rnc
Styles.rnc
Spread.rnc
Story.rnc
BackingStory.rnc
Mapping.rnc
Tags.rnc
```

Note that the content of the `datatype.rnc` file is the same in either method of generating the schema.

## 9.3 Data Types in IDML

IDML data types are declared in a Relax NG Compact Syntax file named `datatype.rnc`, which will be included by all other schema files. This schema can be extended by InDesign plug-ins by adding objects and properties to the scripting object model. New versions of the schema can be generated at any time by InDesign. For more on generating the file, see “Generating IDML Schema”.

The following table shows a list of basic types. The first column lists the script data types, the second column is the corresponding type name used in the Relax NG schema, and the third column is the value of the type attribute that appears in the IDML file (Note that the type attribute will only appear in a child element of the `<Properties>` element):

**Table 1. IDML basic data types**

Script Data Type	Schema Data Type	Value of Type Attribute in an IDML File
boolean	xsd:boolean	boolean
string	xsd:string	string
short integer	xsd:short	short
long integer	xsd:int	long
longlong integer	xsd:int	longlong
double	xsd:double	double
object	xsd:string	object
object list	xsd:string	list of strings as a space-separated string
list	xsd:string	list of simple types as a space-separated string
date	xsd:date	date
file	xsd:string	file
enumeration	xsd:string	Depends on the definition of the enumeration specified in <code>datatype.rnc</code>
unit	xsd:double	unit or double
record	text	record
stream	text	binary

### 9.3.1 Scalar Data Types vs. Complex Data Types

Scalar types are basic types which have a single value associated with them. They are the building blocks upon which all other data types are constructed.

Complex types are comprised of combinations of the basic types. Lists and records are two examples of complex types which exist today in the scripting object model.

- Lists are homogenous collections of elements. They are similar to arrays used by traditional programming languages. Lists contain an attribute that identifies the length of the list.

- Records are heterogeneous collections of elements. Records may contain other records. Records may also be contained within a list.

### 9.3.2 Enumerations

InDesign scripting makes frequent use of enumerations to define the scope of a property value. For example, story types can be `Regulartext`, `Toctext`, or `Indexingtext`. InDesign's IDML export uses the following rules to determine how an enumeration value should be expressed:

- When the value of a property is always an enumeration, it is expressed as an attribute of an element, and its data type is "string".
- If the value of a property's can be one of several different types, including an enumeration, the property will be expressed as a child element of a `<Properties>` element, with the `type` attribute of `enumeration`. For example, the following segment of the `<DocumentPreference>` element indicates the column and margin guide colors are "Green".

IDML schema example (from designmap.rnc, edited to remove off-topic attributes):

#### **Schema Example 1. ColumnGuideColor and MarginGuideColor Elements**

```
DocumentPreference_Object = element DocumentPreference {
    element Properties {
        element ColumnGuideColor { InDesignUIColorType_TypeDef }?&
        element MarginGuideColor { InDesignUIColorType_TypeDef }?
    }
    ?
}
```

Example from an IDML package (edited to remove off-topic attributes):

#### **IDML Example 1. Properties Expressed as Attributes or Elements, Depending on Their Value**

```
<DocumentPreference>
    <Properties>
        <ColumnGuideColor type="enumeration">Violet</ColumnGuideColor>
        <MarginGuideColor type="list">
            <ListItem type="double">66</ListItem>
            <ListItem type="double">60</ListItem>
            <ListItem type="double">196</ListItem>
        </MarginGuideColor>
    </Properties>
</DocumentPreference>
```

In the above example, the value of a `<ColumnGuideColor>` or `<MarginGuideColor>` can be either an InDesign `UIColorType` enumeration (defined in the `datatype.rnc` file), or an RGB color (defined as an array of three doubles). Because the value can be more than a single, simple type, it is expressed as an element, rather than as an attribute. When the color is an enumeration, as in the `<ColumnGuideColor>` element shown above, the name of the enumeration appears as the value of the element. When the color is an RGB array, it is represented by a series of elements, as shown by the child elements of the `<MarginGuideColor>` element.

### 9.3.3 Key String

Many elements and attributes in IDML refer to various settings in an InDesign document by name (default colors or styles, for example) or by the string displayed in the InDesign user interface. These strings can change as the locale (or language) of the application changes.

IDML can use a “key string” and an untranslated string for these values. The key string is used to look up a localized (translated) string. An untranslated string is used “as is” with no lookup. Both of these string usages are represented as script data type `string`.

The four character prefix `$ID/` inserted at the beginning of a string indicates that the string is a key string, and that InDesign should look up the appropriate localized string during the IDML import process. For example, in the following abbreviated `<IndexOptions>` element, the value of the `Title` attribute is a key string.

```
<IndexOptions Title="$ID/Index"/>
```

In the example above, the use of the key string means that InDesign will look up the localized string during the process of importing the IDML document. Once the document is open, the correct string for the current locale will be displayed in the relevant areas of the user interface, and the corresponding value (object or other setting) will be applied to the property. If, instead, the value of the attribute was simply the term “Index”, InDesign would display that string, regardless of the locale of the application, and would attempt to apply an object or setting of that name.

If InDesign cannot find a string value to replace the key string, the key string will be used as is.

The following example shows the syntax used when a key string appears in an element:

```
<RuleAboveGapColor type="string">$ID/Text Color</RuleAboveGapColor>
```

### 9.3.4 Measurement Units

Measurement units in IDML exported from InDesign are always points (defined as 72 units per inch).

## 9.4 Representation of Objects

InDesign objects (spreads, colors, or text frames, for example) are represented by XML elements in IDML. Each object has properties that can be expressed as child XML elements or as attributes.

An object in an InDesign document is generally written into an IDML XML element in the following (simplified) format:

**IDML Example 2. Object Serialization**

```
<Object SimpleProperty="Value" ... Self="UniqueID">
  <Properties>
    <ComplexProperty>Value</ComplexProperty>
  </Properties>
  <ChildObject>...</ChildObject>
  ...
</Object>
```

Where *Object* is the name of the InDesign object type, *SimpleProperty* is a property that can be expressed as an XML attribute, *Self* is a unique identifier for the object, *Value* is the value of the property, *ComplexProperty* is a property that must be expressed as an XML element (see “Representation of Properties”), and *ChildObject* is an object contained by the object (e.g., an object inside a group). The content of the *<ChildObject>* element follows the same pattern as the *<Object>* element.

The *<Properties>* and *<ChildObject>* sections are optional, and depend on the object being serialized. If both are empty, object can be written as simplified form:

```
<Object SimpleAttribute="Value", ... Self="UniqueID"/>
```

## 9.5 Representation of Properties

Properties can be expressed either as attributes or child elements of the containing XML element. In general, simple values are represented as attributes; more complex values are represented as elements.

### 9.5.1 Properties Represented as Attributes

An object property will be expressed as an XML attribute if it meets any of the following conditions:

- The property is defined as simple scripting type (i.e., not *StreamType*, *VariableType*, or *RecordType*, see “Properties Represented as Elements”) and its property name is not “Contents”. Because the Contents property may be a very long string and may contain line ending characters, it is not suitable to be represented as an XML attribute.
- The value of the property is one dimensional array where each member of the array is a simple data type. It is not necessary for every member to be the same type.
- All of the possible values of the property are either objects or enumerations.

The name of the scripting property is generally used as the key of the XML attribute, and the value of the property is stored as the value of the attribute. The value can be either a single value, or a list of values separated by spaces. For example:

```
PointSize="12"
ColorValue="0 0 0 100"
```

Note that if the value of the property contains space character, it is encoded as "%20".

### 9.5.2 Properties Represented as Elements

If a given property does not meet the rules described above for being written as an attribute, it will be expressed as an XML element inside the `<Properties>` child element of the XML element representing the containing object. The name of the scripting property is used as the name of the element, and the value of the property is serialized as the content of the element. In general, the data type of the value will be specified in the `type` attribute of the element.

Properties are expressed as elements in two basic forms: as a single value and as a list of values.

**Table 2. Property Elements**

Single value	<pre>&lt;Properties&gt;   &lt;GuideColor type="enumeration"&gt;LightBlue&lt;/GuideColor&gt; &lt;/Properties&gt;</pre>
List of values	<pre>&lt;Properties&gt;   &lt;GuideColor type="list"&gt;     &lt;ListItem type="double"&gt;128&lt;/ListItem&gt;     &lt;ListItem type="double"&gt;0&lt;/ListItem&gt;     &lt;ListItem type="double"&gt;255&lt;/ListItem&gt;   &lt;/GuideColor&gt; &lt;/Properties&gt;</pre>

In the example above, the guide color property of an InDesign guide is expressed as an element because it can be either a `UIColors_EnumValue` enumeration or a custom RGB color.

The scripting object model contains two data types that are always represented as elements in IDML: `RecordType`, and `Geometry`. In addition, some properties of the type `VariableType` are written to IDML as elements. The following sections discuss these types.

### **RecordType**

Properties whose data type (in the scripting object model) is RecordType are serialized into XML as a series of child elements inside the <Properties> element, as shown in the following example (where *TabList* is the property name and *Alignment*, *AlignmentCharacter*, *Leader*, and *Position* are items in the property record):

#### **IDML Example 3. RecordType Serialization**

```
<TabList type="record">
  <Alignment type="enumeration">LeftAlign</Alignment>
  <AlignmentCharacter type="string">$ID/.</AlignmentCharacter>
  <Leader type="string">$ID/</Leader>
  <Position type="unit">36</Position>
</TabList>
```

### **Geometry**

Page item geometry (the shape of the page item) is represented as a <PathGeometry> element. This element, in turn, contains one or more <GeometryPathType> elements, each of which, in turn, contains two or more <PathPoint> elements.

The <PathGeometry> element has the following form:

#### **IDML Example 4. Geometry Serialization**

```
<PathGeometry>
  <GeometryPathType PathOpen="false">
    <PathPointArray>
      <PathPointType Anchor="72 -324" LeftDirection="72 -324"
        RightDirection="72 -324"/>
      <PathPointType Anchor="72, -252" LeftDirection="72, -252"
        RightDirection="72, -252"/>
      <PathPointType Anchor="144 -252" LeftDirection="144 -252"
        RightDirection="144 -252"/>
      <PathPointType Anchor="144 -324" LeftDirection="144 -324"
        RightDirection="144 -324"/>
    </PathPointArray>
  </GeometryPathType>
</PathGeometry>
```

For more on geometry in IDML, see “[Spreads and Master Spreads](#).”

### **VariableType**

Properties in the scripting object model with the data type *StreamType* or properties whose name is *Contents* are represented as XML elements.

### **Complex Structures**

If the scripting object model data type of a property is more complex than the structures we have discussed, it is serialized as a list of elements. The `<Descriptor>` element (found in the `<Page>` element) is an array containing one or more elements of any type. It is serialized as an element containing a series of property elements, as shown in the following example:

```
<Descriptor type="list">
  <ListItem type="string"></ListItem>
  <ListItem type="enumeration">Arabic</ListItem>
  <ListItem type="boolean">true</ListItem>
  <ListItem type="boolean">false</ListItem>
  <ListItem type="long">1</ListItem>
  <ListItem type="string"></ListItem>
</Descriptor>
```

In general, properties serialized as `<ListItem>` elements are used for storing data for round-trip to and from InDesign and InCopy, and can be thought of as read-only properties. It is unlikely that you will ever need to construct these elements.

#### **9.5.3 Use of the Type Attribute in Property Elements**

If a `<Properties>` element is used to represent the properties of an object, each property is represented as a child element of the `<Properties>` element. Each child element, includes a `type` attribute whose value is set to the data type (see “Data Types in IDML”) of the property. For example:

```
<Properties>
  <BaselineFrameGridColor type="enumeration">LightBlue</BaselineFrameGridColor>
</Properties>
```

In the above example, the value `LightBlue` comes from the definition of the `UIColors` enumeration in the `datatype.rnc` file.

#### **9.5.4 Object Reference Format**

Properties in the scripting object model often contain references to objects. For example, the `FillColor` property of a `Rectangle` will always refer to a `Color`, `Tint`, `MixedInk`, `Gradient`, or `Swatch` object. IDML expresses this relationship by including a reference to the XML element representing the object (which is stored elsewhere in the document) as the value of the XML element or attribute representing the property. Object references are the most common cross reference format within an IDML file.

When serializing an object into an XML element, InDesign generates a unique string for the ID of the element and stores it in the `Self` attribute. The algorithm used for generating the ID is complex, and proprietary to Adobe, but it generally follows these rules:

- If an object belongs to a class that identifies its members by name, then the value of the `Self` attribute of the element will be that name. If the name is not unique, then InDesign will add a prefix and/or postfix strings to create a unique ID upon import.
- If an object is a child of another object, the name or ID of the parent object will be added to the value of the `Self` attribute as a prefix string, using a pattern similar to path syntax:

```
/parent/childname
```

- If an object does not have a name, the value of the ID property of the object will be used as the value of the `Self` attribute of the element during export.
- If multiple objects exist of the same type, index numbers will be added to the `Self` attribute as a postfix string.

The only requirement of the value of the `Self` attribute is that it is unique within the IDML package. If you are writing the IDML yourself, you do not need to observe the above pattern—you can change the value of the `Self` attribute to anything you want as long as it is unique (within the IDML package) and as long as all references to the element are also changed to match.

```
<Story Self = "ucb" .../>
```

The following shows an example of an element reference expressed as the value of an XML element. In the following example, the `<BasedOn>` element of a `<ParagraphStyle>` element refers to another `<ParagraphStyle>` element whose `Self` attribute contains the value `Heading` (the name of the paragraph style). In this case, the paragraph style is defined in the `Styles.html` file inside the Resources folder in the IDML package.

```
<BasedOn type="object">ParagraphStyle\Heading</BasedOn>
```

Cross references can also appear as values of attributes. In the following example, the `TitleStyle` attribute of the `<IndexOptions>` element refers to a `<ParagraphStyle>` element whose `Self` attribute contains the value `IndexTitle` (the name of the paragraph style). In this case, the paragraph style is defined in the `Styles.html` file inside the Resources folder in the IDML package.

```
<IndexOptions TitleStyle="ParagraphStyle\IndexTitle" .../>
```

The following example shows a cross reference that uses the `Self` attribute of an element to refer to the element. The value `ucb` is the unique identifier of the `<Story>` element, as shown above.

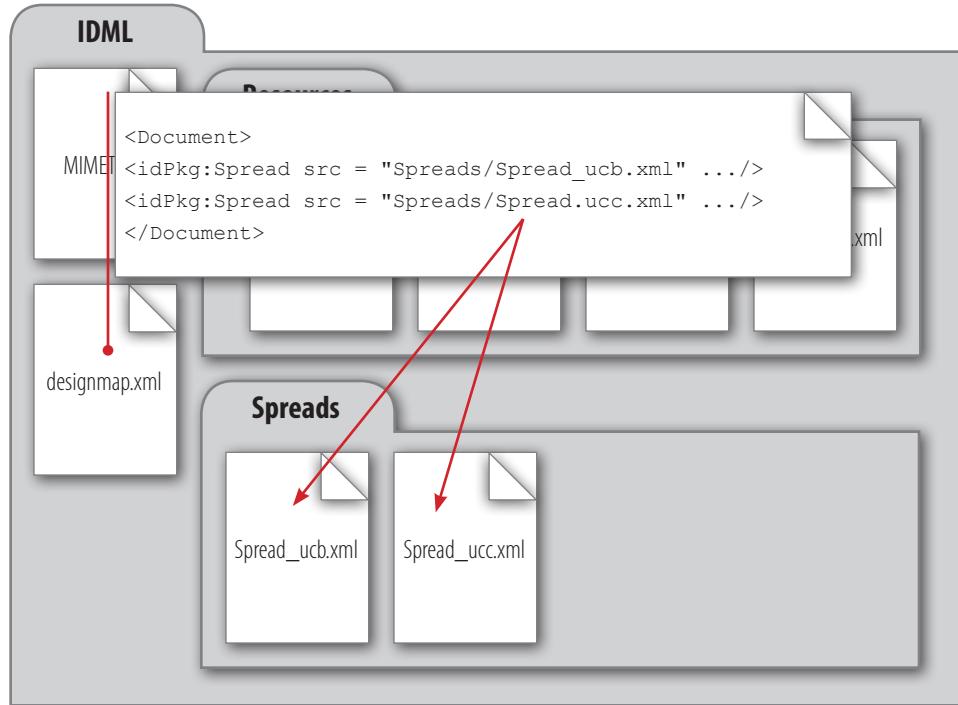
```
<TextFrame ParentStory="ucb" .../>
```

- IDML files can contain forward references (i.e., references to objects which have not yet been included in the IDML file).
- References are valid within an IDML package or within a single IDML file. References from one package to another or one file to another are not allowed.

### 9.5.5 Cross-file Reference Examples

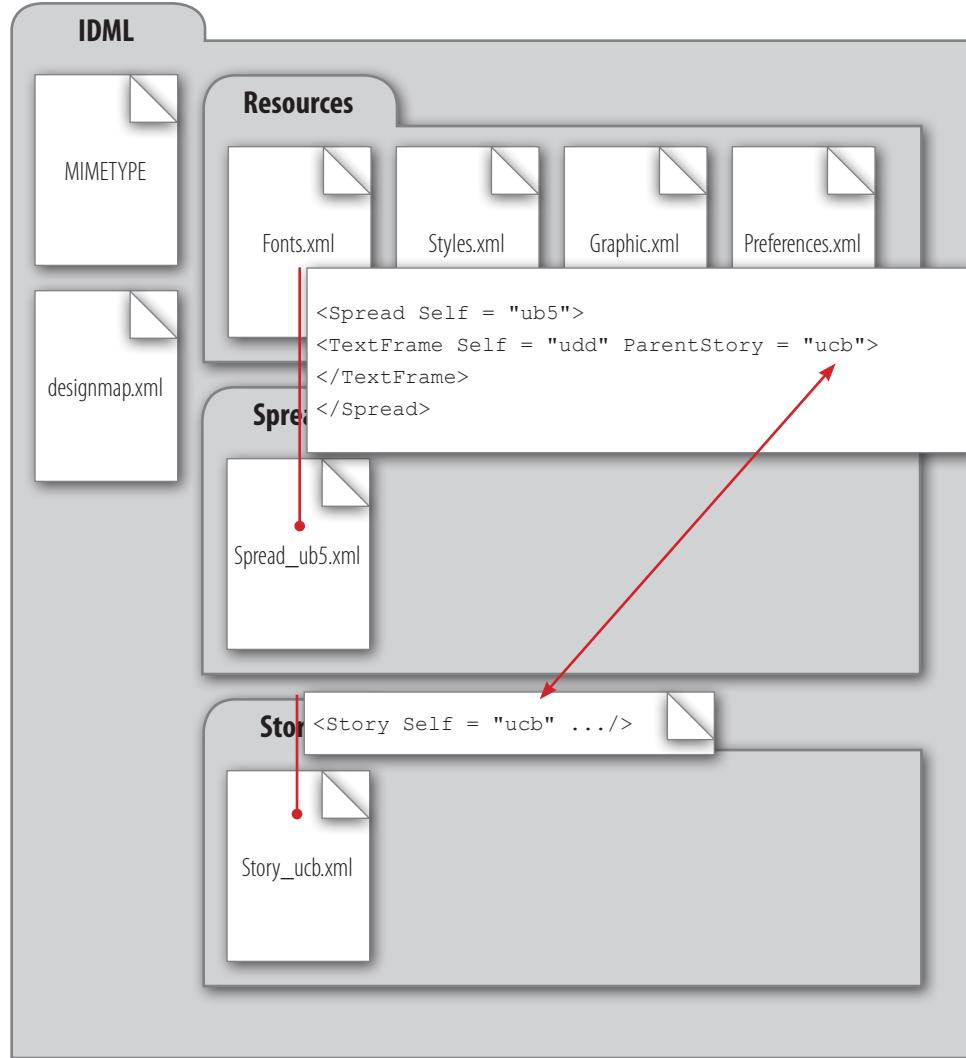
The following example shows how the <Document> element in a `designmap.xml` file can refer to the spreads that make up the document.

Figure 2. Cross Reference: <Document> to Spread file



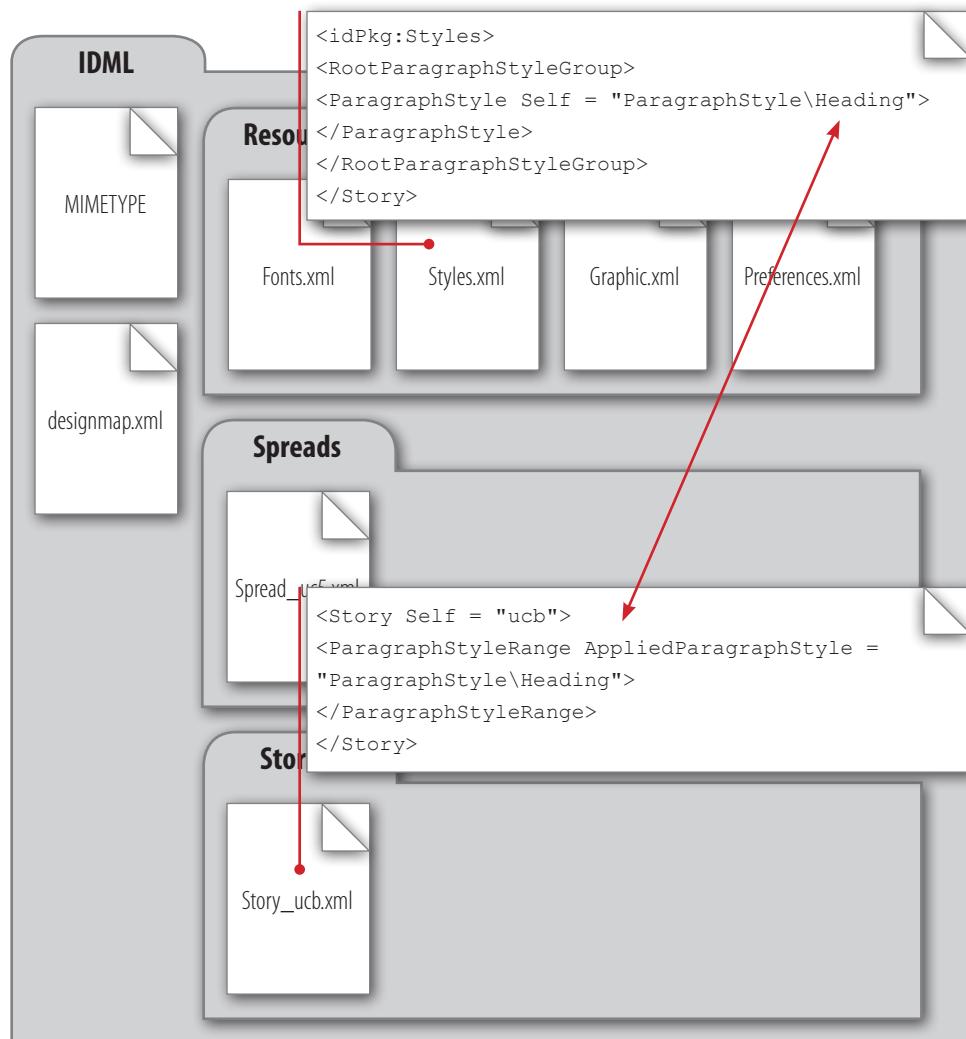
The following example shows how the `ParentStory` attribute of a `<TextFrame>` element in a `Spread.xml` file refers to the story that contains the text in the text frame.

Figure 3. Cross Reference: `<TextFrame>` to `<Story>`



The following example shows how the `AppliedParagraphStyle` attribute of a `<ParagraphStyle-Range>` element in a `Story.xml` file refers to the `Self` attribute of the `<ParagraphStyle>` applied to the text.

Figure 4. Cross Reference: &lt;ParagraphStyleRange&gt; to &lt;ParagraphStyle&gt;



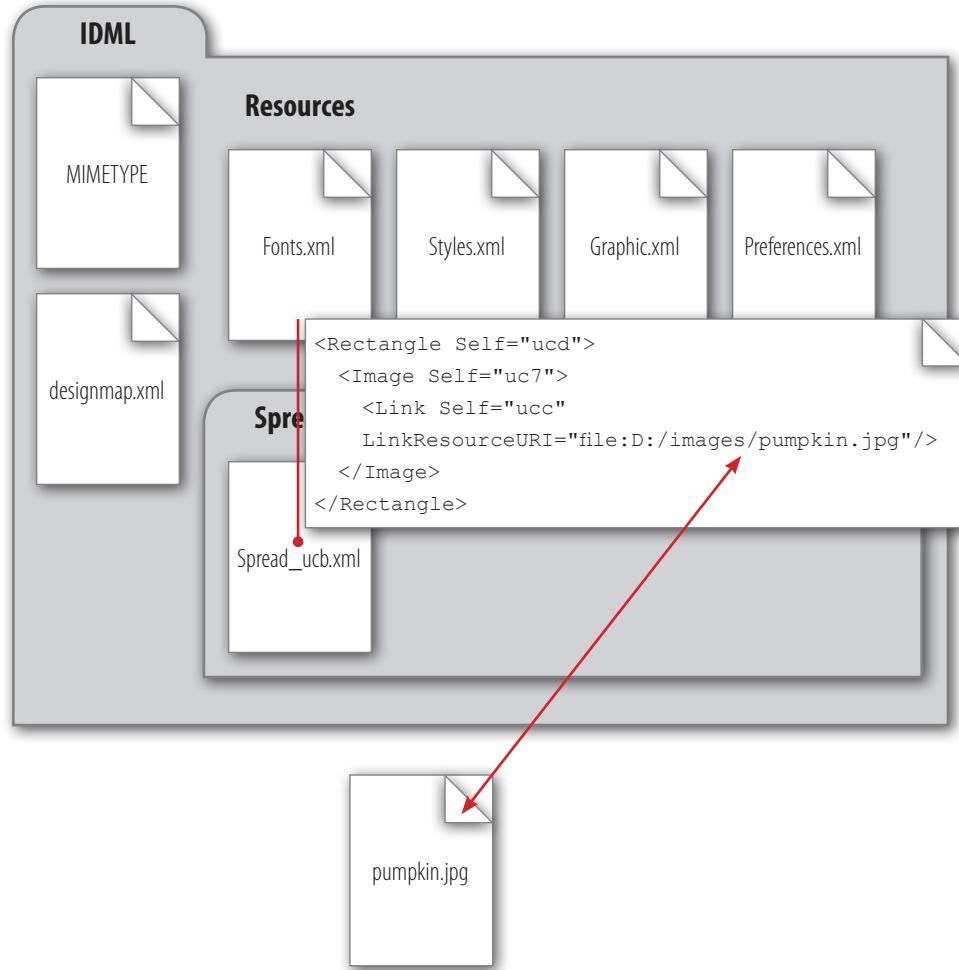
In the example shown above, the `<ParagraphStyle>` is part of the `<RootParagraphStyleGroup>`. IDML uses a syntax similar to that of a file path to refer to paragraph styles within their containing group. For example, if a style named “ListFirst” is in a paragraph style group named “ListStyles,” a reference to the style would look like the following:

```
AppliedParagraphStyle = "ParagraphStyle\ListStyles%30aListFirst"
```

In this example, a colon, encoded as `%30a`, acts as a path separator character.

The following example shows how a <Rectangle> element can refer to an external file. Linked files are stored outside the IDML package file. If a file cannot be found when the IDML file is opened in InDesign, it will be listed as a missing link.

Figure 5. Cross Reference: Referring to an External File



The following example shows how to refer to an external hyperlink destination.

Figure 6. Cross Reference: Hyperlink Source and Hyperlink Destination



# 10 IDML File Reference

This section provides detailed information on the content and structure of each type of file that can appear in an IDML document.

## 10.1 Common Attributes and Elements

A number of attributes and elements are shared by a large number of elements in the files that make up an IDML package. Rather than repeat the definition of these elements, we'll document them in the following sections.

### 10.1.1 Self

The `Self` attribute contains a unique identifier for the elements that contain it. This identifier is used elsewhere in the IDML package to refer to the element, as discussed in the [“Object Reference Format”](#) section of this specification.

**Schema Example 2. Self**

```
attribute Self { xsd:string }
```

The following example shows the `Self` attribute of a `<Story>` element.

**IDML Example 5. Self**

```
<Story Self = "udd" ...>
```

### 10.1.2 Scripting Labels

Many of the elements in the files contained in an IDML package can contain a `<Label>` element. This element represents a feature of InDesign's scripting object model: most non-text objects can contain any number of key/value pairs as strings. These objects have a default label property, but can have custom labels associated with them. Both the key and the value of a script label can be strings of any length.

For more on using script labels, refer to the Scripting chapter of the *Adobe InDesign CS4 Scripting Guide* for your scripting language of choice (AppleScript, JavaScript, or VBScript).

**Schema Example 3. Label**

```
element Properties {
    element Label { element KeyValuePair { KeyValuePair_TypeDef }*
    }?
}
```

The following example shows both the default label (`Key` attribute is “`Label`”) and a custom label that has been added with the scripting method `insertLabel`.

**IDML Example 6. Label**

```
<Label>
<KeyValuePair Key="Label" Value="This is a script label."/>
```

```

</Label>
<Label>
  <KeyValuePair Key="myCustomLabel" Value="This is a custom script label."/>
</Label>

```

Element Name	Element Description		
Label	The scripting label(s) associated with the object. Optional.		
Attribute Name	Type	Req	Description
Key	string	yes	The key of the label.
Value	string	no	The string stored in the label.

**Note:** Only the default label is visible in the InDesign user interface (in the Script Label panel).

### 10.1.3 Optional Values, Defaults, and Preferences

As you look through the IDML schema, you'll notice that most of the attributes and elements are marked as being optional. When you look at the schema for a `<Rectangle>` element, for example, you'll see that the `<PathGeometry>` element is optional. Since the `<PathGeometry>` element contains the list of path points that define the shape of the rectangle (see "Page Item Geometry"), it is difficult to understand how it can be optional. What does this mean, in practical terms?

First, it is technically true that the `<PathGeometry>` element can be omitted. If you omit the `<PathGeometry>` element from the `<Rectangle>` element, InDesign will create a default-sized rectangle when you open the IDML file. The default size and location of the rectangle don't really matter—the point is that InDesign supplies default values for the missing values in the IDML file. While this is an extreme case—we don't expect that you'll ever want to construct a rectangle without specifying its size and location—it shows that you can construct documents from very minimal XML elements.

When you look at an IDML file that you've exported from InDesign, you'll see a very large number of XML elements. A `<Document>` element alone usually contains dozens of `<Language>`, `<TextVariable>`, and `<CrossReferenceFormat>` elements, among others. When you create IDML files yourself, you can omit these elements. They are only included in the IDML file by InDesign to ensure round-trip fidelity of the document. If you omit these default elements, InDesign will create the corresponding default objects in the InDesign document as it opens the IDML file.

This behavior corresponds to the way that InDesign handles defaults and preferences when you create documents using the user interface. When you use user interface controls to change various settings when no documents are open, you're modifying the *application defaults*. Application defaults change the way that all new documents are created, but have no effect on existing documents. When you change settings in a document when no objects are selected, on the other hand, you're changing the *document defaults*. All new objects created in the document will take on the appropriate default values, but existing objects in the document (and objects in other documents) will be unaffected. (For more on InDesign's defaults handling, refer to the online help.)

IDML, however, cannot rely on the application defaults, because they can be changed by the user. IDML also cannot rely on document defaults, because the IDML schema specifies that most or all of the document default values are optional. Instead, IDML gets default values for omitted optional values from the IDML defaults file. This mechanism takes the place of both application and document defaults in the user-interface scenario described above, and guarantees that

the behavior of an IDML document will remain consistent, regardless of the user settings and installation details (such as the locale of the application). If a document default value is not found in the IDML document, InDesign will use the corresponding value from the IDML defaults file.

Typically, users set document defaults to specify a commonly-used formatting attribute—the default font, stroke weight, or fill color, for example. IDML mirrors this practice by providing two very useful and important elements: `<PageItemDefault>` and `<TextDefault>` (in an IDML package file, you'll find these in the `Preferences.xml` file inside the Resources folder). The values that you provide in these elements define the default formatting for text and page items in the document.

When you define a default value for an attribute or element in one of these elements, all elements of a given type that have not explicitly overridden that value will inherit the formatting specified by the default.

Here's an example: let's say that you want the default stroke weight for all page items in a document to be two points. To do this, you change the value of the `StrokeWeight` attribute of the `<PageItemDefault>` element to 2. This means that all page items that you add to the document will have a stroke weight of two points. If you want to override the default stroke weight on a page item, you enter a different value in the corresponding attribute in the element defining the page item. In this example, if you want to specify that the stroke weight of a particular rectangle is one point, you'd set the value of the `StrokeWeight` attribute of the `<Rectangle>` object to 1. This value overrides the formatting specified in the `<PageItemDefault>` element.

In short, if the page item itself specifies the stroke weight, then InDesign uses that stroke weight value. If the page item does not specify the stroke weight, then InDesign uses the stroke weight value stored in the `<PageItemDefault>` element. If the `<PageItemDefault>` element IDML document does not specify a stroke weight, or if the IDML document does not contain a `<PageItemDefault>` element, then InDesign will use the stroke weight value from the `<PageItemDefault>` element in the IDML defaults file.

**Note:** The above example is a bit of an oversimplification in that it assumes that the default object style applied to new rectangles is InDesign's built-in "None" object style (which is an object style that cannot be edited). This is InDesign's default behavior, but can be changed by users. In IDML, you would specify this default by setting the `AppliedObjectStyle` attribute of the `<PageItemDefault>` element to `ObjectStyle/$ID/[None]`.

This makes writing your own IDML much easier—if you know that most of the page items you'll be creating in an IDML document will share the same formatting, you can specify that formatting in the `<PageItemDefault>` element and then omit the corresponding attributes and elements when you enter the `<Rectangle>`, `<Oval>`, `<Polygon>`, and `<GraphicLine>` elements that share those formatting attributes.

**Note:** If you are creating your own InDesign snippet files (.idms), you cannot specify the default formatting in this way—the `<PageItemDefault>` and `<TextDefault>` elements are not used in snippets. Instead, snippet files get their default formatting from the InDesign documents in which they are placed. This is the same behavior as importing a snippet file created by InDesign, but it means that if you're creating a snippet file, you'll need to fully specify all formatting that you don't want overridden by the formatting of the documents the snippet will be imported into. For more on importing and exporting snippets, refer to the InDesign online help. For more on the differences between the snippet format and document-level IDML, refer to "Appendix B: Snippets and ICML Documents."

As you continue to look through an IDML document exported from InDesign, you'll also see a number of preference elements (such as the `<XMLImportPreference>`, `<XMLExportPreference>`, or `<LayoutAdjustmentPreference>` elements). Again, these elements are included to maintain round-trip fidelity of the InDesign document, and have no effect on the construction of spreads, stories, and page items in an IDML file. The settings in the `<XMLImportPreference>` element, for example, will not have an effect until a user chooses *File>Import XML* in the InDesign document created by opening the IDML document. The content of the IDML document does not change.

When you're writing your own IDML, you can generally omit these preferences objects. The only case in which you might want to include these preference elements is if you are creating files in a workgroup setting and need to maintain standard behaviors across all documents. If, for example, your workgroup requires that the Adjust Layout feature is turned on, you'd set the `EnableLayoutAdjustment` attribute of the `<LayoutAdjustmentPreference>` element to `true`. For more on the preferences objects, see "Preferences.xml."

## 10.2 designmap.xml

The `designmap.xml` file contains a “road map” to the XML elements that make up the document, and defines a variety of document-level attributes. Some elements, such as hyperlinks and cross references, are collected in the `<Document>` element; this makes it easier to refer to them from other parts of the IDML file or package.

This section describes the contents of the `designmap.xml` file in an IDML package file. If you’re creating a “single file” IDML document, you’ll use the same elements, and use the same method of referring to elements in the file (generally by the contents of their `Self` attributes) but won’t need to include the cross-file references.

### Schema Example 4. Document

```

Document_Object = element Document {
    attribute DOMVersion { "6.0" },
    attribute Self { xsd:string },
    attribute ActiveProcess { xsd:string }?,
    attribute TransparencyAttributeDefaultProperty { xsd:string }?,
    attribute StoryList { list { xsd:string * } }?,
    attribute FullName { xsd:string }?,
    attribute Name { xsd:string }?,
    attribute Visible { xsd:boolean }?,
    attribute FilePath { xsd:string }?,
    attribute Modified { xsd:boolean }?,
    attribute Saved { xsd:boolean }?,
    attribute ZeroPoint { UnitPointType_TypeDef }?,
    attribute ActiveLayer { xsd:string }?,
    attribute UnusedSwatches { list { xsd:string * } }?,
    attribute Converted { xsd:boolean }?,
    attribute Recovered { xsd:boolean }?,
    attribute ReadOnly { xsd:boolean }?,
    attribute CMYKProfileList { list { xsd:string * } }?,
    attribute RGBProfileList { list { xsd:string * } }?,
    attribute CMYKProfile { xsd:string }?,
    attribute RGBProfile { xsd:string }?,
    attribute SolidColorIntent { RenderingIntent_EnumValue }?,
    attribute AfterBlendingIntent { RenderingIntent_EnumValue }?,
    attribute DefaultImageIntent { RenderingIntent_EnumValue }?,
    attribute RGBPolicy { ColorSettingsPolicy_EnumValue }?,
    attribute CMYKPolicy { ColorSettingsPolicy_EnumValue }?,
    attribute AccurateLABSpots { xsd:boolean }?,
    element Properties {
        element InstanceList { element IndexInstanceType { IndexInstanceType_TypeDef }* }?
    }?&
    element Label { element KeyValuePair { KeyValuePair_TypeDef }* }?
}
?
(
    Language_Object*,
    element idPkg:Graphic { attribute src {"Resources/Graphic.xml"} }?,

```

## 42 IDML File Reference: designmap.xml

```

element idPkg:Fonts { attribute src {"Resources/Fonts.xml"} }?,
KinsokuTable_Object*,
MojikumiTable_Object*,
element idPkg:Styles { attribute src {"Resources/Styles.xml"} }?,
NumberingList_Object*,
NamedGrid_Object*,
Condition_Object*,
ConditionSet_Object*,
(element idPkg:Preferences { attribute src {"Resources/Preferences.xml"} } ?&
MetadataPacketPreference_Object?&
ConditionalTextPreference_Object?),  

TextVariable_Object*,  

element idPkg:Tags { attribute src {"XML/Tags.xml"} }?,  

Layer_Object*,  

element idPkg:MasterSpread { attribute src {xsd:string {pattern = ".*\.\xml"} } }*,  

element idPkg:Spread { attribute src {xsd:string {pattern = ".*\.\xml"} } }*,  

Section_Object*,  

DocumentUser_Object*,  

CrossReferenceFormat_Object*,  

element idPkg:BackingStory { attribute src {"XML/BackingStory.xml"} }?,  

element idPkg:Story { attribute src {xsd:string {pattern = ".*\.\xml"} } }*,  

HyperlinkPageDestination_Object*,  

HyperlinkURLDestination_Object*,  

HyperlinkExternalPageDestination_Object*,  

HyperlinkPageItemSource_Object*,  

Hyperlink_Object*,  

DTD_Object*,  

element idPkg:Mapping { attribute src {"XML/Mapping.xml"} }?,  

Index_Object*,  

Bookmark_Object*,  

(PreflightProfile_Object*&  

DataMergeImagePlaceholder_Object*&  

HyphenationException_Object*&  

IndexingSortOption_Object*&  

ABullet_Object*&  

Assignment_Object*)
)
}

```

**Table 3. Document Properties Represented as Attributes**

Name	Type	Req	Description
AccurateLABSpots	boolean	no	If true, uses LAB alternates for spot colors when available.
ActiveLayer	string	no	The active layer.
ActiveProcess	string	no	The active preflight process for this document.

Name	Type	Req	Description
AfterBlending-Intent	RenderingIntent_EnumValue	no	The rendering intent for colors that result from transparency interactions on the page after blending. Can be UseColorSettings (Uses the current color settings), Perceptual (Preserves the visual relationship between colors at the expense of actual color values; most suitable for photographic images with high percentages of out-of-gamut colors), Saturation (Produces vivid colors at the expense of color accuracy; most suitable for business graphics such as graphs or charts), RelativeColorimetric (Compares the extreme highlight of the source color space to that of the destination color space and shifts all colors accordingly; out-of-gamut colors are shifted to the closest reproducible color in the destination color space), or AbsoluteColorimetric (Maintains color accuracy at the expense of preserving relationships between colors; most suitable for previewing how paper color affects printed colors).
CMYKPolicy	ColorSettings-Policy_EnumValue	no	The policy for handling colors in a CMYK color model, including reading and embedding color profiles, mismatches between embedded color profiles and the working space, and moving colors from one document to another. Can be ColorPolicyOff (Turns off color management for documents whose profiles do not match the working space For imported colors, numeric values override color appearance), PreserveEmbeddedProfiles (Preserves embedded color profiles in newly opened documents), ConvertToWorkingSpace (Converts newly opened documents to the current working space For imported colors, color appearance overrides numeric values), or CombinationOfPreserveAndSafeCmyk (Preserves raw color numbers and ignores embedded color profiles).
CMYKProfile	string	no	The current CMYK profile.
CMYKProfileList		no	A list of valid CMYK profiles.
Converted	boolean	no	If true, the Document was converted.

Name	Type	Req	Description
DefaultImage-Intent	RenderingIntent_EnumValue	no	The rendering intent for bitmap images. Can be UseColorSettings (Uses the current color settings), Perceptual (Preserves the visual relationship between colors at the expense of actual color values; most suitable for photographic images with high percentages of out-of-gamut colors), Saturation (Produces vivid colors at the expense of color accuracy; most suitable for business graphics such as graphs or charts), RelativeColorimetric (Compares the extreme highlight of the source color space to that of the destination color space and shifts all colors accordingly; out-of-gamut colors are shifted to the closest reproducible color in the destination color space), or Absolute-Colorimetric (Maintains color accuracy at the expense of preserving relationships between colors; most suitable for previewing how paper color affects printed colors).
FilePath	string	no	The full path to the file.
FullName	string	no	The full path to the Document, including the name of the Document.
Modified	boolean	no	If true, the Document has been modified since it was last saved.
Name	string	no	The name of the Document.
RGBPolicy	ColorSettings-Policy_EnumValue	no	The policy for handling colors in an RGB color model, including reading and embedding color profiles, handling mismatches between embedded color profiles and the working space, and moving colors from one document to another. Can be ColorPolicyOff (Turns off color management for documents whose profiles do not match the working space For imported colors, numeric values override color appearance), PreserveEmbeddedProfiles (Preserves embedded color profiles in newly opened documents), ConvertToWorkingSpace (Converts newly opened documents to the current working space For imported colors, color appearance overrides numeric values), or CombinationOf-PreserveAndSafeCmyk (Preserves raw color numbers and ignores embedded color profiles).
RGBProfile	string	no	The current RGB profile.
RGBProfileList		no	A list of valid RGB profiles.
ReadOnly	boolean	no	If true, the Document is read-only.
Recovered	boolean	no	If true, the Document was recovered.
Saved	boolean	no	If true, the Document has not been saved since it was created.

Name	Type	Req	Description
SolidColorIntent	RenderingIntent_EnumValue	no	The rendering intent for all vector art (areas of solid color) in native objects. Can be UseColorSettings (Uses the current color settings), Perceptual (Preserves the visual relationship between colors at the expense of actual color values; most suitable for photographic images with high percentages of out-of-gamut colors), Saturation (Produces vivid colors at the expense of color accuracy; most suitable for business graphics such as graphs or charts), RelativeColorimetric (Compares the extreme highlight of the source color space to that of the destination color space and shifts all colors accordingly; out-of-gamut colors are shifted to the closest reproducible color in the destination color space), or AbsoluteColorimetric (Maintains color accuracy at the expense of preserving relationships between colors; most suitable for previewing how paper color affects printed colors).
StoryList	string	no	The list of stories in the document, as a sequence of references to the Self attribute of each story, separated by spaces.
Transparency-AttributeDefault-Property	string	no	Transparency defaults for the document.
UnusedSwatches	string	no	A list of the swatches that are not being used, as a sequence of references to the Self attribute of each swatch, separated by spaces.
Visible	boolean	no	If true, the Document is visible.
ZeroPoint	UnitPointType_TypeDef	no	The ruler origin, specified as page coordinates in the format [x, y].

**Table 4. Document Properties Represented as Elements**

InstanceList	IndexInstance-Type_TypeDef	no	A list of the index instances that have been placed in the document.
--------------	----------------------------	----	--

**IDML Example 7. Document Element Using References to Other Files in the IDML Package**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Document xmlns:idPkg="http://ns.adobe.com/AdobeInDesign/idml/1.0/packaging"
Self="d" StoryList="ucb">
  <idPkg:Graphic src="Resources/Graphic.xml"/>
  <idPkg:Fonts src="Resources/Fonts.xml"/>
  <idPkg:Styles src="Resources/Styles.xml"/>
  <idPkg:Preferences src="Resources/Preferences.xml"/>
  <idPkg:Tags src="XML/Tags.xml"/>
  <idPkg:MasterSpread src="MasterSpreads/MasterSpread_ub6.xml"/>
  <idPkg:Spread src="Spreads/Spread_ub5.xml"/>
  <idPkg:BackingStory src="XML/BackingStory.xml"/>
  <idPkg:Story src="Stories/Story_ucb.xml"/>
</Document>
```

In the above example, the document contains a single story file, `Story_ucb.xml`, and a single spread file, `Spread_ub5.xml`. All of the other `idPkg:` elements refer to standard files and file locations inside the IDML package file. In the following example, the `<Story>` and `<Spread>` elements are contained within the `designmap.xml` file itself.

#### **IDML Example 8. Document Element in a *designmap.xml* File**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<?aid style="50" type="document" readerVersion="6.0" featureSet="257"
product="6.0(351)" ?>
<Document DOMVersion="6.0" Self="d" StoryList="u122" ActiveLayer="uc2" >
  <DocumentPreference PageHeight="792" PageWidth="612" PagesPerDocument="1"
FacingPages="true" DocumentBleedTopOffset="0" DocumentBleedBottomOffset="0"
DocumentBleedInsideOrLeftOffset="0" DocumentBleedOutsideOrRightOffset="0"
DocumentBleedUniformSize="true" SlugTopOffset="0" SlugBottomOffset="0"
SlugInsideOrLeftOffset="0" SlugRightOrOutsideOffset="0"
DocumentSlugUniformSize="false" PreserveLayoutWhenShuffling="true"
AllowPageShuffle="true" OverprintBlack="true" PageBinding="LeftToRight"
ColumnDirection="Horizontal" ColumnGuideLocked="true" MasterTextFrame="false"
SnippetImportUsesOriginalLocation="false"/>
<Layer Self="uc2" Name="Layer 1" Visible="true" Locked="false" IgnoreWrap="false"
ShowGuides="true" LockGuides="false" UI="true" Expendable="true"
Printable="true"/>
<Section Self="ua6" Length="1" Name="" PageNumberStyle="Arabic"
ContinueNumbering="true" IncludeSectionPrefix="false" PageNumberStart="1"
Marker="" PageStart="ucf" SectionPrefix="" />
<MasterSpread Self="uc3" ItemTransform="1 0 0 1 0 0" Name="A-Master"
NamePrefix="A" BaseName="Master" ShowMasterItems="true" PageCount="2"
AppliedMaster="n">
  <Page Self="uc8" Name="A" AppliedMaster="n">
    <MarginPreference ColumnCount="1" ColumnGutter="12" Top="36"
Bottom="36" Left="36" Right="36" ColumnDirection="Horizontal"
ColumnsPositions="0 540"/>
  </Page>
  <Page Self="uc9" Name="A" AppliedMaster="n">
    <MarginPreference ColumnCount="1" ColumnGutter="12" Top="36" Bottom="36"
Left="36" Right="36" ColumnDirection="Horizontal" ColumnsPositions="0 540"/>
  </Page>
</MasterSpread>
<Spread Self="uca" PageCount="1" AppliedMaster="uc3" BindingLocation = "0"
ItemTransform="1 0 0 1 0 0">
  <Page Self="ucf" Name="1" AppliedMaster="uc3"/>
  <TextFrame Self="ud1" ParentStory="u122" PreviousTextFrame="n"
NextTextFrame="n" ItemLayer="uc2" ContentType="TextType"
ItemTransform="1 0 0 1 0 0">
    <Properties>
      <PathGeometry>
        <GeometryPathType PathOpen="false">
          <PathPointArray>
            <PathPointType Anchor="72 -324" LeftDirection="72 -324"
RightDirection="72 -324"/>
            <PathPointType Anchor="72 -252" LeftDirection="72 -252"
RightDirection="72 -252"/>
            <PathPointType Anchor="144 -252" LeftDirection="144 -252"
RightDirection="144 -252"/>
          </PathPointArray>
        </GeometryPathType>
      </PathGeometry>
    </Properties>
  </TextFrame>
</Spread>
```

```

        <PathPointType Anchor="144 -324" LeftDirection="144 -324"
                      RightDirection="144 -324"/>
      </PathPointArray>
    </GeometryPathType>
  </PathGeometry>
</Properties>
</TextFrame>
</Spread>
<Story Self="u122">
  <ParagraphStyleRange>
    <CharacterStyleRange>
      <Content>Hello World!</Content>
    </CharacterStyleRange>
  </ParagraphStyleRange>
</Story>
</Document>

```

### 10.2.1 Documents and Color Management

The appearance of all swatches (colors, tints, gradients, mixed inks, and mixed ink groups, described in section <hyperlink>“Graphics.xml”</hyperlink>) and imported graphics is determined by the color management profiles applied to the document. The profile does not change the base properties of these objects (e.g., it does not change the CMYK color values of a color defined in the `Graphic.xml` file inside the IDML package); it only affects the rendering of the color for display or output (printing and export).

#### IDML Example 9. Color Management Attributes

```

<Document Self="d" CMYKProfile="U.S. Web Coated (SWOP) v2"
          RGBProfile="sRGB IEC61966-2.1" SolidColorIntent="UseColorSettings"
          AfterBlendingIntent="UseColorSettings" DefaultImageIntent="UseColorSettings"
          RGBPolicy="PreserveEmbeddedProfiles" CMYKPolicy="CombinationOfPreserveAndSafeCmyk"
          AccurateLABSpots="false">

```

A complete discussion of InDesign’s color management features is beyond the scope of this document. For more information, refer to the InDesign documentation.

**Note:** The colors used for drawing user interface items (guides, grids, layer highlights, etc.) are simply RGB screen values on a given system, and are not color managed.

### 10.2.2 Language

The <Language> elements in an IDML package define the language dictionaries available for the document. You cannot create languages by adding new <Language> elements; they are included for use as references (from, for example, <ParagraphStyle> elements in the `Styles.xml` file in the Resources folder of the IDML package), and to maintain round-trip fidelity for InDesign and In-Copy documents.

#### Schema Example 5. Language

```

Language_Object = element Language {
  attribute Self { xsd:string },
  attribute Name { xsd:string },
  attribute SingleQuotes { xsd:string }?,
}

```

```

        attribute DoubleQuotes { xsd:string }?,
        attribute PrimaryLanguageName { xsd:string }?,
        attribute SublanguageName { xsd:string }?,
        attribute Id { xsd:int }?,
        attribute HyphenationVendor { xsd:string }?,
        attribute SpellingVendor { xsd:string }?,
        element Properties {
            element Label { element KeyValuePair { KeyValuePair_TypeDef }*
                }?
        }
    ?
}

```

**Table 5. Language Properties Represented as Attributes**

Name	Type	Req	Description
DoubleQuotes	string	no	The double quotes pair for the language.
HyphenationVendor	string	no	The hyphenation rules source.
Id	int	no	The unique ID of the Language.
Name	string	yes	The name of the Language.
PrimaryLanguage- Name	string	no	The name of the language.
SingleQuotes	string	no	The single quotes pair for the language.
SpellingVendor	string	no	The spell-checking source.
SublanguageName	string	no	The sub-language name of the language.

### 10.2.3 Condition

InDesign documents can feature conditional text–text that is only visible in the layout when a specific state is enabled. The `<Condition>` element controls the appearance of the text governed by a condition.

#### Schema Example 6. Condition

```

Condition_Object = element Condition {
    attribute Self { xsd:string },
    attribute Name { xsd:string },
    attribute IndicatorMethod { ConditionIndicatorMethod_EnumValue }?,
    attribute UnderlineIndicatorAppearance
    { ConditionUnderlineIndicatorAppearance_EnumValue }?,
    attribute Visible { xsd:boolean }?,
    element Properties {
        element IndicatorColor { InDesignUIColorType_TypeDef }?&
        element Label { element KeyValuePair { KeyValuePair_TypeDef }*
            }?
    }
}

```

**Table 6. Condition Properties Represented as Attributes**

Name	Type	Req	Description
IndicatorMethod	Condition- IndicatorMethod_ EnumValue	no	The condition indicator method.
Name	string	yes	The name of the Condition.
Underline- Indicator- Appearance	Condition- Underline- Indicator- Appearance_Enum- Value	no	The condition underline indicator appearance.
Visible	boolean	no	If true, the Condition is visible.

**Table 7. Condition Properties Represented as Elements**

Name	Type	Req	Description
IndicatorColor	InDesignUIColor- Type_TypeDef	no	The color for the condition indicator, specified either as an array of three doubles, each in the range 0 to 255 and representing R, G, and B values, or as a UI color. Can return: Array of 3 Reals (0 - 255) or UIColors enumerator.

#### 10.2.4 ConditionalTextPreference

##### Schema Example 7. ConditionalTextPreference

```
ConditionalTextPreference_Object = element ConditionalTextPreference {
    attribute Self { xsd:string },
    attribute ShowConditionIndicators { xsd:boolean }?,
    attribute ActiveConditionSet { xsd:string }?
}
```

**Table 8. ConditionalTextPreference Properties Represented as Attributes**

Name	Type	Req	Description
ShowCondition- Indicators	boolean	no	If true, display the condition indicators in the user interface.
ActiveCondition- Set	string	no	A reference to the active condition set, as the value of the Self attribute of the <Condition> element.

### 10.2.5 TextVariable

A text variable is an item you insert in text that varies according to the context. For example, the Last Page Number variable displays the page number of the last page of the document. If you add or remove pages, the variable is updated accordingly. The text variables in an IDML document are defined by `<TextVariable>` elements. Text variables come in a variety of types: `<CustomText-VariablePreference>`, `<FileNameVariablePreference>`, `<PageNumberVariablePreference>`, `<ChapterNumberVariablePreference>`, `<DateVariablePreference>`, `<MatchCharacterStyle-Preference>`, or `<MatchParagraphStylePreference>` element.

The type of a text variable is defined by the `VariableType` attribute of the `<TextVariable>` element, and the definition of the text variable is specified in a child element of the `<Text-Variable>` element.

A `<TextVariable>` element stored in the `<Document>` element is only the definition of the text variable. Text variable instances appear in `<Story>` elements, and all of the formatting of the text variable instance is defined there, not in the `<Document>` element. For more on text variables, refer to the InDesign online help.

#### Schema Example 8. TextVariable

```
TextVariable_Object = element TextVariable {
    attribute Self { xsd:string },
    attribute Name { xsd:string },
    attribute VariableType { VariableTypes_EnumValue }?,
    (
        CustomTextVariablePreference_Object?&
        FileNameVariablePreference_Object?&
        PageNumberVariablePreference_Object?&
        ChapterNumberVariablePreference_Object?&
        DateVariablePreference_Object?&
        MatchCharacterStylePreference_Object?&
        MatchParagraphStylePreference_Object?
    )
}
```

---

**Table 9. TextVariable Properties Represented as Attributes**

Name	Type	Req	Description
Name	string	no	The name of the text variable.
VariableType	VariableTypes_EnumValue	no	The TextVariable type. Can be <code>CustomText-Type</code> (Custom text variable), <code>FileNameType</code> (File name variable), <code>LastPageNumberType</code> (Last page number variable), <code>ChapterNumber-Type</code> (Chapter number variable), <code>OutputDate-Type</code> (Output date variable), <code>CreationDate-Type</code> (Creation date variable), <code>Modification-DateType</code> (Modification date variable), <code>MatchCharacterStyleType</code> (Running header (character style) variable), or <code>MatchParagraph-StyleType</code> (Running header (paragraph style) variable).

Text variables share some attributes, as shown in the following table.

---

**Table 10. Common Text Variable Properties Represented as Attributes**

Name	Type	Req	Description
TextBefore	string	no	Text that appears before the text variable instance.
TextAfter	string	no	Text that appears after a text variable instance.

**Schema Example 9. CustomTextVariablePreference**

```
CustomTextVariablePreference_Object = element CustomTextVariablePreference {
    attribute Self { xsd:string },
    element Properties {
        element Contents {
            (string_type, xsd:string ) |
            (enum_type, SpecialCharacters_EnumValue ) |
            (object_type, xsd:string )
        }?
    }
    ?
}
```

---

**Table 11. CustomTextVariablePreference Properties Represented as Elements**

Name	Type	Req	Description
Contents	string or Special- Characters_Enum- Value or string (a reference to a self attribute)	no	The text contents of the CustomTextVariable- Preference.

**Schema Example 10. FileNameVariablePreference**

```
FileNameVariablePreference_Object = element FileNameVariablePreference {
    attribute Self { xsd:string },
    attribute TextBefore { xsd:string }?,
    attribute IncludePath { xsd:boolean }?,
    attribute IncludeExtension { xsd:boolean }?,
    attribute TextAfter { xsd:string }?
}
```

---

**Table 12. FileNameVariablePreference Properties Represented as Attributes**

Name	Type	Req	Description
IncludePath	boolean	no	If true, include the file path in the text variable instances.
IncludeExtension	boolean	no	If true, include the file extension in the text variable instances.

**Schema Example 11. PageNumberVariablePreference**

```
PageNumberVariablePreference_Object = element PageNumberVariablePreference {
    attribute Self { xsd:string },
    attribute TextBefore { xsd:string }?,
    attribute Format { VariableNumberingStyles_EnumValue }?,
    attribute TextAfter { xsd:string }?,
    attribute Scope { VariableScopes_EnumValue }?
}
```

---

**Table 13. PageNumberVariablePreference Properties Represented as Attributes**

Name	Type	Req	Description
Format	Variable-NumberingStyles_EnumValue	no	The format of the page number. Can be Current, Arabic, UpperRoman, LowerRoman, UpperLetters, LowerLetters, Kanji, FullWidthArabic, SingleLeadingZeros, or DoubleLeadingZeros.
Scope	VariableScopes_EnumValue	no	The scope of the page number variable. Can be DocumentScope or SectionScope.

**Schema Example 12. ChapterNumberVariablePreference**

```
ChapterNumberVariablePreference_Object = element ChapterNumberVariablePreference {
    attribute Self { xsd:string },
    attribute TextBefore { xsd:string }?,
    attribute Format { VariableNumberingStyles_EnumValue }?,
    attribute TextAfter { xsd:string }?
}
```

---

**Table 14. ChapterNumberVariablePreference Properties Represented as Attributes**

Name	Type	Req	Description
Format	Variable-NumberingStyles_EnumValue	no	The format of the chapter number. Can be Current, Arabic, UpperRoman, LowerRoman, UpperLetters, LowerLetters, Kanji, FullWidthArabic, SingleLeadingZeros, or DoubleLeadingZeros.

**Schema Example 13. DateVariablePreference**

```
DateVariablePreference_Object = element DateVariablePreference {
    attribute Self { xsd:string },
    attribute TextBefore { xsd:string }?,
    attribute Format { xsd:string }?,
    attribute TextAfter { xsd:string }?
}
```

---

**Table 15. DateVariablePreference Properties Represented as Attributes**

Name	Type	Req	Description
Format	string	no	The format of the date variable.

**Schema Example 14. MatchCharacterStylePreference**

```
MatchCharacterStylePreference_Object = element MatchCharacterStylePreference {
    attribute Self { xsd:string },
    attribute TextBefore { xsd:string }?,
    attribute TextAfter { xsd:string }?,
    attribute AppliedCharacterStyle { xsd:string }?,
    attribute SearchStrategy { SearchStrategies_EnumValue }?,
    attribute ChangeCase { ChangeCaseOptions_EnumValue }?,
    attribute DeleteEndPunctuation { xsd:boolean }?
}
```

---

**Table 16. MatchCharacterStylePreference Properties Represented as Attributes**

Name	Type	Req	Description
AppliedCharacter-Style	string	no	A reference to the character style applied to the text variable (as the value of the Self attribute of the <CharacterStyle>).
ChangeCase	ChangeCase-Options_EnumValue	no	Change the case of the text variable. Can be Uppercase, Lowercase, Titlecase, or Sentencecase.
DeleteEnd-Punctuation	boolean	no	If true, delete any punctuation at the end of the text variable.
SearchStrategy	SearchStrategies_EnumValue	no	The search strategy applied to the text variable. Can be FirstOnPage or LastOnPage.

**Schema Example 15. MatchParagraphStylePreference**

```
MatchParagraphStylePreference_Object = element MatchParagraphStylePreference {
    attribute Self { xsd:string },
    attribute TextBefore { xsd:string }?,
    attribute TextAfter { xsd:string }?,
    attribute AppliedParagraphStyle { xsd:string }?,
    attribute SearchStrategy { SearchStrategies_EnumValue }?,
    attribute ChangeCase { ChangeCaseOptions_EnumValue }?,
    attribute DeleteEndPunctuation { xsd:boolean }?
}
```

---

**Table 17. MatchParagraphStylePreference Properties Represented as Attributes**

Name	Type	Req	Description
AppliedCharacter-Style	string	no	A reference to the paragraph style applied to the text variable (as the value of the Self attribute of the <ParagraphStyle>).
ChangeCase	ChangeCase-Options_EnumValue	no	Change the case of the text variable. Can be Uppercase, Lowercase, Titlecase, or Sentencecase.
DeleteEnd-Punctuation	boolean	no	If true, delete any punctuation at the end of the text variable.
SearchStrategy	SearchStrategies_EnumValue	no	The search strategy applied to the text variable. Can be FirstOnPage or LastOnPage.

**IDML Example 10. TextVariable**

```
<TextVariable Self="dTextVariablesLast Page Number" Name="Last Page Number"
  VariableType="LastPageNumberType">
  <PageNumberVariablePreference
    Self="dTextVariablesLast Page NumberPageNumberVariablePreference1"
    TextBefore="" Format="Current" TextAfter="" Scope="SectionScope"/>
</TextVariable>
```

**10.2.6 Layer**

InDesign documents can contain layers, which are transparent planes on which you can arrange the page items in your layout. Layers can be used to control the stacking order of objects in a document, but they can also be used to organize objects in a document. Layers in an InDesign document are document-wide. In IDML, `<Layer>` elements appear inside the `<Document>` element in the `designmap.xml` file.

For more on layers, refer to the InDesign online help.

**Schema Example 16. Layer**

```
Layer_Object = element Layer {
  attribute Self { xsd:string },
  attribute Name { xsd:string }?,
  attribute Visible { xsd:boolean }?,
  attribute Locked { xsd:boolean }?,
  attribute IgnoreWrap { xsd:boolean }?,
  attribute ShowGuides { xsd:boolean }?,
  attribute LockGuides { xsd:boolean }?,
  attribute UI { xsd:boolean }?,
  attribute Expendable { xsd:boolean }?,
  attribute Printable { xsd:boolean }?,
  element Properties {
    element LayerColor { InDesignUIColorType_TypeDef }?&
    element Label { element KeyValuePair { KeyValuePair_TypeDef }* }?
  }?
}
```

---

**Table 18. Layer Properties Represented as Attributes**

Name	Type	Req	Description
Expendable	boolean	no	If true, the layer can be deleted.

Name	Type	Req	Description
IgnoreWrap	boolean	no	If true, objects on the layer ignore text wrap.
Locked	boolean	no	If true, the layer is locked.
LockGuides	boolean	no	If true, the guides on the layer are locked.
Name	string	no	The name of the layer.
Printable	boolean	no	If true, the objects on the layer can be printed.
ShowGuides	boolean	no	If true, show the guides assigned to the layer.
Visible	boolean	no	If true, the layer is visible in the user interface.

**Table 19. Layer Properties Represented as Elements**

Name	Type	Req	Description
LayerColor	string or InDesignUIColorType	no	The color of the layer, specified either as an array of three doubles, each in the range 0 to 255 and representing R, G, and B values, or as an InDesignUIColorType enumeration.

**IDML Example 11. Layer**

```
<Layer Self="ub5" Name="Layer 1" Visible="true" Locked="false" IgnoreWrap="false"
ShowGuides="true" LockGuides="false" UI="true" Expendable="true" Printable="true">
<Properties><LayerColor type="enumeration">LightBlue</LayerColor></Properties>
</Layer>
```

**10.2.7 Section**

Page ranges in an InDesign document can be broken up into sections. Section properties control the page numbering system in use in the pages of the section. For more on sections, refer to the InDesign online help.

**Schema Example 17. Section**

```
Section_Object = element Section {
    attribute Self { xsd:string },
    attribute Length { xsd:int }?,
    attribute Name { xsd:string }?,
    attribute PageNumberStyle { PageNumberStyle_EnumValue }?,
    attribute ContinueNumbering { xsd:boolean }?,
    attribute IncludeSectionPrefix { xsd:boolean }?,
    attribute PageNumberStart { xsd:int {minInclusive="1" maxInclusive="999999"} }?,
    attribute Marker { xsd:string }?,
    attribute PageStart { xsd:string }?,
    attribute SectionPrefix { xsd:string }?,
    element Properties {
        element Label { element KeyValuePair { KeyValuePair_TypeDef }* }?
    }?
}
```

**Note:** InDesign features a variety of approaches to numbering and cross references; section numbering can interact with special characters, text variables, and paragraph numbering.

**Table 20. Section Properties Represented as Attributes**

Name	Type	Req	Description
ContinueNumbering	boolean	no	If true, continue page numbering from the previous section in the document.
IncludeSection-Prefix	boolean	no	If true, include the value of the SectionPrefix attribute in the nameo of the section.
Length	int	no	The number of pages in the section.
Marker	string	no	The marker character for the section.
Name	string	no	The name of the section.
PageNumberStart	int	no	The starting page number of the section. Range: 1 to 999999.
PageNumberStyle	PageNumber_Enum-Style	no	The page numbering style applied to the section. Can be UpperRoman, LowerRoman, UpperLetters, LowerLetters, Arabic, Kanji, DoubleLeadingZeros, TripleLeadingZeros, ArabicAlifBaTah, ArabicAbjad, Hebrew-Biblical, or HebrewNonStandard.
PageStart	string	no	A reference to the page that starts the section (as the value of the Self attribute of the <Page> element).
SectionPrefix	string	no	The prefix for the section.

### 10.2.8 CrossReferenceFormat

InDesign documents can contain cross references. Cross references are made up of <Building-Block> elements.

#### Schema Example 18. CrossReferenceFormat

```
CrossReferenceFormat_Object = element CrossReferenceFormat {
    attribute Self { xsd:string },
    attribute Name { xsd:string },
    attribute AppliedCharacterStyle { xsd:string }?,
    element Properties {
        element Label { element KeyValuePair { KeyValuePair_TypeDef }* }
    }?
}
?
(
    BuildingBlock_Object*
)
}
```

**Table 21. CrossReferenceFormat Properties Represented as Attributes**

Name	Type	Req	Description
AppliedCharacter-Style	string	no	A reference to the character style applied to the cross reference format (as the value of the Self attribute of the <CharacterStyle>) . A reference to the character style applied to the text variable (as the value of the Self attribute of the <CharacterStyle>) .
Name	string	yes	The name of the cross reference format.

Cross references are made up of “building blocks”—elements that specify text and text formatting that can be added to a cross reference (for more on building blocks, refer to the InDesign documentation). The <CrossReferenceFormat> element can contain multiple <BuildingBlock> elements.

#### Schema Example 19. BuildingBlock

```
BuildingBlock_Object = element BuildingBlock {
    attribute Self { xsd:string },
    attribute BlockType { BuildingBlockTypes_EnumValue },
    attribute AppliedCharacterStyle { xsd:string }?,
    attribute CustomText { xsd:string }?,
    attribute AppliedDelimiter { xsd:string }?,
    attribute IncludeDelimiter { xsd:boolean }?
}
```

**Table 22. BuildingBlock Properties Represented as Attributes**

Name	Type	Req	Description
AppliedCharacter-Style	string	no	A reference to the character style applied to the cross reference format (as the value of the Self attribute of the <CharacterStyle>) . A reference to the character style applied to the text variable (as the value of the Self attribute of the <CharacterStyle>) .
AppliedDelimiter	string	no	The delimiter character of the building block.
BlockType	BuildingBlock-Types_EnumValue	no	The type of the building block. Can be CustomStringBuildingBlock, FileName-BuildingBlock, ChapterNumberBuilding-Block, PageNumberBuildingBlock, Full-ParagraphBuildingBlock, Paragraph-NumberBuildingBlock, ParagraphText-BuildingBlock, or BookmarkNameBuilding-Block.
CustomText	string	no	The text of the building block. Valid only when the BlockType is CustomStringBuilding-Block.
IncludeDelimiter	boolean	no	If true, include the delimiter character in the building block instances.

**IDML Example 12. BuildingBlock**

```
<BuildingBlock Self="u8bBuildingBlock1" BlockType="ParagraphNumberBuildingBlock"
AppliedCharacterStyle="n" CustomText="$ID/" AppliedDelimiter="$ID/" Include-
Delimiter="false"/>
```

**10.2.9 Hyperlinks**

You can create hyperlinks in an InDesign document so that when you export to PDF, a viewer can click a link to jump to other locations in the same PDF document, to other PDF documents, or to web sites. Hyperlinks are made up of a *hyperlink source* and a *hyperlink destination*, and various display/formatting options.

A hyperlink source is hyperlinked text, a hyperlinked text frame, or a hyperlinked graphics frame. A hyperlink destination is the URL, position in text, or page to which a hyperlink jumps. A source can jump to only one destination, but any number of sources can jump to the same destination.

**Schema Example 20. Hyperlink**

```
Hyperlink_Object = element Hyperlink {
    attribute Self { xsd:string },
    attribute Name { xsd:string }?,
    attribute Source { xsd:string },
    attribute Visible { xsd:boolean }?,
    attribute Highlight { HyperlinkAppearanceHighlight_EnumValue }?,
    attribute Width { HyperlinkAppearanceWidth_EnumValue }?,
    attribute BorderStyle { HyperlinkAppearanceStyle_EnumValue }?,
    attribute Hidden { xsd:boolean }?,
    attribute DestinationUniqueKey { xsd:int }?,
    element Properties {
        element BorderColor { InDesignUIColorType_TypeDef }?&
        element Destination {
            (list_type, element ListItem {
                (string_type, xsd:string ) |
                (long_type, xsd:int ) |
                (bool_type, xsd:boolean )
            }* ) |
            (object_type, xsd:string )
        }?&
        element Label { element KeyValuePair { KeyValuePair_TypeDef }* }?
    }
}?
```

**Table 23. Hyperlink Properties Represented as Attributes**

Name	Type	Req	Description
Name	string	yes	The name of the hyperlink.
Source	string	yes	A reference to the source of the hyperlink (as the value of the Self attribute of the element).
Visible	boolean	no	If true, they hyperlink will be visible in the exported PDF.

Name	Type	Req	Description
Highlight	Hyperlink-AppearanceHighlight_EnumValue	no	The highlight of the hyperlink. Can be None, Invert, Outline or Inset.
Width	Hyperlink-AppearanceWidth_EnumValue		The width of the stroke applied to the hyperlink. Can be Thin, Medium, or Thick.
BorderStyle	Hyperlink-AppearanceStyle_EnumValue	no	The border style of the hyperlink. Can be Solid or Dashed.
Hidden	boolean	no	If true, the hyperlink is hidden in the output PDF.
Destination-UniqueKey	int	no	A unique key identifying the hyperlink destination.

### HyperlinkPageDestination

A hyperlink page destination specifies a page in the document as the destination for a hyperlink.

#### Schema Example 21. HyperlinkPageDestination

```
HyperlinkPageDestination_Object = element HyperlinkPageDestination {
    attribute Self { xsd:string },
    attribute Name { xsd:string }?,
    attribute NameManually { xsd:boolean }?,
    attribute DestinationPage { xsd:string }?,
    attribute ViewSetting { HyperlinkDestinationPageSetting_EnumValue }?,
    attribute ViewPercentage { xsd:double {minInclusive="5" maxInclusive="4000"} }?,
    attribute Hidden { xsd:boolean }?,
    attribute DestinationUniqueKey { xsd:int }?,
    HyperlinkPageDestlement Properties {
        element ViewBounds { UnitRectangleBoundsType_TypeDef }?&
        element Label { element KeyValuePair { KeyValuePair_TypeDef }* }?
    }?
}
```

**Table 24. HyperlinkPageDestination Properties Represented as Attributes**

Name	Type	Req	Description
DestinationPage	string	no	The destination (target) page of the hyperlink.
Destination-UniqueKey	int	no	A unique key identifying the hyperlink destination.
Hidden	boolean	no	If true, the hyperlink is hidden in the PDF.
Name	string	no	The name of the hyperlink page destination.
NameManually	boolean	no	If true, name the hyperlink page destination manually.
ViewPercentage	double	no	The view percentage, if ViewSetting is Fixed.

Name	Type	Req	Description
ViewSetting	Hyperlink-DestinationPage-Setting_EnumValue	no	The view at which to view the content of the hyperlink. Can be Fixed, FitView, Fit-Window, FitWidth, FitHeight, FitVisible, or InheritZoom .

**Table 25. HyperlinkPageDestination Properties Represented as Elements**

Name	Type	Req	Description
ViewBounds	UnitRectangle-BoundsType_TypeDef	yes	The view rectangle, specified in the format [y1, x1, y2, x2]. Note: Valid only when view setting is Fixed.

### ***HyperlinkURLDestination***

A hyperlink page destination specifies a web address as the destination for a hyperlink.

#### **Schema Example 22. HyperlinkURLDestination**

```
HyperlinkURLDestination_Object = element HyperlinkURLDestination {
    attribute Self { xsd:string },
    attribute Name { xsd:string }?,
    attribute DestinationURL { xsd:string }?,
    attribute Hidden { xsd:boolean }?,
    attribute DestinationUniqueKey { xsd:int }?,
    element Properties {
        element Label { element KeyValuePair { KeyValuePair_TypeDef }* }
    }?
}
```

**Table 26. HyperlinkPageDestination Properties Represented as Attributes**

Name	Type	Req	Description
Destination-UniqueKey	int	no	A unique key identifying the hyperlink URL destination.
DestinationURL	string	no	The URL of the hyperlink.
Name	string	no	The name of the hyperlink URL destination.
Hidden	boolean	no	If true, the hyperlink is hidden in the PDF.

### ***HyperlinkExternalPageDestination***

A hyperlink page destination specifies a page outside the document as the destination for a hyperlink.

#### **Schema Example 23. HyperlinkExternalPageDestination**

```
HyperlinkExternalPageDestination_Object = element HyperlinkExternalPageDestination {
```

```

        attribute Self { xsd:string },
        attribute Name { xsd:string }?,
        attribute DocumentPath { xsd:string }?,
        attribute DestinationPageIndex { xsd:int {minInclusive="1"
maxInclusive="9999"} }?,
        attribute ViewSetting { HyperlinkDestinationPageSetting_EnumValue }?,
        attribute ViewPercentage { xsd:double {minInclusive="5" maxInclusive="4000"} }?,
        attribute Hidden { xsd:boolean }?,
        attribute DestinationUniqueKey { xsd:int }?,
        element Properties {
            element ViewBounds { UnitRectangleBoundsType_TypeDef }?&
            element Label { element KeyValuePair { KeyValuePair_TypeDef }* }?
        }?
    }?
}

```

**Table 27. HyperlinkExternalPageDestination Properties Represented as Attributes**

Name	Type	Req	Description
DestinationPage-Index	int	no	The index of the destination page in the target document. Range: 1 to 9999.
DocumentPath	string	no	The path to the target document of the hyperlink.
Destination-UniqueKey	int	no	A unique key identifying the hyperlink URL destination.
Name	string	yes	The name of the hyperlink external page destination.
ViewSetting	Hyperlink-DestinationPage-Setting_EnumValue	no	The view at which to view the content of the hyperlink. Can be Fixed, FitView, Fit-Window, FitWidth, FitHeight, FitVisible, or InheritZoom .
ViewPercentage	double	no	The view percentage, if ViewSetting is Fixed.

**Table 28. HyperlinkExternalPageDestination Properties Represented as Elements**

Name	Type	Req	Description
ViewBounds	UnitRectangle-BoundsType_TypeDef	yes	The view rectangle, specified in the format [y1, x1, y2, x2]. Note: Valid only when view setting is Fixed.

### **HyperlinkPageItemSource**

A hyperlink page item source is a hyperlink associated with a page item.

#### **Schema Example 24. HyperlinkPageItemSource**

```

HyperlinkPageItemSource_Object = element HyperlinkPageItemSource {
    attribute Self { xsd:string },
    attribute Name { xsd:string }?,

```

```

        attribute SourcePageItem { xsd:string },
        attribute Hidden { xsd:boolean }?,
        element Properties {
            element Label { element KeyValuePair { KeyValuePair_TypeDef }* }
        }?
    }
?
```

**Table 29. HyperlinkExternalPageDestination Properties Represented as Attributes**

Name	Type	Req	Description
Name	string	yes	The name of the hyperlink external page destination.
Hidden	boolean	no	If true, the hyperlink page item source will be hidden in the output PDF.
SourcePageItem	string	yes	A reference to a page item as the value of its <i>Self</i> attribute.

#### 10.2.10 Bookmark

InDesign documents can add bookmarks for navigation in PDFs you export.

##### Schema Example 25. Bookmark

```

Bookmark_Object = element Bookmark {
    attribute Self { xsd:string },
    attribute Name { xsd:string }?,
    attribute Destination { xsd:string },
    element Properties {
        element Label { element KeyValuePair { KeyValuePair_TypeDef }* }
    }?
}
?
(
    Bookmark_Object*
)
}
```

**Table 30. Bookmark Properties Represented as Attributes**

Name	Type	Req	Description
Destination	string	no	The destination of the bookmark, as a reference to the <i>Self</i> attribute of a <a href="#"><i>&lt;Page&gt;</i></a> element.
Name	string	yes	The name of the bookmark.

## 10.3 Spreads and Master Spreads

The following sections describe the content and structure of the `<Spread>` and `<MasterSpread>` elements in IDML. In an IDML package, `<Spread>` elements are stored in the `Spread` files stored within the Spreads folder (which are named according to the rules described in “IDML Component Names”). For the remainder of this section, we will refer to these files as `Spread.xml`.

The `<MasterSpread>` elements are very similar to the `<Spread>` elements in each `Spread.xml` file. This distinction is purely for organizational purposes; the XML structures of the two elements are fundamentally the same, and most of the information in this section can be applied to either type of file. In an IDML package, `<MasterSpread>` elements are stored in `MasterSpreads` folder.

The pages in an InDesign document are grouped into spreads and master spreads. Master spreads differ from document page spreads in that they can be applied to other pages, and are typically used for repeating layout elements, such as page numbers or running headers. Spreads and master spreads contain pages, and all page items that can appear in an InDesign document. In addition, spreads contain a number of spread-level preferences, such as flattener settings, the spread binding location, and the display state of master page items on the spread.

### **Schema Example 26. Spread Schema**

```
Spread_Object = element Spread {
    attribute Self { xsd:string },
    attribute PageTransitionType { PageTransitionTypeOptions_EnumValue }?,
    attribute PageTransitionDirection { PageTransitionDirectionOptions_EnumValue }?,
    attribute PageTransitionDuration { PageTransitionDurationOptions_EnumValue }?,
    attribute FlattenerOverride { SpreadFlattenerLevel_EnumValue }?,
    attribute ShowMasterItems { xsd:boolean }?,
    attribute PageCount { xsd:int }?,
    attribute BindingLocation { xsd:int }?,
    attribute AllowPageShuffle { xsd:boolean }?,
    attribute ItemTransform { TransformationMatrixType_TypeDef }?,
    attribute AppliedMaster { xsd:string }?,
    element Properties {
        element Label { element KeyValuePair { KeyValuePair_TypeDef }* }
    }?
}
?
'
(
    FlattenerPreference_Object?&
    Page_Object*&
    Oval_Object*&
    Rectangle_Object*&
    GraphicLine_Object*&
    TextFrame_Object*&
    Polygon_Object*&
    Group_Object*&
    EPSText_Object*&
    FormField_Object*&
    Button_Object*
)
}
```

Note that the `<MasterSpread>` element differs from the `<Spread>` element only in that it can contain `Name`, `NamePrefix`, `BaseName`, and `OverriddenPageItemProps` attributes—only master spreads have these properties (refer to the InDesign documentation for more on master spread options). `<MasterSpread>` elements lack the `BindingLocation`, `AllowPageShuffle`, and `FlattenerOverride` attributes, as these elements only apply to document spreads (again, for more on the differences between master spreads and document spreads, refer to the InDesign documentation). Apart from these minor differences, the two elements—and the child elements that they can contain, such as `<TextFrame>`, `<Rectangle>`, or `<Group>` elements—are the same.

#### Schema Example 27. Master Spread Schema

```
MasterSpread_Object = element MasterSpread {
    attribute Self { xsd:string },
    attribute ItemTransform { TransformationMatrixType_TypeDef }?,
    attribute Name { xsd:string }?,
    attribute NamePrefix { xsd:string }?,
    attribute BaseName { xsd:string }?,
    attribute ShowMasterItems { xsd:boolean }?,
    attribute PageCount { xsd:int }?,
    attribute AppliedMaster { xsd:string }?,
    attribute OverriddenPageItemProps { list { xsd:int * } }?,
    element Properties {
        element Label { element KeyValuePair { KeyValuePair_TypeDef }*
            }?
    }
    ?
    ,
    (
        Page_Object*&
        Oval_Object*&
        Rectangle_Object*&
        GraphicLine_Object*&
        TextFrame_Object*&
        Polygon_Object*&
        Group_Object*&
        EPSText_Object*&
        FormField_Object*&
        Button_Object*
    )
}
```

Most of the properties of a spread or master spread are represented by attributes. The following table shows the attributes that can appear in a `<Spread>` or `<MasterSpread>` element.

**Table 31. Spread/MasterSpread Attributes**

Attribute	Data Type	Req	Description
FlattenerOverride	SpreadFlattener_Level_EnumValue	no	The transparency flattener preferences override for the spread. ( <code>&lt;Spread&gt;</code> only). Can be Default, None, or Custom.

Attribute	Data Type	Req	Description
ItemTransform	Transformation-Matrix	no	A transformation matrix applied to the spread. In InDesign, spreads can be rotated, but not translated, scaled, or skewed, so this matrix will only specify rotation, and that only in 90-degree increments.
ShowMasterItems	Boolean	no	If true, displays master page items on document pages in the spread.
PageCount	int	no	The number of pages in the spread.
BindingLocation	int	no	The location of the binding edge of the spread. (<Spread> only)
AllowPageShuffle	Boolean	no	If true, allows the pages of the spread to move to other spreads during repagination. If false, keeps the pages of the spread together, regardless of repagination. (<Spread> only)
AppliedMaster	string	no	A reference to the unique ID (the value of the Self attribute) of the master spread applied to the <Spread> or <MasterSpread>.
Name	string	no	The name of the master spread. (<MasterSpread> only)
NamePrefix	string	no	The name prefix of the master spread. (<MasterSpread> only)
BaseName	string	no	The base name of the master spread. (<MasterSpread> only)

### Minimal Spread Example

The <Spread> element is simply a container for other elements that can appear on a spread. In the following example, we have omitted the details of the `Page` and `Rectangle` elements. For more information on those objects, refer to the corresponding reference sections.

#### IDML Example 13. Minimal Spread

```
<Spread Self="ub5" PageCount="1" AppliedMaster="ubc">
  <Page>...</Page>
  <Rectangle>...</Rectangle>
</Spread>
```

#### 10.3.1 Page Items

Page items are the rectangles, ellipses, graphic lines, polygons, text frames, groups, and buttons that can appear on an InDesign page. In IDML, page items are collected on spreads as child elements of the <Spread> element.

Page items appear in elements named after their specific class: `Rectangle`, `Oval`, `GraphicLine`, `Polygon`, `TextFrame`, `Group`, and `Button`. There is no specific page item for a graphic frame; any rectangle, ellipse, graphic line, or polygon can contain an imported graphic. The only difference between the `Rectangle`, `Oval`, `GraphicLine`, and `Polygon` elements is in the number and arrangement of the `PathPoint` elements contained in their `PathGeometry` elements.

From the point of view of the InDesign scripting model, page items frequently change their type as you alter their content type or geometry. Add a point to a rectangle, and it becomes a polygon; change the content type of a polygon to text type, and it becomes a text frame. IDML works in a similar fashion—you can add multiple paths to a `<Rectangle>` element, for example, and InDesign will still be able to open and interpret the element.

The schemas for `Rectangle`, `Oval`, `GraphicLine`, `Polygon`, and `TextFrame` elements are almost identical. The following example is for the `Rectangle` element; variations for other object types will be discussed later in this section.

#### **Schema Example 28. Page Item (Rectangle, Oval, GraphicLine, or Polygon)**

```
Rectangle_Object = element Rectangle {
    attribute Self { xsd:string },
    attribute StoryTitle { xsd:string }?,
    attribute ContentType { ContentType_EnumValue }?,
    attribute AllowOverrides { xsd:boolean }?,
    attribute FillColor { xsd:string }?,
    attribute FillTint { xsd:double }?,
    attribute OverprintFill { xsd:boolean }?,
    attribute StrokeWeight { xsd:double }?,
    attribute MiterLimit { xsd:double {minInclusive="1" maxInclusive="500"} }?,
    attribute EndCap { EndCap_EnumValue }?,
    attribute EndJoin { EndJoin_EnumValue }?,
    attribute StrokeType { xsd:string }?,
    attribute StrokeCornerAdjustment { StrokeCornerAdjustment_EnumValue }?,
    attribute StrokeDashAndGap { list { xsd:double * } }?,
    attribute LeftLineEnd { ArrowHead_EnumValue }?,
    attribute RightLineEnd { ArrowHead_EnumValue }?,
    attribute StrokeColor { xsd:string }?,
    attribute StrokeTint { xsd:double }?,
    attribute CornerRadius { xsd:double }?,
    attribute GradientFillStart { UnitPointType_TypeDef }?,
    attribute GradientFillLength { xsd:double }?,
    attribute GradientFillAngle { xsd:double }?,
    attribute GradientStrokeStart { UnitPointType_TypeDef }?,
    attribute GradientStrokeLength { xsd:double }?,
    attribute GradientStrokeAngle { xsd:double }?,
    attribute OverprintStroke { xsd:boolean }?,
    attribute GapColor { xsd:string }?,
    attribute GapTint { xsd:double }?,
    attribute OverprintGap { xsd:boolean }?,
    attribute StrokeAlignment { StrokeAlignment_EnumValue }?,
    attribute Nonprinting { xsd:boolean }?,
    attribute ItemLayer { xsd:string }?,
    attribute Locked { xsd:boolean }?,
    attribute LocalDisplaySetting { DisplaySettingOptions_EnumValue }?,
    attribute GradientFillHiliteLength { xsd:double }?,
    attribute GradientFillHiliteAngle { xsd:double }?,
    attribute GradientStrokeHiliteLength { xsd:double }?,
    attribute GradientStrokeHiliteAngle { xsd:double }?,
    attribute AppliedObjectStyle { xsd:string }?,
    attribute CornerOption { CornerOptions_EnumValue }?,
    attribute ItemTransform { TransformationMatrixType_TypeDef }?,
    element Properties {
```

```

element PathGeometry { element GeometryPathType { GeometryPathType_TypeDef }*
} ?&
element Label { element KeyValuePair { KeyValuePair_TypeDef }*
} ?
}
?
'
(
TextPath_Object*,
(TransparencySetting_Object?&
StrokeTransparencySetting_Object?&
FillTransparencySetting_Object?&
ContentTransparencySetting_Object?&
AnchoredObjectSetting_Object?&
TextWrapPreference_Object?&
InCopyExportOption_Object?&
FrameFittingOption_Object?),  

(Sound_Object*&
Movie_Object*&
Link_Object*&
FormField_Object*&
Button_Object*&
Oval_Object*&
Rectangle_Object*&
GraphicLine_Object*&
TextFrame_Object*&
Graphic_Object*&
Image_Object*&
EPS_Object*&
WMF_Object*&
PICT_Object*&
PDF_Object*&
Polygon_Object*&
Group_Object*&
EPSText_Object*&
ImportedPage_Object*)
)
}

```

**Table 32. Common Page Item Properties Represented as Attributes**

Name	Type	Req	Description
AllowOverrides	boolean	no	If true, the master page page item allows overrides.
AppliedObject-Style	string	no	The object style applied to the page item.
ContentType	ContentType_Enum-Value	no	The type of content that a frame can contain. Can be Unassigned (No content type assigned), GraphicType (The frame is a graphics frame), or TextType (The frame is a text frame).

Name	Type	Req	Description
CornerOption	CornerOptions_EnumValue	no	The shape to apply to corner points in a path. Note: corner option differs from end join in that you can set a radius for a corner option, whereas the rounded or beveled effect of an end join depends upon the stroke weight. Can be None (No corner option), RoundedCorner (Rounded corner option), InverseRoundedCorner (Inverted rounded corner option), InsetCorner (Inset corner option), BevelCorner (Beveled corner option), or FancyCorner (Fancy corner option).
CornerRadius	double	no	The radius in measurement units of the corner effect.
EndCap	EndCap_EnumValue	no	The end shape of an open path. Can be ButtEndCap (A squared end that stops at the path's endpoint), RoundEndCap (A semicircular end that extends beyond the endpoint by half the stroke-width), or ProjectingEndCap (A squared end that extends beyond the endpoint by half the stroke-width).
EndJoin	EndJoin_EnumValue	no	The corner join applied to the page item. Can be MiterEndJoin (Miter end join), RoundEndJoin (Rounded end join), or BevelEndJoin (Beveled end join).
FillColor	string	no	The swatch (color, gradient, tint, or mixed ink) applied to the to fill the page item. .
FillTint	double	no	The percent of tint to use in the page item's fill color. (To specify a tint percent, use a number in the range of 0 to 100; to use the inherited or overridden value, use -1.)
GapColor	string	no	The swatch (color, gradient, tint, or mixed ink) applied to the gap of a dashed, dotted, or striped stroke. For information, see stroke type.
GapTint	double	no	The tint as a percentage of the gap color. (To specify a tint percent, use a number in the range of 0 to 100; to use the inherited or overridden value, use -1.)
GradientFillAngle	double	no	The angle of a linear gradient applied to the fill of the page item. (Range: -180 to 180)
GradientFill-HiliteAngle	double	no	The angle of the gradient fill highlight.
GradientFill-HiliteLength	double	no	The length of the gradient fill highlight.
GradientFill-Length	double	no	The length (for a linear gradient) or radius (for a radial gradient) applied to the fill of the page item.
GradientFillStart	UnitPointType_TypeDef	no	The starting point (in page coordinates) of a gradient applied to the fill of the page item, in the format [x, y].

Name	Type	Req	Description
GradientStroke-Angle	double	no	The angle of a linear gradient applied to the stroke of the page item. (Range: -180 to 180)
GradientStroke-HiliteAngle	double	no	The angle of the gradient stroke highlight.
GradientStroke-HiliteLength	double	no	The length of the gradient stroke highlight.
GradientStroke-Length	double	no	The length (for a linear gradient) or radius (for a radial gradient) applied to the stroke of the page item.
GradientStroke-Start	UnitPointType_TypeDef	no	The starting point (in page coordinates) of a gradient applied to the stroke of the page item, in the format [x, y].
ItemLayer	string	no	The layer that the page item is on.
ItemTransform	Transformation-MatrixType_TypeDef	no	The transformation matrix applied to the page item.
LeftLineEnd	ArrowHead_Enum-Value	no	The arrowhead applied to the start of the path. Can be None (None), SimpleArrowHead (An arrow head formed by two slanting lines whose intersection forms a 45-degree angle and whose stroke weight is the same as the path's stroke), SimpleWideArrowHead (An arrow head formed by two slanting lines whose intersection forms a 90-degree angle and whose stroke weight is the same as the path's stroke), TriangleArrowHead (A solid triangle arrow head whose point describes a 45-degree angle), TriangleWideArrowHead (A solid triangle arrow head whose point describes a 90-degree angle), BarbedArrowHead (A solid arrow head whose pierced end bows sharply toward the point and whose point describes a 45-degree angle), CurvedArrowHead (A solid arrow head whose pierced end concaves toward the point and whose point describes a 45-degree angle), CircleArrowHead (A hollow circle whose outline is the same weight as the stroke The circle's diameter is 5 times the stroke width), CircleSolidArrowHead (A solid circle whose diameter is 5 times the stroke width), SquareArrowHead (A hollow square set perpendicular to the path, whose outline is the same weight as the stroke The length of one side of the square is 5 times the stroke width), SquareSolidArrowHead (A solid square set perpendicular to the end of the path The length of one side of the square is 5 times the stroke width), or BarArrowHead (A vertical bar bisected by the stroke, which meets the stroke at a right angle and is the same weight as the stroke The bar's length is 45 times the stroke width).

Name	Type	Req	Description
LocalDisplaySetting	DisplaySettingOptions_EnumValue	no	Display performance options for the page item. Can be HighQuality (Slower performance; displays high-resolution graphics and high-quality transparencies and turns on anti-aliasing), Typical (Moderate performance speed; displays proxy graphics and low-quality transparencies and turns on anti-aliasing), Optimized (Best performance; grays out graphics and turns off transparency and anti-aliasing), or Default (Uses the container object's default display performance preferences setting. For information, see default display settings).
Locked	boolean	no	If true, the page item is locked.
MiterLimit	double	no	The limit of the ratio of stroke width to miter length before a miter (pointed) join becomes a bevel (squared-off) join.
Nonprinting	boolean	no	If true, the page item does not print.
OverprintFill	boolean	no	If true, the page item's fill color overprints any underlying objects. If false, the fill color knocks out the underlying colors.
OverprintGap	boolean	no	If true, the gap color overprints any underlying colors. If false, the gap color knocks out the underlying colors.
OverprintStroke	boolean	no	If true, the page item's stroke color overprints any underlying objects. If false, the stroke color knocks out the underlying colors.

Name	Type	Req	Description
RightLineEnd	ArrowHead_EnumValue	no	The arrowhead applied to the end of the path. Can be None (None), SimpleArrowHead (An arrow head formed by two slanting lines whose intersection forms a 45-degree angle and whose stroke weight is the same as the path's stroke), SimpleWideArrowHead (An arrow head formed by two slanting lines whose intersection forms a 90-degree angle and whose stroke weight is the same as the path's stroke), TriangleArrowHead (A solid triangle arrow head whose point describes a 45-degree angle), TriangleWideArrowHead (A solid triangle arrow head whose point describes a 90-degree angle), BarbedArrowHead (A solid arrow head whose pierced end bows sharply toward the point and whose point describes a 45-degree angle), CurvedArrowHead (A solid arrow head whose pierced end concaves toward the point and whose point describes a 45-degree angle), CircleArrowHead (A hollow circle whose outline is the same weight as the stroke The circle's diameter is 5 times the stroke width), CircleSolidArrowHead (A solid circle whose diameter is 5 times the stroke width), SquareArrowHead (A hollow square set perpendicular to the path, whose outline is the same weight as the stroke The length of one side of the square is 5 times the stroke width), SquareSolidArrowHead (A solid square set perpendicular to the end of the path The length of one side of the square is 5 times the stroke width), or BarArrowHead (A vertical bar bisected by the stroke, which meets the stroke at a right angle and is the same weight as the stroke The bar's length is 45 times the stroke width).
StoryTitle	string	no	The title of the story associated with this page item, if any.
StrokeAlignment	StrokeAlignment_EnumValue	no	The stroke alignment applied to the page item. Can be CenterAlignment (The stroke straddles the path), InsideAlignment (The stroke is inside the path), or OutsideAlignment (The stroke is outside the path, like a picture frame).
StrokeColor	string	no	The swatch (color, gradient, tint, or mixed ink) applied to the to stroke of the page item.

Name	Type	Req	Description
StrokeCorner-Adjustment	StrokeCorner-Adjustment_Enum-Value	no	The corner adjustment applied to the page item. Can be None (No adjustment), Dashes (Changes the length of dashes so that dashes always occur at path ends and corners; maintains set gap length Note: Can cause dashes to be different lengths on shapes whose sides are of different lengths, such as rectangles), Gaps (Changes the length of gaps so that dashes or dots always occur at ends and corners; maintains dash length or dot diameter Note: Can cause gaps to be different lengths on shapes whose sides are of different lengths, such as rectangles), or DashesAndGaps (Adjusts both dashes and gaps to cover corners and end points Note: Causes dash and gap sizes to be consistent on all sides of the shape).
StrokeDashAndGap	string	no	The dash and gap measurements that define the pattern of a custom dashed line. Define up to six values (in points) in the format [dash1, gap1, dash2, gap2, dash3, gap3].
StrokeTint	double	no	The percent of tint to use in object's stroke color. (To specify a tint percent, use a number in the range of 0 to 100; to use the inherited or overridden value, use -1.)
StrokeType	string	no	The name of the stroke style to apply.
StrokeWeight	double	no	The weight (in points) to apply to the page item's stroke.

**Table 33. Common Page Item Properties Represented by Elements**

Name	Type	Req	Description
PathGeometry	Array of GeometryPathType elements	yes	An element containing the geometry of the page item. For more information, refer to “Geometry in IDML.”

### 10.3.2 Nested Objects and IDML Structure

Page items (and other objects, such as imported graphics) in an InDesign layout are frequently nested inside other page items. This nesting is explicit in the IDML structure. A `<Document>` element, for example contains `<Spread>` elements, which, in turn, contain `<Rectangle>` elements or other page item elements. An imported image is stored as a child XML element of the containing page item element. Objects inside a group appear as child elements of the `<Group>` element.

Only the top-level page items of a spread appear as child elements of the `<Spread>` element; all nested objects appear as child elements of their respective container elements.

### 10.3.3 Geometry in IDML

When you are editing or creating an IDML file, it's likely that you will need to draw a page item at a specific location on a page. You're probably already familiar with the geometry of the paths in page items—that they're made up of points positioned at pairs of x and y coordinates—but it's likely that you'll have a few questions in mind. What units of measurement should you use? Where is the origin of the coordinate system? Are the coordinates used in page items relative to the page or to the spread? In this section, we'll attempt to answer these questions.

The first thing you need to know is that the coordinates used in IDML are not the same as the coordinates you see in the InDesign user interface or in the InDesign scripting object model (they are, however, the same coordinates as used in the InDesign C++ model). At the same time, IDML offers a robust and consistent set of elements and attributes for specifying object locations—anything that can be drawn in the InDesign user interface can be achieved in IDML.

This section will focus on how geometry is expressed in IDML; the general topic of the geometry of page items and spreads in an InDesign document is far beyond the scope of this specification. For more information, refer to the “Layout Fundamentals” chapter of the *InDesign SDK Programming Guide*. For more on coordinate systems and transformation matrices, refer to the *PostScript Language Reference Manual*.

**Note:** In this section, we'll refer to a generic `<PageItem>` element. There are no `<PageItem>` elements in IDML; this element is provided for explanatory purposes only—it could be a `<Rectangle>`, `<Oval>`, `<GraphicLine>`, `<TextFrame>`, or `<Polygon>` element.

#### ***Coordinates, Transformation Matrices, and the IDML Element Hierarchy***

As we mentioned in “Nested Objects and IDML Structure,” above, elements in IDML are arranged in a specific hierarchy. A document contains spreads; spreads contain page items; and page items can contain other page items. The location of a page item on a spread depends on its position in the hierarchy. The relationships between the elements are defined by the following:

- A document has a base coordinate system (see “Pasteboard Coordinates,” below).
- Each spread or master spread has its own coordinate system (see “Spread Coordinates,” below).
- Each page item element has its own coordinate system (see “Page Item Geometry,” below).
- The relationship of an element's coordinates to the coordinate system of its parent is determined by the *transformation matrix* applied to the element (see “Transformations,” below).

In the paragraphs above, we talk about each element having its own coordinate system. In the rest of this section, we'll refer to the coordinate system of an element as the element's *inner coordinates*.

#### ***Pasteboard Coordinates***

It's natural to think of the coordinates in an InDesign document in terms of the rulers you see in the layout window—*ruler coordinates*. In ruler coordinates, a page item can actually straddle two different sets of rulers. This, among other reasons, makes ruler coordinates inconvenient for developers.

Instead, IDML geometry is built on the *pasteboard coordinate system*. This isn't the pasteboard you see surrounding a spread in the user interface; instead, it's a global coordinate system that underlies all of the spreads in an InDesign document. The pasteboard coordinate system starts at zero, above the first page in the document, and encompasses all of the spreads in the document. (The pasteboard coordinate system does not have a negative extent.)

- The units used by the pasteboard coordinate system are points, defined as 72 units per inch. Changing the definition of points in the InDesign user interface has no effect on the definition of points used in IDML.
- Increasing a vertical coordinate (y) moves the specified location *down* in pasteboard coordinates. This is the same as ruler coordinates, but is “flipped” relative to the x and y axes of traditional geometry (i.e., what you learned in geometry and trigonometry classes), PostScript, and PDF.
- Increasing the horizontal coordinate (x) moves the specified location to the right in pasteboard coordinates (the same as in ruler coordinates and in traditional geometry).

### **Transformations**

InDesign documents consist of (among other things) page items on spreads. These page items have an intrinsic shape that consists of one or more paths. The shape of the paths are completely determined by a sequence of coordinate pairs that specify either a point the path is to pass through (the anchor point) or the curvature of the line segments connected to the anchor point (control points). The shape of page items is completely determined by a collection of x and y coordinates.

If you were to draw the points that make up the path of a page item on a standard sheet of graph paper marked at 72 units per inch, the path would, in general, appear different than it does when printed from InDesign. The coordinate system represented by this fictitious graph paper is the item’s inner coordinate system. Each page item has a distinct inner coordinate system—its own sheet of graph paper. The path points that make up the shape of the page item are always described in inner coordinates.

A transformation matrix specifies the relationship between the coordinate system of the page element (the graph paper we talked about earlier) and the coordinate system of its parent element (another sheet of graph paper). By modifying a transformation matrix, objects can be scaled, rotated, moved, or transformed in other ways. For example, in InDesign, when you rotate a page item by 30 degrees, it is the item’s transformation matrix that is affected, and not the coordinates of the item’s individual path points. To return to our metaphor we’re rotating the the graph paper the element is drawn on, relative to the graph paper of its parent element.

The relationship between the pasteboard coordinates of a `<Document>` element and the coordinate system of a `<Spread>` element is determined by the `ItemTransform` attribute of the `<Spread>` element.

In IDML, page item elements are child elements of a `<Spread>` or `<MasterSpread>` element, so you’ll be more concerned with the coordinate system of those elements than with the pasteboard coordinate system. (Note that this is unlike the InDesign C++ model, where page items are collected on layers.)

You can think of the sequence of transformations applied to an element as a “stack” of transformations from the pasteboard coordinate system of the `<Document>` element, through the inner coordinate system of the `<Spread>` element, and to the inner coordinate system of a `<PageItem>` element contained as a child of the `<Spread>` element. To position the element within its parent element, InDesign transforms the inner coordinates of the child element into the coordinate system of its parent element.

A transformation between two coordinate systems in two dimensional space is represented by a 3-by-3 transformation matrix written as:

$$\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{bmatrix}$$

Because a transformation matrix has only six elements that can be changed, it is specified in IDML using the standard form `[a b c d e f]`. This transformation matrix is stored in `ItemTransform` attribute of a `<PageItem>` element, or in the `ItemTransform` attribute of a `<Spread>` or `<MasterSpread>` element.

To transform the inner coordinates of an element into the inner coordinates of its parent element, InDesign performs the following calculation (where  $T$  is the transformation matrix of the child element):

$$\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{bmatrix} = T_{\text{Scale}} \times T_{\text{Shear}} \times T_{\text{Rotation}} \times T_{\text{Translate}}$$

If the transformation matrix applied to an element is the identity matrix (`[1 0 0 1 0 0]`), then the inner coordinate system of the element and the inner coordinate system of its parent coincide.

The following list shows the transformation matrices for performing the most common transformations.

- Translations are specified as `[1 0 0 1 tx ty]`, where  $t_x$  and  $t_y$  are the horizontal and vertical distances to move the object, relative to the center of the pasteboard.
- Scaling is obtained using the matrix `[sx 0 0 sy 0 0]`. This scales the object in a given element by  $s_x$  (horizontal scaling factor) and  $s_y$  (vertical scaling factor), relative to the pasteboard.
- Rotations are produced by the matrix `[\cos(\theta) \sin(\theta) -\sin(\theta) \cos(\theta) 0 0]`, which has the effect of rotating the object counterclockwise by the angle  $\theta$ , relative to the pasteboard.
- Skewing is specified by the matrix `[1 \tan(\alpha) \tan(\beta) 1 0 0]`, which skews the x (horizontal) axis of the object by the angle  $\alpha$  and the y (vertical) axis of the object by the angle  $\beta$ , relative to the x and y axis of the pasteboard.

Transformations are applied in the following order: scale, shear, rotate, translate.

### Spread Coordinates

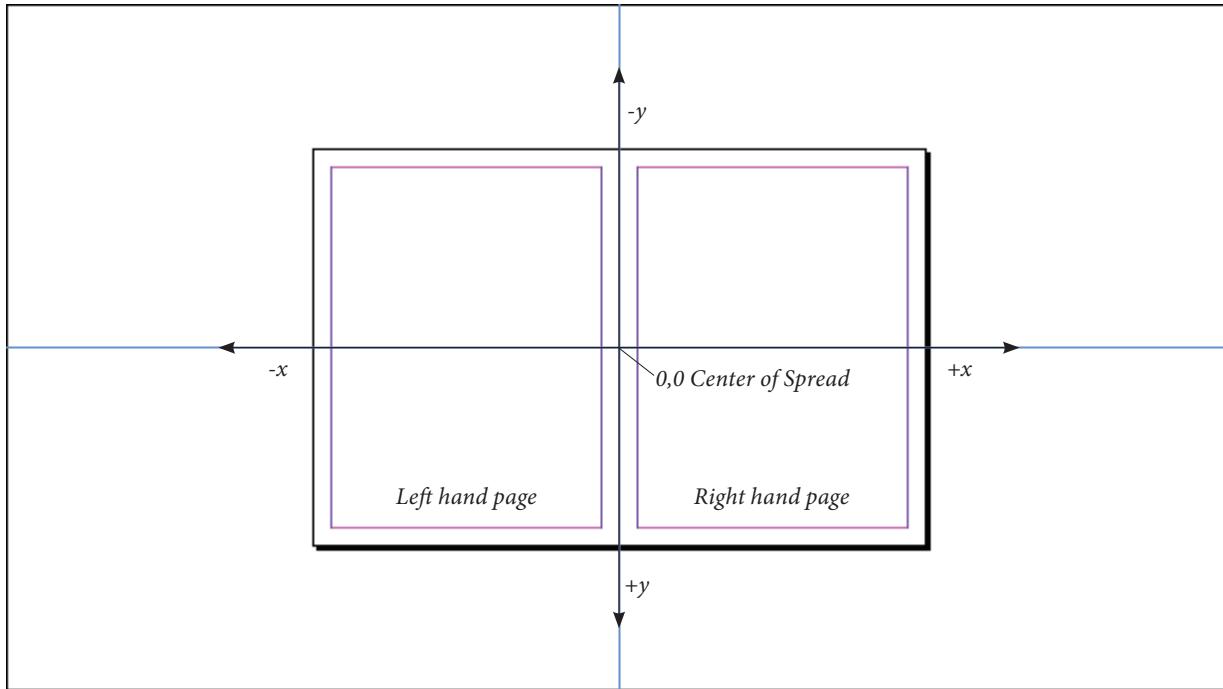
The `ItemTransform` attribute of a `<Spread>` or `<MasterSpread>` element defines the relationship of the spread to the pasteboard coordinate system. In general, only the y translation and rotation of the transformation matrix defined in the `ItemTransform` attribute will be applied to the spread. Since pages cannot be rotated to arbitrary angles in InDesign's user interface, only 90-degree rotations are allowed:

<code>ItemTransform="1 0 0 1 0 0"</code>	No spread rotation
<code>ItemTransform="0 1 -1 0 0 0"</code>	90 degree clockwise spread rotation
<code>ItemTransform="0 -1 1 0 0 0"</code>	90 degree counterclockwise spread rotation
<code>ItemTransform="-1 0 -0 -1 0 0"</code>	180 degree spread rotation

**Note:** Spreads cannot be scaled or skewed using the transformation matrix in the `ItemTransform` attribute.

The origin of the spread coordinate system is located at center of the spread. The left edge of the first right hand page in the spread aligns with the horizontal center of the spread; the right edge of the first left hand page in the spread appears to the left. The vertical centers of the pages align with the vertical center of the spread. Each spread has its own coordinate system origin.

Figure 7. IDML Spread Coordinate System



**Note:** The horizontal center of the spread does not correspond to the binding edge of the spread, and is not marked in any way in the InDesign user interface. Left hand pages are always added to the left of the horizontal center of the spread; right hand pages are always added to the right.

### **Page Item Geometry**

The location of a `<Spread>` element, relative to the pasteboard coordinate system, is determined by its `ItemTransform` attribute. The `<Spread>` element uses the same units of measurement as the pasteboard coordinate system: points. The coordinate system used by a `<PageItem>` element, however, can be entirely arbitrary (though the coordinates are always points in IDML exported form InDesign). The relationship of the inner coordinates of the child element to the coordinate system of the `<Spread>` element (or other parent element) is defined by the `ItemTransform` attribute of the child element. What may appear to be absolute coordinates in a child element are not absolute with respect to the spread coordinate system, because the values of the `ItemTransform` attribute may cause the child element to move, shrink, or expand, relative to the spread.

Most of the time, when you export IDML from InDesign, `<PageItem>` elements that are direct children of a `<Spread>` element (i.e., that are not contained by other page items, groups, or anchored in text) will have a coordinate system that coincides with the spread coordinates.

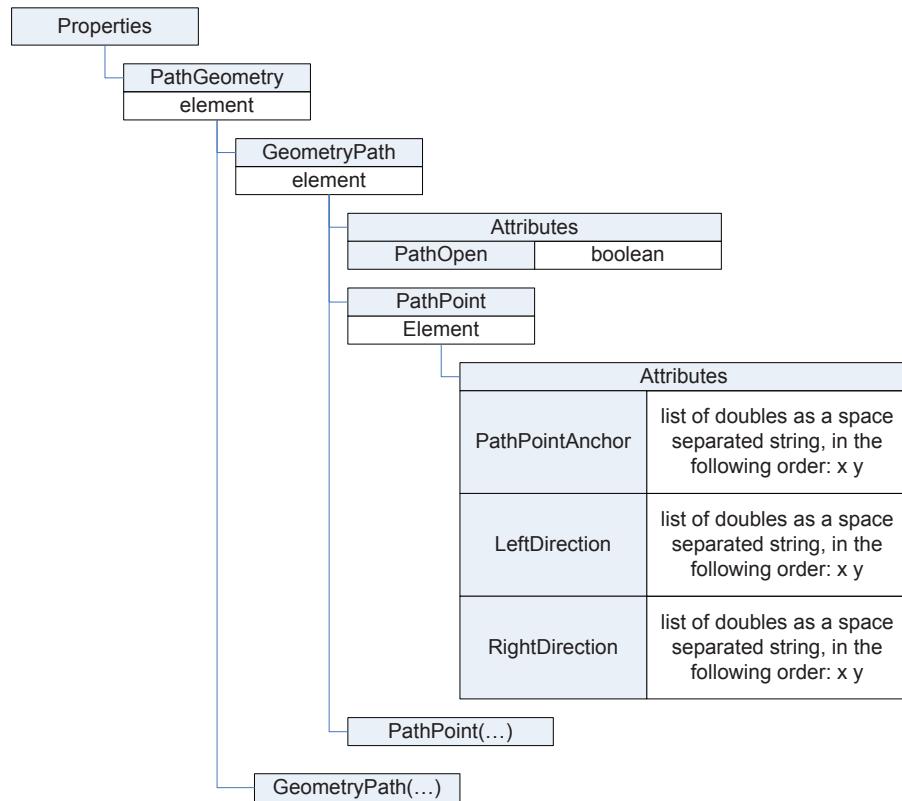
The shape of each page item is defined by the contents of the `<PathGeometry>` element (which is inside the `<Properties>` element). The `<PathGeometry>` element contains:

- One or more `<GeometryPathType>` elements representing the paths that make up the page item. Each `<GeometryPathType>` element contains a `<PathPointArray>` element, which contains a list of `<PathPoint>` elements.
- Each `<PathPoint>` element contains attributes defining three coordinate pairs that define the location of the path point and the curvature of the line segments connecting the path point to the other points in the path. These attributes are named `PathPointAnchor` (the location of the point), `LeftDirection` (incoming Bezier control point), and `RightDirection` (outgoing Bezier control point). Each of these attributes take the form `x y` (where `x` is the horizontal location of the point and `y` is the vertical location, expressed in the inner coordinate system of the page item).
- The `PathOpen` attribute of the `<GeometryPathType>` element specifies whether the path is an open path or a closed path. If `IsOpen` is `true`, the path is an open path; if it's `false`, the path is a closed path. An InDesign path does not need to be closed to display a fill that has been applied to it.

**Note:** InDesign paths use the Nonzero Winding Number rule (not the Even-Odd rule) to determine the fill areas of self-intersecting paths. For more on the Nonzero Winding Number rule, refer to the *PostScript Language Reference Manual*.

While the shape of the page item is defined by the contents of the `<GeometryPathType>` element, the actual location of the page item in the spread depends on the contents of the `ItemTransform` attribute of the page item. This transformation matrix defines the relationship between the inner coordinate system of the page item and the inner coordinate system of its parent.

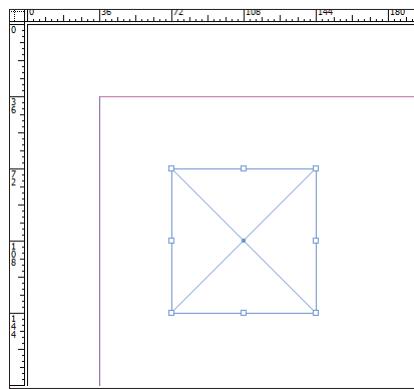
Figure 8. PathGeometry Element



### Geometry Example

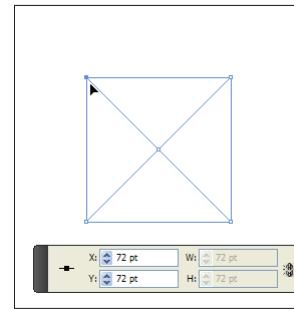
To better demonstrate the effect of transformation and object nesting in IDML, let's follow a point on a simple rectangle as it is transformed, nested inside another object, and then has another transformation applied to the parent. In this example, we'll create a rectangle with a geometric bounds defined by putting the upper left corner of the rectangle at (72, 72) and the lower right corner at (144, 144). Both corners are specified in the user interface in points, according to ruler coordinates, and assuming that the zero point is at the upper left corner of the page.

Figure 9. Example Rectangle



In the following figure, we've selected the point using the Direct Selection tool; the location of the point in ruler coordinates is shown in the X and Y fields of the Control panel.

*Figure 10. Selected Point*



The rectangle in the above example appears in IDML as shown in the following example. The point we're following is highlighted in red.

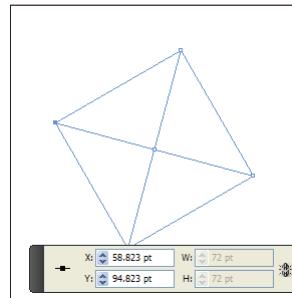
#### IDML Example 14. Untransformed Rectangle

```
<Rectangle Self="ud0" ItemTransform="1 0 0 1 0 0">
  <Properties>
    <PathGeometry>
      <GeometryPathType PathOpen="false">
        <PathPointArray>
          <PathPointType Anchor="72 -324" LeftDirection="72 -324"
RightDirection="72 -324"/>
          <PathPointType Anchor="72 -252" LeftDirection="72 -252"
RightDirection="72 -252"/>
          <PathPointType Anchor="144 -252" LeftDirection="144 -252"
RightDirection="144 -252"/>
          <PathPointType Anchor="144 -324" LeftDirection="144 -324"
RightDirection="144 -324"/>
        </PathPointArray>
      </GeometryPathType>
    </PathGeometry>
  </Properties>
</Rectangle>
```

The parent of the `<Rectangle>` element is a `<Spread>` element. Since the rectangle has not been transformed, the coordinates of the rectangle are therefore coincident with the same coordinates in the spread. Again, the vertical coordinate origin for the spread is not the same as that shown on the rulers in the user interface, which is why the `y` coordinates in the IDML differ from what you see in the fields of the Control panel.

Next, we'll rotate the triangle by 30 degrees around its center point. When we do this, the location of the point in ruer coordinates changes, as shown in the following figure.

Figure 11. Rotation Changes the Point Location Relative to the Rulers



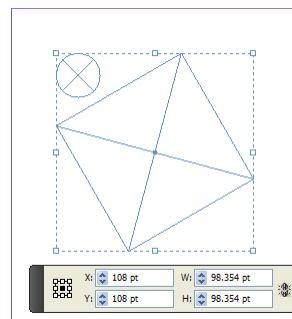
In IDML, however, the coordinates of the point do not change. The transformation is accomplished using the `ItemTransform` attribute of the `<Rectangle>` element, as shown in the example below (again, we've used red to highlight the significant parts of the IDML).

#### IDML Example 15. Rotation Changes the ItemTransform Attribute in IDML

```
<Rectangle Self="ud0" ItemTransform="0.8660254037844387 -0.50000000000000001
0.50000000000000001 0.8660254037844387 158.46925639128065 15.415316289918337">
<Properties>
  <PathGeometry>
    <GeometryPathType PathOpen="false">
      <PathPointArray>
        <PathPointType Anchor="72 -324" LeftDirection="72 -324"
RightDirection="72 -324"/>
        <PathPointType Anchor="72 -252" LeftDirection="72 -252"
RightDirection="72 -252"/>
        <PathPointType Anchor="144 -252" LeftDirection="144 -252"
RightDirection="144 -252"/>
        <PathPointType Anchor="144 -324" LeftDirection="144 -324"
RightDirection="144 -324"/>
      </PathPointArray>
    </GeometryPathType>
  </PathGeometry>
</Properties>
</Rectangle>
```

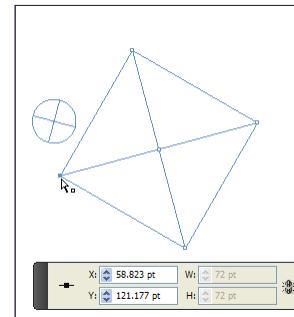
Next, we'll add the rectangle to a group.

Figure 12. Example Rectangle Grouped with an Ellipse



Next, we'll rotate the group by 30 degrees around its center point. As you can see from the X and Y fields in the Control panel, the location of the point changes relative to the ruler coordinates in the user interface.

*Figure 13. The Example Rectangle Inside a Rotated Group*



In IDML, you'll see that the `<Rectangle>` element now appears within a `<Group>` element. The location of the point inside the `<Rectangle>` element remains the same; the transformations are applied to the rectangle through the `ItemTransform` attributes of both the `<Group>` element and the `<Rectangle>` element (again, we've highlighted the significant IDML fragments with red). Because we rotated the group by the same angle as the rectangle, the `ItemTransform` values are the same (showing a 30 degree rotation), but the important point is that the `ItemTransform` and `PathGeometry` sections of the rectangle do not change when the group was transformed.

#### IDML Example 16. The Rectangle is Transformed by the ItemTransform

##### Attribute of both the Rectangle and the Group

```
<Group Self="ud2" ItemTransform="0.8660254037844387 -0.5000000000000001
0.5000000000000001 0.8660254037844387 158.46925639128065 15.415316289918337">
  <Rectangle Self="ud0" ItemTransform="0.8660254037844387 -0.5000000000000001
0.5000000000000001 0.8660254037844387 158.46925639128065 15.415316289918337">
    <Properties>
      <PathGeometry>
        <GeometryPathType PathOpen="false">
          <PathPointArray>
            <PathPointType Anchor="72 -324" LeftDirection="72 -324"
RightDirection="72 -324"/>
            <PathPointType Anchor="72 -252" LeftDirection="72 -252"
RightDirection="72 -252"/>
            <PathPointType Anchor="144 -252" LeftDirection="144 -252"
RightDirection="144 -252"/>
            <PathPointType Anchor="144 -324" LeftDirection="144 -324"
RightDirection="144 -324"/>
          </PathPointArray>
        </GeometryPathType>
      </PathGeometry>
    </Properties>
  </Rectangle>
  <Oval Self="ud1" ItemTransform="1 0 0 1 0 0">
    <Properties>
      <PathGeometry>
        <GeometryPathType PathOpen="false">
          <PathPointArray>
            <PathPointType Anchor="69.6615427318801 -315.5"
```

```

        LeftDirection="63.67562810529901 -315.5"
        RightDirection="75.6474573584612 -315.5"/>
      <PathPointType Anchor="80.5 -326.3384572681199"
        LeftDirection="80.5 -320.35254264153883"
        RightDirection="80.5 -332.32437189470096"/>
      <PathPointType Anchor="69.6615427318801
        -337.1769145362398"
        LeftDirection="75.6474573584612 -337.1769145362398"
        RightDirection="63.67562810529901 -337.1769145362398"/>
      <PathPointType Anchor="58.82308546376021
        -326.3384572681199"
        LeftDirection="58.82308546376021 -332.32437189470096"
        RightDirection="58.82308546376021 -320.35254264153883"/>
    </PathPointArray>
  </GeometryPathType>
</PathGeometry>
</Properties>
</Oval>
</Group>

```

The coordinates of the example point remain the same, because they're expressed in the *inner coordinates* of the rectangle. The successive application of transformations via the `ItemTransform` attributes of the `<Rectangle>` element and its container elements (a `<Group>` element and a `<Spread>` element, in this example) determine where the point appears on the page you see in the InDesign user interface.

#### 10.3.4 Page Item Examples

The following IDML example shows a simple rectangle.

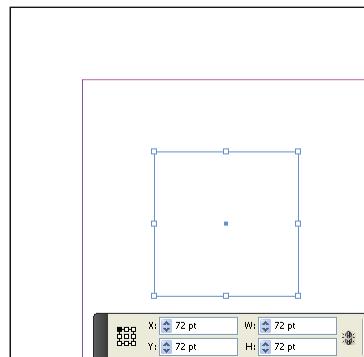
##### IDML Example 17. Page Item

```

<Rectangle Self="ucd" ItemTransform="1 0 0 1 0 0">
  <Properties>
    <PathGeometry>
      <GeometryPath PathOpen="false">
        <PathPointArray>
          <PathPointType Anchor="72 -324" LeftDirection="72 -324"
            RightDirection="72 -324"/>
          <PathPointType Anchor="72 -252" LeftDirection="72 -252"
            RightDirection="72 -252"/>
          <PathPointType Anchor="144 -252" LeftDirection="144 -252"
            RightDirection="144 -252"/>
          <PathPointType Anchor="144 -324" LeftDirection="144 -324"
            RightDirection="144 -324"/>
        </PathPointArray>
      </GeometryPath>
    </PathGeometry>
  </Rectangle>

```

Figure 14. Simple Page Item (a Rectangle)



### Nested Page Item

InDesign page items can be pasted inside other page items. IDML reflects this capability by storing the nested page items as child elements of the page item element, as shown in the following example.

#### IDML Example 18. Nested Page Item

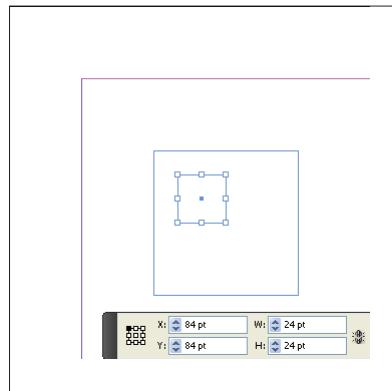
```
<Rectangle Self="ucd" ItemTransform="1 0 0 1 0 0">
  <Properties>
    <PathGeometry>
      <GeometryPath PathOpen="false">
        <PathPointArray>
          <PathPointType Anchor="72 -324" LeftDirection="72 -324"
            RightDirection="72 -324"/>
          <PathPointType Anchor="72 -252" LeftDirection="72 -252"
            RightDirection="72 -252"/>
          <PathPointType Anchor="144 -252" LeftDirection="144 -252"
            RightDirection="144 -252"/>
          <PathPointType Anchor="144 -324" LeftDirection="144 -324"
            RightDirection="144 -324"/>
        </PathPointArray>
      </GeometryPath>
    </PathGeometry>
  </Properties>
<Rectangle Self="uda" ItemTransform="1 0 0 1 0 0">
  <Properties>
    <PathGeometry>
      <GeometryPath PathOpen="false">
        <PathPointArray>
          <PathPointType Anchor="84 -312" LeftDirection="84 -312"
            RightDirection="84 -312"/>
          <PathPointType Anchor="84 -288" LeftDirection="84 -288"
            RightDirection="84 -288"/>
          <PathPointType Anchor="108 -288" LeftDirection="108 -288"
            RightDirection="108 -288"/>
          <PathPointType Anchor="108 -312" LeftDirection="108 -312"
            RightDirection="108 -312"/>
        </PathPointArray>
      </GeometryPath>
    </PathGeometry>
  </Properties>
```

```

    </PathGeometry>
    </Properties>
</Rectangle>
</Rectangle>

```

Figure 15. Nested Page Item



### **Rotated Page Item**

The following example shows a page item that has been rotated by 30 degrees around its center point. Note that the path point coordinates and geometric bounds remain the same as in the previous example; only the transformation matrix changes.

In general, applying a counterclockwise rotation using the transformation matrix takes the form:

$$a = \cos(\text{angle}), b = -\sin(\text{angle}), c = \sin(\text{angle}), d = \cos(\text{angle})$$

Where  $a$ ,  $b$ ,  $c$ , and  $d$  are the first four values in the transformation matrix and  $\text{angle}$  is the angle of rotation (in degrees).

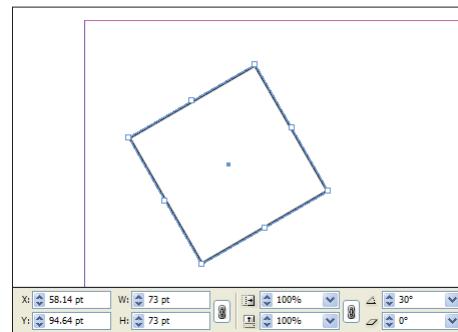
#### **IDML Example 19. Rotated Page Item**

```

<Rectangle Self="ucd" ItemTransform=" 0.8660254037844387 -0.5 0.5
0.8660254037844387 0 0">
<Properties>
    <PathGeometry>
        <GeometryPath PathOpen="false">
            <PathPointArray>
                <PathPointType Anchor="72 -324" LeftDirection="72 -324"
RightDirection="72 -324"/>
                <PathPointType Anchor="72 -252" LeftDirection="72 -252"
RightDirection="72 -252"/>
                <PathPointType Anchor="144 -252" LeftDirection="144 -252"
RightDirection="144 -252"/>
                <PathPointType Anchor="144 -324" LeftDirection="144 -324"
RightDirection="144 -324"/>
            </PathPointArray>
        </GeometryPath>
    </PathGeometry>
</Properties>
</Rectangle>

```

Figure 16. Rotated Page Item



The following example shows the transformation matrix for the rectangle:

```
a = 0.8660254037844387 = Cosine of 30 degrees.  
b = -0.5 = Negative sine of 30 degrees.  
c = 0.5 = Sine of 30 degrees.  
d = 0.8660254037844387 = Cosine of 30 degrees.
```

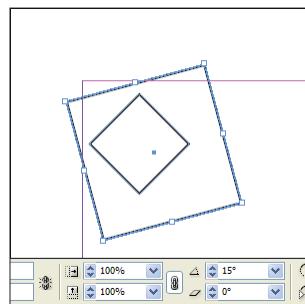
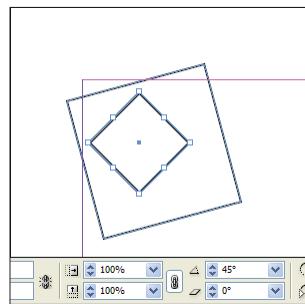
### **Transformations and Nested Page Items**

The following example shows a rectangle that has been rotated 30 degrees and then pasted inside another rectangle. The container rectangle is then rotated a further 15 degrees.

```
<Rectangle Self="ucd" ItemTransform="0.9659258262890683 -0.25881904510252074  
0.25881904510252074 0.9659258262890683 0 0">  
  <Properties>  
    <PathGeometry>  
      <GeometryPath PathOpen="false">  
        <PathPointArray>  
          <PathPointType Anchor="72 -324" LeftDirection="72 -324"  
RightDirection="72 -324"/>  
          <PathPointType Anchor="72 -252" LeftDirection="72 -252"  
RightDirection="72 -252"/>  
          <PathPointType Anchor="144 -252" LeftDirection="144 -252"  
RightDirection="144 -252"/>  
          <PathPointType Anchor="144 -324" LeftDirection="144 -324"  
RightDirection="144 -324"/>  
        </PathPointArray>  
      </GeometryPath>  
    </PathGeometry>  
  </Properties>  
  <Rectangle Self="uda" ItemTransform=" 0.8660254037844387 -0.5 0.5  
0.8660254037844387 0 0">  
    <Properties>  
      <PathGeometry>  
        <GeometryPath PathOpen="false">  
          <PathPointArray>  
            <PathPointType Anchor="84 -312" LeftDirection="84 -312"  
RightDirection="84 -312"/>  
            <PathPointType Anchor="84 -288" LeftDirection="84 -288"  
RightDirection="84 -288"/>
```

```

    RightDirection="84 -288"/>
    <PathPointType Anchor="108 -288" LeftDirection="108 -288"
    RightDirection="108 -288"/>
    <PathPointType Anchor="108 -312" LeftDirection="108 -312"
    RightDirection="108 -312"/>
  </PathPointArray>
</GeometryPath>
</PathGeometry>
</Properties>
</Rectangle>
</Rectangle>
```

*Figure 17. Outer Rectangle**Figure 18. Inner Rectangle*

Viewed from InDesign’s user interface, you can see that the rotation is cumulative—the rotation applied to the inner rectangle is the sum of its original rotation plus the rotation of the outer rectangle. The transformation matrices in the IDML example reflect this rotation. The following is the transformation matrix for the outer rectangle (formatted for easier reading):

```

a = 0.9659258262890683 = Cosine of 15 degrees.
b = -0.25881904510252074 = Negative sine of 15 degrees.
c = 0.25881904510252074 = Sine of 15 degrees.
d = 0.9659258262890683 = Cosine of 15 degrees.
```

The following example shows the transformation matrix for the inner rectangle:

```

a = 0.8660254037844387 = Cosine of 30 degrees.
b = -0.5000000000000001 = Negative sine of 30 degrees.
c = 0.5000000000000001 = Sine of 30 degrees.
d = 0.8660254037844387 = Cosine of 30 degrees.
```

### **Page Item Containing an Imported Graphic**

Placed (imported) graphics always appear inside a page item. This page item can be a rectangle, a graphic line, an oval, or a polygon. There is no special class of page item for a graphic frame.

#### **Schema Example 29. Image Schema**

```

image_Object = element Image {
    attribute Self { xsd:string },
    attribute FillColor { xsd:string }?,
    attribute FillTint { xsd:double }?,
    attribute OverprintFill { xsd:boolean }?,
    attribute Nonprinting { xsd:boolean }?,
    attribute Space { xsd:string }?,
    attribute ActualPpi { list { xsd:double ,xsd:double } }?,
    attribute EffectivePpi { list { xsd:double ,xsd:double } }?,
    attribute ImageRenderingIntent { RenderingIntent_EnumValue }?,
    attribute LocalDisplaySetting { DisplaySettingOptions_EnumValue }?,
    attribute ImageTypeName { xsd:string }?,
    attribute AppliedObjectStyle { xsd:string }?,
    attribute ItemTransform { TransformationMatrixType_TypeDef }?,
    element Properties {
        element Profile {
            (enum_type, Profile_EnumValue ) |
            (string_type, xsd:string )
        }?&
        element Contents { text }?&
        element GraphicProxy { text }?&
        element ClippingPathGeometry { element GeometryPath { GeometryPath_TypeDef }* }?&
        element GraphicBounds { RectangleBoundsType_TypeDef }?&
        element Label { element KeyValuePair { KeyValuePair_TypeDef }* }?
    }
    ?
    '
}
TransparencySetting_Object?&
StrokeTransparencySetting_Object?&
FillTransparencySetting_Object?&
ContentTransparencySetting_Object?&
TextWrapPreference_Object?&
MetadataPacketPreference_Object?&
Link_Object*&
ClippingPathSettings_Object?&
ImageIOPreference_Object?&
GraphicLayerOption_Object?&
LayerCompOption_Object?
)
}

```

**Table 34. Image Properties Represented as Attributes**

Name	Type	Req	Description
AppliedObject-Style	string	no	The object style applied to the image..
FillColor	string	no	The swatch (color, gradient, tint, or mixed ink) applied to the to fill the image.
FillTint	double	no	The percent tint to use in the fill color of the image. (To specify a tint percent, use a number in the range of 0 to 100; to use the inherited or overridden value, use -1.)
ImageRendering-Intent	RenderingIntent_EnumValue	no	The rendering intent override applied to the image. Can be <code>UseColorSettings</code> (Uses the current color settings), <code>Perceptual</code> (Preserves the visual relationship between colors at the expense of actual color values; most suitable for photographic images with high percentages of out-of-gamut colors), <code>Saturation</code> (Produces vivid colors at the expense of color accuracy; most suitable for business graphics such as graphs or charts), <code>RelativeColorimetric</code> (Compares the extreme highlight of the source color space to that of the destination color space and shifts all colors accordingly; out-of-gamut colors are shifted to the closest reproducible color in the destination color space), or <code>Absolute-Colorimetric</code> (Maintains color accuracy at the expense of preserving relationships between colors; most suitable for previewing how paper color affects printed colors).
ImageTypeName	string	no	The type of the image.
ItemTransform	Transformation-MatrixType_TypeDef	no	The transformation matrix applied to the image.
LocalDisplay-Setting	DisplaySetting-Options_EnumValue	no	Display performance options for the Image. Can be <code>HighQuality</code> (Slower performance; displays high-resolution graphics and high-quality transparencies and turns on anti-aliasing), <code>Typical</code> (Moderate performance speed; displays proxy graphics and low-quality transparencies and turns on anti-aliasing), <code>Optimized</code> (Best performance; grays out graphics and turns off transparency and anti-aliasing), or <code>Default</code> (Uses the container object's default display performance setting).
Nonprinting	boolean	no	If true, the image does not print.
OverprintFill	boolean	no	If true, the Ifill color of the image overprints any underlying objects. If false, the fill color knocks out the underlying colors.
Space	string	no	The color space.

**Image Properties Represented as Elements**

Name	Type	Req	Description
ClippingPath-Geometry	GeometryPath	no	The clipping path for the image.
Contents	text	no	If the graphic is embedded (rather than stored externally), this element contains the data of the graphic.
GraphicBounds	RectangleBounds	no	The geometric bounds of the image, in the form [x1, y1, x2, y2].
GraphicProxy	text	no	The data of the proxy image.
Profile	Profile_EnumValue or string	no	The color profile. Can return: Profile enumerator or String. Can be PostScriptCMS (Uses the PostScript CMS profile), UseDocument (Uses the document profile), Working (Uses the working profile), or NoCMS (No CMS profile is used).

The following example shows an image that has been placed inside the rectangle that we've used in the previous examples. We've omitted the XMP metadata from this example.

**IDML Example 20. Page Item Containing an Imported Graphic**

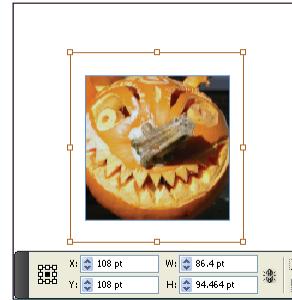
```
<Rectangle Self="ucd" ItemTransform="1 0 0 1 0 0">
  <Properties>
    <PathGeometry>
      <GeometryPath PathOpen="false">
        <PathPointArray>
          <PathPointType Anchor="72 -324" LeftDirection="72 -324"
RightDirection="72 -324"/>
          <PathPointType Anchor="72 -252" LeftDirection="72 -252"
RightDirection="72 -252"/>
          <PathPointType Anchor="144 -252" LeftDirection="144 -252"
RightDirection="144 -252"/>
          <PathPointType Anchor="144 -324" LeftDirection="144 -324"
RightDirection="144 -324"/>
        </PathPointArray>
      </GeometryPath>
    </PathGeometry>
  </Properties>
  <Image Self="udd" Space="$ID/#Links_RGB" ActualPpi="300 300"
EffectivePpi="500 500" ImageRenderingIntent="UseColorSettings"
LocalDisplaySetting="Default" ImageTypeName="$ID/JPEG"
AppliedObjectStyle="ObjectStyle/$ID/[None]"
ItemTransform="0.6 0 0 0.6 64.77191011235956 -335.27547826086965">
    <Properties>
      <Profile type="string">$ID/Embedded</Profile>
      <GraphicBounds Left="0" Top="0" Right="144" Bottom="157.44"/>
    </Properties>
    <Link Self="ue2" AssetURL="$ID/" AssetID="$ID/"
LinkResourceURI="file://ruri/documents/IDML/assets/pumpkin.jpg"
LinkResourceFormat="$ID/JPEG" StoredState="Normal" LinkClassID="35906"
LinkClientID="257" LinkResourceModified="false" LinkObjectModified="false"
ShowInUI="true" CanEmbed="true" CanUnembed="true" CanPackage="true"
```

```

ImportPolicy="NoAutoImport" ExportPolicy="NoAutoExport"
LinkImportStamp="file 128385019586602016 396019"
LinkImportModificationTime="2007-11-02T11:32:38"
LinkImportTime="2008-06-12T17:37:35"/>
<ClippingPathSettings ClippingType="None" InvertPath="false"
IncludeInsideEdges="false" RestrictToFrame="false"
UseHighResolutionImage="true" Threshold="25" Tolerance="2"
InsetFrame ="0" AppliedPathName="$ID/" Index="-1"/>
<ImageIOPreference ApplyPhotoshopClippingPath="true"
AllowAutoEmbedding="true" AlphaChannelName="$ID/"/>
</Image>
</Rectangle>
```

The example image is slightly larger than the rectangle. When you select the graphic in the InDesign user interface, you can see that the bounds of the graphic are outside the bounds of the containing rectangle (in the illustration below, the graphic is selected; the frame itself is the light blue square within the bounding box of the graphic).

*Figure 19. Page Item Containing an Imported Graphic*



## ***Text Frames***

<TextFrame> elements have a few properties that other page items do not have. The following sections describe these properties.

### **Schema Example 30. TextFrame**

```

TextFrame_Object = element TextFrame {
    attribute Self { xsd:string },
    attribute ParentStory { xsd:string }?,
    attribute PreviousTextFrame { xsd:string }?,
    attribute NextTextFrame { xsd:string }?,
    attribute ContentType { ContentType_EnumValue }?,
    attribute AllowOverrides { xsd:boolean }?,
    attribute FillColor { xsd:string }?,
    attribute FillTint { xsd:double }?,
    attribute OverprintFill { xsd:boolean }?,
    attribute StrokeWeight { xsd:double }?,
    attribute MiterLimit { xsd:double {minInclusive="1" maxInclusive="500"} }?,
    attribute EndCap { EndCap_EnumValue }?,
    attribute EndJoin { EndJoin_EnumValue }?,
    attribute StrokeType { xsd:string }?,
    attribute StrokeCornerAdjustment { StrokeCornerAdjustment_EnumValue }?,
    attribute StrokeDashAndGap { list { xsd:double * } }?,
```

```

attribute LeftLineEnd { ArrowHead_EnumValue }?,
attribute RightLineEnd { ArrowHead_EnumValue }?,
attribute StrokeColor { xsd:string }?,
attribute StrokeTint { xsd:double }?,
attribute CornerRadius { xsd:double }?,
attribute GradientFillStart { UnitPointType_TypeDef }?,
attribute GradientFillLength { xsd:double }?,
attribute GradientFillAngle { xsd:double }?,
attribute GradientStrokeStart { UnitPointType_TypeDef }?,
attribute GradientStrokeLength { xsd:double }?,
attribute GradientStrokeAngle { xsd:double }?,
attribute OverprintStroke { xsd:boolean }?,
attribute GapColor { xsd:string }?,
attribute GapTint { xsd:double }?,
attribute OverprintGap { xsd:boolean }?,
attribute StrokeAlignment { StrokeAlignment_EnumValue }?,
attribute Nonprinting { xsd:boolean }?,
attribute ItemLayer { xsd:string }?,
attribute Locked { xsd:boolean }?,
attribute LocalDisplaySetting { DisplaySettingOptions_EnumValue }?,
attribute GradientFillHiliteLength { xsd:double }?,
attribute GradientFillHiliteAngle { xsd:double }?,
attribute GradientStrokeHiliteLength { xsd:double }?,
attribute GradientStrokeHiliteAngle { xsd:double }?,
attribute AppliedObjectStyle { xsd:string }?,
attribute CornerOption { CornerOptions_EnumValue }?,
attribute ItemTransform { TransformationMatrixType_TypeDef }?,
element Properties {
    element PathGeometry { element GeometryPathType { GeometryPathType_TypeDef }* }?&
    element Label { element KeyValuePair { KeyValuePair_TypeDef }* }?
}
?
(
    (TextPath_Object*&
    GridDataInformation_Object?),
    (TransparencySetting_Object?&
    StrokeTransparencySetting_Object?&
    FillTransparencySetting_Object?&
    ContentTransparencySetting_Object?&
    TextFramePreference_Object?&
    AnchoredObjectSetting_Object?&
    BaselineFrameGridOption_Object?&
    TextWrapPreference_Object?)?
)

```

**Table 35. Text Frame Properties Represented as Attributes**

Name	Type	Req	Description
NextTextFrame	string	no	The value of the <code>Self</code> attribute of the next text frame (relative to the threading order of the text frames in the story) linked to this text frame.
ParentStory	string	no	The value of the <code>Self</code> attribute of the story that appears in this text frame.
PreviousTextFrame	string	no	The value of the <code>Self</code> attribute of the previous text frame (relative to the threading order of the text frames in the story) linked to this text frame.

It is important to note that the text of a text frame does not appear in the `<TextFrame>` element. Instead, the `<TextFrame>` element contains a reference to a `<Story>` element (usually stored within a `Story.xml` file within the IDML package). That element contains the text that appears in the text frame. This is true even when the story is linked to an external file—the text frame refers to the `Story.xml` file in the IDML package, not to the external file, and the `<Story>` element contains the reference to the external file.

The following example shows a simple text frame.

#### IDML Example 21. Text Frame

```

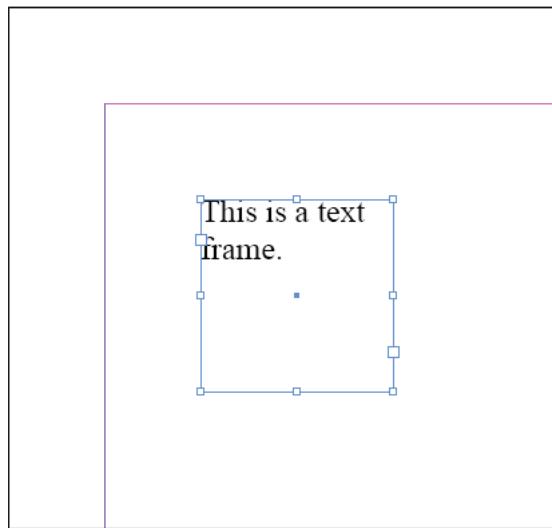
<TextFrame Self="ucd" ParentStory="ue7" PreviousTextFrame="n" NextTextFrame="n"
ContentType="TextType" ItemTransform="1 0 0 1 0 0">
  <Properties>
    <PathGeometry>
      <GeometryPathType PathOpen="false">
        <PathPointArray>
          <PathPointType Anchor="72 -324" LeftDirection="72 -324"
RightDirection="72 -324"/>
          <PathPointType Anchor="72 -252" LeftDirection="72 -252"
RightDirection="72 -252"/>
          <PathPointType Anchor="144 -252" LeftDirection="144 -252"
RightDirection="144 -252"/>
          <PathPointType Anchor="144 -324" LeftDirection="144 -324"
RightDirection="144 -324"/>
        </PathPointArray>
      </GeometryPathType>
    </Properties>
    <TextFramePreference TextColumnCount="1" TextColumnGutter="12" TextColumnFixed-
Width="72" UseFixedColumnWidth="false" FirstBaselineOffset="AscentOffset" Minimum-
FirstBaselineOffset="0" VerticalJustification="TopAlign" VerticalThreshold="0"
IgnoreWrap="false">
      <Properties>
        <InsetSpacing type="list">
          <ListItem type="unit">0</ListItem>
          <ListItem type="unit">0</ListItem>
          <ListItem type="unit">0</ListItem>
          <ListItem type="unit">0</ListItem>
        </InsetSpacing>
      </Properties>
    </TextFramePreference>
  
```

```

<BaselineFrameGridOption UseCustomBaselineFrameGrid="false"
StartingOffsetForBaselineFrameGrid="0"
BaselineFrameGridRelativeOption="TopOfInset" BaselineFrameGridIncrement="12">
<Properties>
<BaselineFrameGridColor type="enumeration">LightBlue
</BaselineFrameGridColor>
</Properties>
</BaselineFrameGridOption>
</TextFrame>

```

Figure 20. Text Frame



The following example shows how to use the `NextTextFrame` attribute to thread (or link) the text frame to another text frame. Note that the two text frames refer to the same story; you cannot have linked text frames that refer to separate stories.

#### IDML Example 22. Threaded (Linked) Text Frames

```

<TextFrame Self="ucd" AppliedObjectStyle="ObjectStyle\[Normal Graphics Frame]"
ParentStory="uf2" PreviousTextFrame="n" NextTextFrame="u108" ContentType="Text-
Type" StoryOffset="n" StrokeWeight="0"
StrokeColor="Swatch\cNone" ItemTransform="1 0 0 1 0 -396" >
<Properties>
<PathGeometry>
<GeometryPathType PathOpen="false">
<PathPointArray>
<PathPointType Anchor="36 72" LeftDirection="36 72"
RightDirection="36 72"/>
<PathPointType Anchor="36 96" LeftDirection="36 96"
RightDirection="36 96"/>
<PathPointType Anchor="136 96" LeftDirection="136 96"
RightDirection="136 96"/>
<PathPointType Anchor="136 72" LeftDirection="136 72"
RightDirection="136 72"/>
</PathPointArray>
</GeometryPathType>
</PathGeometry>
</Properties>

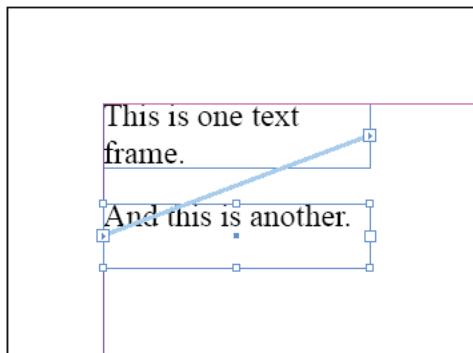
```

```

    </TextFrame>
    <TextFrame Self="u108" AppliedObjectStyle="ObjectStyle\k[None]" ParentStory="uf2"
    PreviousTextFrame="ucd" NextTextFrame="n" ContentType="TextType" StoryOffset="n"
    StrokeWeight="1" ItemTransform="1 0 0 1 0 -396">
        <Properties>
            <PathGeometry>
                <GeometryPathType PathOpen="false">
                    <PathPointArray>
                        <PathPointType Anchor="36 108" LeftDirection="36 108"
                        RightDirection="36 108"/>
                        <PathPointType Anchor="36 132" LeftDirection="36 132"
                        RightDirection="36 132"/>
                        <PathPointType Anchor="136 132" LeftDirection="136 132"
                        RightDirection="136 132"/>
                        <PathPointType Anchor="136 108" LeftDirection="136 108"
                        RightDirection="136 108"/>
                    </PathPointArray>
                </GeometryPathType>
            </PathGeometry>
        </Properties>
    </TextFrame>

```

Figure 21. Threaded Text Frames



### **TextFramePreference**

The `<TextFrame>` element also differs from other page items in that it contains a `<TextFramePreference>` element. The `<TextFramePreference>` element contains attributes and elements that control properties such as number of columns in the text frame, the text frame inset distances, and the method used to calculate the location of the first baseline of text in the text frame.

#### **Schema Example 31. TextFramePreference**

```

TextFramePreference_Object = element TextFramePreference {
    attribute TextColumnCount { xsd:int {minInclusive="1" maxInclusive="40"} }?,
    attribute TextColumnGutter { xsd:double {minInclusive="0"
    maxInclusive="8640"} }?,
    attribute TextColumnFixedWidth { xsd:double {minInclusive="0"
    maxInclusive="8640"} }?,
    attribute UseFixedColumnWidth { xsd:boolean }?,
    attribute FirstBaselineOffset { FirstBaseline_EnumValue }?,
    attribute MinimumFirstBaselineOffset { xsd:double {minInclusive="0"
    maxInclusive="8640"} }?
}

```

```

maxInclusive="8640"} }?,
attribute VerticalJustification { VerticalJustification_EnumValue }?,
attribute VerticalThreshold { xsd:double {minInclusive="0"
maxInclusive="8640"} }?,
attribute IgnoreWrap { xsd:boolean }?,
element Properties {
    element InsetSpacing {
        (unit_type, xsd:double {minInclusive="0" maxInclusive="8640"} ) |
        (list_type,
            element ListItem { unit_type, xsd:double {minInclusive="0"
maxInclusive="8640"} },
            element ListItem { unit_type, xsd:double {minInclusive="0"
maxInclusive="8640"} },
            element ListItem { unit_type, xsd:double {minInclusive="0"
maxInclusive="8640"} },
            element ListItem { unit_type, xsd:double {minInclusive="0"
maxInclusive="8640"} })
    }?
}
?
}
}

```

**Table 36. TextFramePreference Properties Represented as Attributes**

Name	Type	Req	Description
FirstBaseline-Offset	FirstBaseline_EnumValue	no	The distance between the baseline of the text and the top inset of the text frame. Values can be AscentOffset, CapHeight, LeadingOffset, EmboxHeight, XHeight, or FixedHeight.
IgnoreWrap	boolean	no	If true, ignores text wrap settings for drawn or placed objects which intersect the text frame.
MinimumFirst-BaselineOffset	double {min-Inclusive="0" max-Inclusive="8640"}	no	The minimum distance between the baseline of the text and the top inset of the text frame.
Self	string	yes	The unique ID of the object.
TextColumnCount	int	no	The number of columns in the text frame. Note: Depending on the value of the UseFixed-ColumnWidth attribute, the number of columns can change automatically when the text frame size changes. Range: 1 to 40.
TextColumnFixed-Width	double	no	The column width of the columns in the text frame. Range: 0 to 8640.
TextColumnGutter	double	no	The space between columns of the text frame. Range: 0 to 8640.

Name	Type	Req	Description
UseFixedColumnWidth	boolean	no	If true, maintains column width when the text frame is resized. If false, causes columns to resize when the text frame is resized. Note: When true, resizing the frame can change the number of columns in the frame.
VerticalJustification	VerticalJustification_EnumValue	no	The vertical alignment of the text in the text frame. Values can be TopAlign, CenterAlign, BottomAlign, or JustifyAlign.
VerticalThreshold	double	no	The maximum amount of vertical space between two paragraphs. Note: Valid only when the VerticalJustification attribute is JustifyAlign; the specified amount is applied in addition to the space before or space after values defined for the paragraphs in the text frame. Range: 0 to 8640.

**Table 37. TextFramePreference Properties Represented as Elements**

Name	Type	Req	Description
InsetSpacing	double or ListItem elements	no	The text insets applied to the text frame. Rectangular text frames can use four inset values (for the left, top, right, and bottom of the frame); non-rectangular text frames can use a single value (all sides). Can contain up to four ListItem elements.

**IDML Example 23. TextFramePreference (Single Inset Spacing Value)**

```
<TextFramePreference Self="udaTextFramePreference1" TextColumnCount="2" Text-
ColumnFixedWidth="120" FirstBaselineOffset="LeadingOffset">
  <Properties>
    <InsetSpacing type="unit">6</InsetSpacing>
  </Properties>
</TextFramePreference>
```

**IDML Example 24. TextFramePreference (Multiple Inset Spacing Values)**

```
<TextFramePreference Self="udaTextFramePreference1" TextColumnCount="2" Text-
ColumnFixedWidth="120" FirstBaselineOffset="LeadingOffset">
  <Properties>
    <InsetSpacing type="list">
      <ListItem type="unit">2</ListItem>
      <ListItem type="unit">3</ListItem>
      <ListItem type="unit">6</ListItem>
      <ListItem type="unit">3</ListItem>
    </InsetSpacing>
  </Properties>
</TextFramePreference>
```

The following example shows how to create a multi-column text frame using the properties of the TextFramePreference object.

**IDML Example 25. Multi-Column Text Frame**

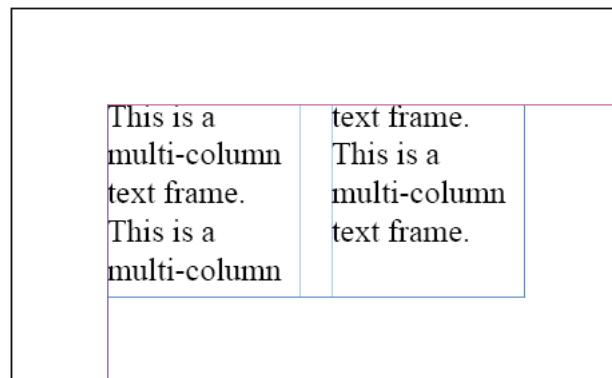
```
<TextFrame Self="ucd" AppliedObjectStyle="ObjectStyle\k[Normal Graphics Frame]">
```

```

ParentStory="uf2" PreviousTextFrame="n" NextTextFrame="n" ContentType="TextType"
StoryOffset="n" StrokeWeight="0" StrokeColor="Swatch\cNone"
ItemTransform="1 0 0 1 6 -432">
<Properties>
  <PathGeometry>
    <GeometryPathType PathOpen="false">
      <PathPointArray>
        <PathPointType Anchor="30 72" LeftDirection="30 72"
          RightDirection="30 72"/>
        <PathPointType Anchor="30 144" LeftDirection="30 144"
          RightDirection="30 144"/>
        <PathPointType Anchor="186 144" LeftDirection="186 144"
          RightDirection="186 144"/>
        <PathPointType Anchor="186 72" LeftDirection="186 72"
          RightDirection="186 72"/>
      </PathPointArray>
    </GeometryPathType>
  </PathGeometry>
</Properties>
<TextFramePreference Self="ucdTextFramePreference1" TextColumnCount="2"
TextColumnGutter="12" TextColumnFixedWidth="72" UseFixedColumnWidth="false"
FirstBaselineOffset="AscentOffset" MinimumFirstBaselineOffset="0"
VerticalJustification="TopAlign" VerticalThreshold="0" IgnoreWrap="false">
  <Properties>
    <InsetSpacing type="list">
      <ListItem type="unit">0</ListItem>
      <ListItem type="unit">0</ListItem>
      <ListItem type="unit">0</ListItem>
      <ListItem type="unit">0</ListItem>
    </InsetSpacing>
  </Properties>
</TextFramePreference>
</TextFrame>

```

Figure 22. Multi-Column Text Frame



### **BaselineFrameGridOptions**

Text frames can also contain a `<BaselineFrameGridOption>` element. This element contains properties expressed as attributes and elements that control the baseline grid options for the text

frame. The baseline frame grid affects paragraphs in the text frame that have been set to snap to the baseline grid—if the `UseCustomBaselineFrameGrid` attribute is `true`, then the baselines of the paragraphs will snap to the grid defined by the `<BaselineFrameGrid>` element; if it's `false`, they'll snap to the document baseline grid (which is defined in the `<GridPreference>` element). For more on grids and guides, refer to the InDesign online help.

#### **Schema Example 32. BaselineFrameGrid**

```
BaselineFrameGridOption_Object = element BaselineFrameGridOption {
    attribute Self { xsd:string },
    attribute UseCustomBaselineFrameGrid { xsd:boolean }?,
    attribute StartingOffsetForBaselineFrameGrid { xsd:double {minInclusive="0"
maxInclusive="8640"} }?,
    attribute BaselineFrameGridRelativeOption {
        BaselineFrameGridRelativeOption_EnumValue }?,
    attribute BaselineFrameGridIncrement { xsd:double {minInclusive="1"
maxInclusive="8640"} }?,
    element Properties {
        element BaselineFrameGridColor { InDesignUIColorType_TypeDef }?
    }
    ?
}
```

#### **IDML Example 26. BaselineFrameGrid**

```
<BaselineFrameGridOption Self="udaBaselineFrameGridOption1" UseCustomBaseline-
FrameGrid="true" BaselineFrameGridRelativeOption="TopOfMargin">
    <Properties>
        <BaselineFrameGridColor type="enumeration">LightBlue</BaselineFrameGridColor>
    </Properties>
</BaselineFrameGridOption>
```

**Table 38. Baseline Frame Grid Properties Expressed as Attributes**

Name	Type	Req	Description
UseCustomBaselineFrameGrid	boolean	no	If true, uses a custom baseline frame grid.
StartingOffsetForBaselineFrameGrid	double	no	The amount to offset the baseline grid. Minimum 0, maximum 8640.
BaselineFrameGridRelativeOption	Baseline-FrameGrid-RelativeOption_EnumValue	no	The object from which to offset the custom baseline grid. Can be TopOfPage, TopOfMargin, TopOfFrame, or TopOfInset.
BaselineFrameGridIncrement	double	no	The distance between grid lines. Minimum 0, maximum 8640.

**Table 39. BaselineFrameGrid Properties Represented as Elements**

Name	Type	Req	Description
BaselineFrameGridColor	InDesignUIColorType	no	The grid line color, specified either as an array of three doubles, each in the range 0 to 255 and representing R, G, and B values, or as a UI color.

## **Transparency**

You can apply transparency effects to page items in an InDesign layout. In IDML, you accomplish this using the <TransparencySetting> element. A child element (or elements) of this element specify the transparency effect you want to apply.

### **Schema Example 33. TransparencySetting**

```
TransparencySetting_Object = element TransparencySetting {
    attribute Self { xsd:string },
    (
        BlendingSetting_Object?&
        FindChangeBlendingSetting_Object?&
        DropShadowSetting_Object?&
        FindChangeDropShadowSetting_Object?&
        FeatherSetting_Object?&
        FindChangeFeatherSetting_Object?&
        InnerShadowSetting_Object?&
        FindChangeInnerShadowSetting_Object?&
        OuterGlowSetting_Object?&
        FindChangeOuterGlowSetting_Object?&
        InnerGlowSetting_Object?&
        FindChangeInnerGlowSetting_Object?&
        BevelAndEmbossSetting_Object?&
        FindChangeBevelAndEmbossSetting_Object?&
        SatinSetting_Object?&
        FindChangeSatinSetting_Object?&
        DirectionalFeatherSetting_Object?&
        FindChangeDirectionalFeatherSetting_Object?&
        GradientFeatherSetting_Object?&
        FindChangeGradientFeatherSetting_Object?
    )
}
```

### **Schema Example 34. BlendingSetting**

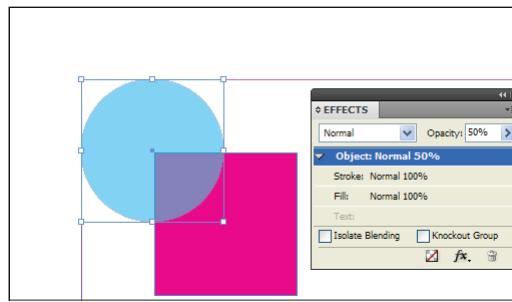
```
BlendingSetting_Object = element BlendingSetting {
    attribute Self { xsd:string },
    attribute BlendMode { BlendMode_EnumValue }?,
    attribute Opacity { xsd:double {minInclusive="0" maxInclusive="100"} }?,
    attribute KnockoutGroup { xsd:boolean }?,
    attribute IsolateBlending { xsd:boolean }?
}
```

In the following example, a <BlendingSetting> element applies transparency to the <Oval> element. The appearance of a transparent object is affected by the transparency flattener settings of the document and of the spread containing the transparent page items.

### **IDML Example 27. Transparency**

```
<Oval Self="ucd" FillColor="Color\cC=100 M=0 Y=0 K=0">
    <TransparencySetting Self="ucdTransparencySetting1">
        <BlendingSetting Self="ucdTransparencySetting1BlendingSetting1" Opacity="50"/>
    </TransparencySetting>
</Oval>
```

Figure 23. Transparency



## Group

InDesign page items can be grouped, and groups can contain other groups. Each object stored inside a group is represented as a child element of the `<Group>` element, as shown in the following example.

### IDML Example 28. Group

```

<Group Self="u111" ItemTransform="1 0 0 1 0 0">
    <Rectangle Self="u10d" AppliedObjectStyle="ObjectStyle\k[Normal Graphics Frame]" ItemTransform="1 0 0 1 0 0">
        <Properties>
            <PathGeometry>
                <GeometryPathType PathOpen="false">
                    <PathPointArray>
                        <PathPointType Anchor="36.5 -359.5" LeftDirection="36.5 -359.5" RightDirection="36.5 -359.5"/>
                        <PathPointType Anchor="36.5 -330.5" LeftDirection="36.5 -330.5" RightDirection="36.5 -330.5"/>
                        <PathPointType Anchor="65.5 -330.5" LeftDirection="65.5 -330.5" RightDirection="65.5 -330.5"/>
                        <PathPointType Anchor="65.5 -359.5" LeftDirection="65.5 -359.5" RightDirection="65.5 -359.5"/>
                    </PathPointArray>
                </GeometryPathType>
            </PathGeometry>
        </Properties>
    </Rectangle>
    <Polygon Self="u10e" AppliedObjectStyle="ObjectStyle\k[Normal Graphics Frame]" ItemTransform="1 0 0 1 0 0">
        <Properties>
            <PathGeometry>
                <GeometryPathType PathOpen="false">
                    <PathPointArray>
                        <PathPointType Anchor="113.71119986540519 -359.5" LeftDirection="113.71119986540519 -359.5" RightDirection="113.71119986540519 -359.5"/>
                        <PathPointType Anchor="97.28830013459482 -359.5" LeftDirection="97.28830013459482 -359.5" RightDirection="97.28830013459482 -359.5"/>
                        <PathPointType Anchor="89.07685026918963 -345.2773516293136" LeftDirection="89.07685026918963 -345.2773516293136"/>
                    </PathPointArray>
                </GeometryPathType>
            </PathGeometry>
        </Properties>
    </Polygon>
</Group>

```

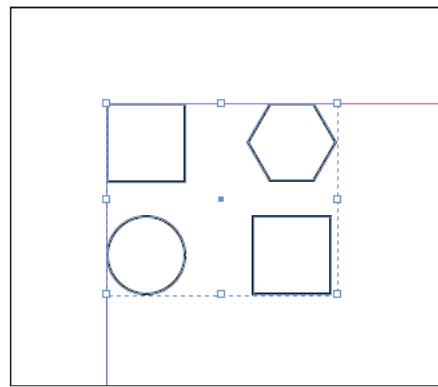
```

    RightDirection="89.07685026918963 -345.2773516293136"/>
    <PathPointType Anchor="97.28830013459482 -331.0547032586272"
    LeftDirection="97.28830013459482 -331.0547032586272">
    RightDirection="97.28830013459482 -331.0547032586272"/>
    <PathPointType Anchor="113.71119986540518 -331.0547032586272"
    LeftDirection="113.71119986540518 -331.0547032586272"
    RightDirection="113.71119986540518 -331.0547032586272"/>
    <PathPointType Anchor="121.92264973081038 -345.2773516293136"
    LeftDirection="121.92264973081038 -345.2773516293136"
    RightDirection="121.92264973081038 -345.2773516293136"/>
  </PathPointArray>
  </GeometryPathType>
</PathGeometry>
</Properties>
</Polygon>
<Oval Self="u10f" AppliedObjectStyle="ObjectStyle\k[Normal Graphics Frame]"
ItemTransform="1 0 0 1 0 0">
  <Properties>
    <PathGeometry>
      <GeometryPathType PathOpen="false">
        <PathPointArray>
          <PathPointType Anchor="51 -288.5005"
LeftDirection="42.9918711733334 -288.5005"
RightDirection="59.00812882666667 -288.5005"/>
          <PathPointType Anchor="65.5 -303.0005"
LeftDirection="65.5 -294.992371173333"
RightDirection="65.5 -311.0086288266666"/>
          <PathPointType Anchor="51 -317.5005"
LeftDirection="59.00812882666667 -317.5005"
RightDirection="42.9918711733334 -317.5005"/>
          <PathPointType Anchor="36.5 -303.0005"
LeftDirection="36.5 -311.0086288266666"
RightDirection="36.5 -294.992371173333"/>
        </PathPointArray>
      </GeometryPathType>
    </PathGeometry>
  </Properties>
</Oval>
<Rectangle Self="u110" AppliedObjectStyle="ObjectStyle\k[Normal Graphics Frame]"
ItemTransform="1 0 0 1 0 0">
  <Properties>
    <PathGeometry>
      <GeometryPathType PathOpen="false">
        <PathPointArray>
          <PathPointType Anchor="90.99950000000001 -317.5005"
LeftDirection="90.99950000000001 -317.5005"
RightDirection="90.99950000000001 -317.5005"/>
          <PathPointType Anchor="90.99950000000001 -288.5005"
LeftDirection="90.99950000000001 -288.5005"
RightDirection="90.99950000000001 -288.5005"/>
          <PathPointType Anchor="119.99950000000001 -288.5005"
LeftDirection="119.99950000000001 -288.5005"
RightDirection="119.99950000000001 -288.5005"/>
          <PathPointType Anchor="119.99950000000001 -317.5005"
LeftDirection="119.99950000000001 -317.5005"
RightDirection="119.99950000000001 -317.5005"/>
        </PathPointArray>
      </GeometryPathType>
    </PathGeometry>
  </Properties>

```

```
        LeftDirection="119.99950000000001 -317.5005"
        RightDirection="119.99950000000001 -317.5005"/>
    </PathPointArray>
  </GeometryPathType>
</PathGeometry>
</Properties>
</Rectangle>
</Group>
```

*Figure 24. Group*



## *Buttons*

The `<Button>` element can contain `<State>` elements, which contain page items that define the appearance of the button and attributes that define the behavior of the button. Buttons are objects you can define in InDesign that become interactive elements in exported PDF and SWF.

### Schema Example 35. Button

```
Button_Object = element Button {
    attribute Self { xsd:string },
    attribute Name { xsd:string }?,
    attribute Description { xsd:string }?,
    attribute VisibilityInPdf { VisibilityInPdf_EnumValue }?,
    attribute AllowOverrides { xsd:boolean }?,
    attribute FillColor { xsd:string }?,
    attribute FillTint { xsd:double }?,
    attribute OverprintFill { xsd:boolean }?,
    attribute StrokeWeight { xsd:double }?,
    attribute MiterLimit { xsd:double {minInclusive="1" maxInclusive="500"} }?,
    attribute EndCap { EndCap_EnumValue }?,
    attribute EndJoin { EndJoin_EnumValue }?,
    attribute StrokeType { xsd:string }?,
    attribute StrokeCornerAdjustment { StrokeCornerAdjustment_EnumValue }?,
    attribute StrokeDashAndGap { list { xsd:double * } }?,
    attribute LeftLineEnd { ArrowHead_EnumValue }?,
    attribute RightLineEnd { ArrowHead_EnumValue }?,
    attribute StrokeColor { xsd:string }?,
    attribute StrokeTint { xsd:double }?,
    attribute CornerRadius { xsd:double }?,
    attribute GradientFillStart { UnitPointType TypeDef }?,
    attribute GradientFillEnd { UnitPointType TypeDef }?,
    attribute GradientFillCenter { UnitPointType TypeDef }?,
    attribute GradientFillAngle { xsd:double }?,
    attribute GradientFillType { xsd:string }?,
    attribute GradientFillMatrix { list { xsd:double * } }?
}
```

```

        attribute GradientFillLength { xsd:double }?,
        attribute GradientFillAngle { xsd:double }?,
        attribute GradientStrokeStart { UnitPointType_TypeDef }?,
        attribute GradientStrokeLength { xsd:double }?,
        attribute GradientStrokeAngle { xsd:double }?,
        attribute OverprintStroke { xsd:boolean }?,
        attribute GapColor { xsd:string }?,
        attribute GapTint { xsd:double }?,
        attribute OverprintGap { xsd:boolean }?,
        attribute StrokeAlignment { StrokeAlignment_EnumValue }?,
        attribute Nonprinting { xsd:boolean }?,
        attribute ItemLayer { xsd:string }?,
        attribute Locked { xsd:boolean }?,
        attribute LocalDisplaySetting { DisplaySettingOptions_EnumValue }?,
        attribute GradientFillHiliteLength { xsd:double }?,
        attribute GradientFillHiliteAngle { xsd:double }?,
        attribute GradientStrokeHiliteLength { xsd:double }?,
        attribute GradientStrokeHiliteAngle { xsd:double }?,
        attribute AppliedObjectStyle { xsd:string }?,
        attribute CornerOption { CornerOptions_EnumValue }?,
        attribute ItemTransform { TransformationMatrixType_TypeDef }?,
        element Properties {
            element PathBoundingBox { RectangleBoundsType_TypeDef }?&
            element PathGeometry { element GeometryPathType { GeometryPathType_TypeDef }* }?&
            element Label { element KeyValuePair { KeyValuePair_TypeDef }* }?
        }
        ?
        '
    (
        AnchoredObjectSetting_Object?&
        TextWrapPreference_Object?&
        State_Object*&
        Behavior_Object*&
        GotoFirstPageBehavior_Object*&
        GotoLastPageBehavior_Object*&
        GotoNextPageBehavior_Object*&
        GotoPreviousPageBehavior_Object*&
        GotoNextViewBehavior_Object*&
        GotoPreviousViewBehavior_Object*&
        GotoURLBehavior_Object*&
        GotoAnchorBehavior_Object*&
        MovieBehavior_Object*&
        SoundBehavior_Object*&
        ShowHideFieldsBehavior_Object*&
        OpenFileBehavior_Object*&
        CloseWindowBehavior_Object*&
        QuitBehavior_Object*&
        ViewZoomBehavior_Object*&
        GotoPageBehavior_Object*
    )
}

```

Most of the properties of a <Button> element are shared with other page items. For a listing of these properties, see “Common Page Item Properties.” A <Button> element can also have the following properties.

**Table 40. Button Properties Expressed as Attributes**

Name	Type	Req	Description
Description	string	no	The description of the button.
Name	string	no	The name of the button.
VisibilityInPdf	VisibilityInPdf_EnumValue	no	The field’s visibility in the PDF document. Can be VisibleInPdf (The field is visible), HiddenInPdf (The field is not visible), VisibleButDoesNotPrintInPdf (The field is visible when the PDF document is displayed on-screen but invisible when the document is printed), or HiddenButPrintableInPdf (The field is invisible when the PDF document is displayed on-screen but visible when the document is printed).

**Table 41. Button Properties Expressed as Properties**

Name	Type	Req	Description
PathBoundingBox	RectangleBounds-Type_TypeDef	no	The geometric bounds of the button, in the form [x1, y1, x2, y2].

**Schema Example 36. State**

```
State_Object = element State {
    attribute Self { xsd:string },
    attribute Name { xsd:string }?,
    attribute Active { xsd:boolean }?,
    attribute Enabled { xsd:boolean }?,
    attribute Statetype { StateTypes_EnumValue }?,
    (
        FormField_Object*&
        Button_Object*&
        Oval_Object*&
        Rectangle_Object*&
        GraphicLine_Object*&
        TextFrame_Object*&
        Graphic_Object*&
        ImportedPage_Object*&
        Image_Object*&
        EPS_Object*&
        WMF_Object*&
        PICT_Object*&
        PDF_Object*&
        Polygon_Object*&
        Group_Object*&
        EPSText_Object*
    )
}
```

**Table 42. State Properties Represented as Attributes**

Name	Type	Req	Description
Active	boolean	no	If true, the state is the active (or front most) state in the user interface.
Enabled	boolean	no	If true, objects that use the state appear in PDF documents. If false, objects that use the state do not appear in the document when the event that activates the state, such as a mouseover, occurs.
Name	string	yes	The name of the state.
Statetype	StateTypes_Enum-Value	no	The type of user action that dictates the button's appearance. Can be Up, Rollover, or Down.

Button properties represented as attributes are identical to those of other page items (see “Common Page Item Properties”). The main difference between a `<Button>` element and other page item elements is that the `<Button>` contains `<State>` elements (elements that define the appearance of the button in response to certain actions in an exported PDF or SWF), and can contain one or more of the “Behavior” objects, which are discussed after the following example. For more on button states and behaviors, refer to the InDesign online help.

#### IDML Example 29. Button

```

<Button Self="ud6" Name="GoToFirstPage" Description="" FillColor="Color/Black"
ItemTransform="1 0 0 1 0 0">
  <Properties>
    <PathBoundingBox Left="36" Top="-336.25" Right="76" Bottom="-312"/>
  </Properties>
  <State Self="ud6i0" Name="Up" Active="true" Enabled="true" Statetype="Up">
    <Group Self="ud5" FillColor="Color/Black" ItemTransform="1 -0 -0 1 -0 -0">
      <Polygon Self="ucc" ContentType="Unassigned" FillColor="Color/Black"
      StrokeWeight="0" ItemTransform="1 0 0 1 0 0">
        <Properties>
          <PathGeometry>
            <GeometryPathType PathOpen="false">
              <PathPointArray>
                <PathPointType Anchor="36 -324" LeftDirection="36 -324"
                RightDirection="36 -324"/>
                <PathPointType Anchor="60 -312" LeftDirection="60 -312"
                RightDirection="60 -312"/>
                <PathPointType Anchor="60 -336.25" LeftDirection="60 -336.25"
                RightDirection="60 -336.25"/>
              </PathPointArray>
            </GeometryPathType>
          </PathGeometry>
        </Properties>
        <TransparencySetting>
          <BevelAndEmbossSetting Applied="true" Style="Emboss" Size="3"/>
        </TransparencySetting>
      </Polygon>
      <Rectangle Self="ud3" StoryTitle="$ID/" ContentType="Unassigned"
      FillColor="Color/Black" StrokeWeight="0" StrokeColor="Swatch/None"
      ItemTransform="1 0 0 1 0 0">
        <Properties>
          <PathGeometry>

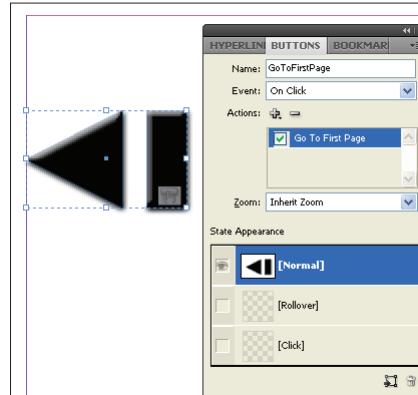
```

```

<GeometryPathType PathOpen="false">
    <PathPointArray>
        <PathPointType Anchor="66 -336.25" LeftDirection="66 -336.25"
            RightDirection="66 -336.25"/>
        <PathPointType Anchor="66 -312" LeftDirection="66 -312"
            RightDirection="66 -312"/>
        <PathPointType Anchor="76 -312" LeftDirection="76 -312"
            RightDirection="76 -312"/>
        <PathPointType Anchor="76 -336.25" LeftDirection="76 -336.25"
            RightDirection="76 -336.25"/>
    </PathPointArray>
</GeometryPathType>
</PathGeometry>
</Properties>
<TransparencySetting>
    <BevelAndEmbossSetting Applied="true" Style="Emboss" Size="3"/>
</TransparencySetting>
</Rectangle>
</Group>
</State>
<GotoFirstPageBehavior Self="ud7" ZoomSetting="InheritZoom"
    Name="Go To First Page" EnableBehavior="true" BehaviorEvent="MouseDown"/>
</Button>

```

Figure 25. Button



## Behaviors

You can create buttons in InDesign that perform an action when the document is exported to PDF format. For example, you can create a button that jumps to a different page of the PDF document or plays a movie clip. The type of action that a button can perform is called a *behavior*. For more on InDesign's button features, refer to the InDesign documentation.

### Schema Example 37. Behavior

```

Behavior_Object = element Behavior {
    attribute Self { xsd:string },
    attribute Name { xsd:string }?,
    attribute EnableBehavior { xsd:boolean }?,
    attribute BehaviorEvent { BehaviorEvents_EnumValue }?,
    element Properties {

```

```

element Label { element KeyValuePair { KeyValuePair_TypeDef }*
}?
}
?
}

```

The schemas for the behavior elements are all very similar. Rather than show them all here, we'll present the attributes common to all behavior elements, then describe the details of specific behavior element schemas when they differ from the others.

**Table 43. Common Behavior Properties Represented as Attributes**

Name	Type	Req	Description
Name	string	no	The name of the behavior.
EnableBehavior	boolean	no	If true, enable the button behavior.
BehaviorEvent	BehaviorEvents_EnumValue	no	The event that triggers the behavior. Can be MouseUp, MouseDown, MouseEnter, MouseExit, OnFocus, or OnBlur.

The GoToPage, GotoFirstPageBehavior, GotoLastPageBehavior, GotoNextPageBehavior, GotoPreviousPageBehavior, GotoNextViewBehavior, and GotoPreviousViewBehavior differ from the Behavior element only in that they contain a ZoomSetting attribute.

Name	Type	Req	Description
ZoomSetting	GoToZoomOptions_EnumValue	no	The zoom setting of the behavior. Can be InheritZoom, FitWindow, FitWidth, FitVisible, or ActualSize.

The GotoURLBehavior element differs from the Behavior element in that it has an additional attribute, URL.

Name	Type	Req	Description
URL	string	no	The URL hyperlink of the button.

The `GotoAnchorBehavior` contains three attributes that are not shared with the `Behavior` element: `ZoomSetting` (described above), `AnchorName`, and `FilePath`.

Name	Type	Req	Description
<code>AnchorName</code>	string	no	The name of the anchor.
<code>FilePath</code>	string	no	The file path to the file containing the anchor.

The `MovieBehavior` element differs from the `Behavior` element in that it has two additional attributes, `MovieItem` and `Operation`.

Name	Type	Req	Description
<code>MovieItem</code>	string	no	The path to the movie file.
<code>Operation</code>	<code>PlayOperations_EnumValue</code>	no	Can be <code>Play</code> , <code>Stop</code> , <code>Pause</code> , or <code>Resume</code> .

The `SoundBehavior` element differs from the `Behavior` element in that it has two additional attributes, `SoundItem` and `Operation`.

Name	Type	Req	Description
<code>SoundItem</code>	string	no	The path to the sound file.
<code>Operation</code>	<code>PlayOperations_EnumValue</code>	no	Can be <code>Play</code> , <code>Stop</code> , <code>Pause</code> , or <code>Resume</code> .

The `ShowHideFieldsBehavior` element has two attributes that differ from other `Behavior` elements, `FieldsToShow` and `FieldsToHide`.

Name	Type	Req	Description
<code>FieldsToShow</code>	list of strings as a space-separated string	no	A list of the fields to show, as a series of references (using the value of the <code>Self</code> attribute of the elements to refer to).
<code>FieldsToHide</code>	list of strings as a space-separated string	no	A list of the fields to hide, as a series of references (using the value of the <code>Self</code> attribute of the elements to refer to).

The `OpenFileBehavior` element has an attributes that differs from the `Behavior` element: `FilePath`.

Name	Type	Req	Description
<code>FilePath</code>	string	no	The file path to the file to open.

The `ViewZoomBehavior` element has an attributes that differs from the `Behavior` element: `viewZoomStyle`.

Name	Type	Req	Description
<code>ViewZoomStyle</code>	<code>ViewZoomStyle_EnumValue</code>	no	Can be <code>FullScreen</code> , <code>ZoomIn</code> , <code>ZoomOut</code> , <code>FitPage</code> , <code>ActualSize</code> , <code>FitWidth</code> , <code>FitVisible</code> , <code>Reflow</code> , <code>SinglePage</code> , <code>OneColumn</code> , <code>TwoColumn</code> , <code>RotateCW</code> , or <code>RotateCCW</code> .

The `GotoPageBehavior` element has two attributes that differ from the `Behavior` element: `View-ZoomStyle` (described above) and `PageNumber`.

Name	Type	Req	Description
PageNumber	int	no	The number of the page to display.

### Associating Page Items with XML Elements

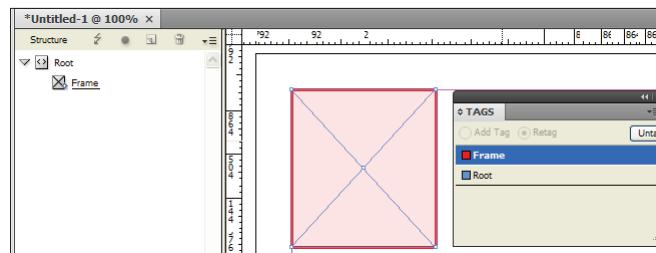
Page items associated with XML elements in the XML structure of an InDesign document do not differ from page items that are not associated with the structure—what sets them apart are references from `<XMLElement>` elements. In an IDML package, these elements appear in the `BackingStory.xml` file.

In the following example, the `<XMLElement>` element refers to a `<Rectangle>` element in the layout.

#### IDML Example 30. Associating a Frame with an XML Element

```
<XMLElement Self="di2" MarkupTag="XMLTag\cRoot">
  <XMLElement Self="di2i3" MarkupTag="XMLTag\cFrame" XMLContent="ud6"/>
</XMLElement>
<!--From Spread_ub8.xml-->
<Rectangle Self="ud6" .../>
```

Figure 26. Associating an XML Element with a Frame

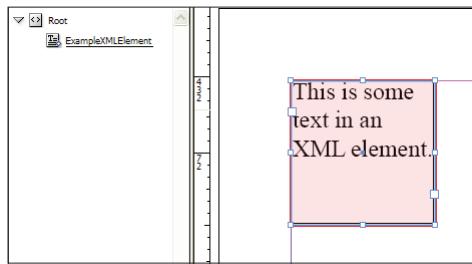


In the following example, the `<XMLElement>` refers to a `<Story>` element in the same package, and that the `<TextFrame>` element also refers to this `<Story>`. XML elements that have been associated with text appear inside the `<Story>` element.

#### IDML Example 31. Associating a Text Frame with an XML Element

```
<!--From BackingStory.xml-->
<XMLElement Self="di2i3" MarkupTag="XMLTag\ExampleXMLElement" XMLContent="ucf"/>
<!--From Story_ucf.xml-->
<Story Self="ucf" ...>
<!--From Spread_ub8.xml-->
<TextFrame Self="ucd" ParentStory="ucf" ... />
```

Figure 27. Associating an XML Element with a Text Frame



### 10.3.5 Pages

Spreads contain one or more pages. The pages in a spread are stored as `<Page>` elements, which properties that override document- or spread-based settings, such as trapping presets, the applied master spread, and page margin settings. Pages in a spread are of a single size (InDesign does not support multiple page sizes within a single document), and are added to the spread based on the order in which they appear inside the `<Spread>` element. The location of the pages inside the spread is determined by the binding direction of the document, which is defined by the `Binding-Location` attribute of the `<Spread>` element—refer to the InDesign documentation for more information on binding direction.

`<Page>` elements appear in the `<Spread>` element in the sequence in which they appear in the spread, relative to the binding direction specified by the `PageBinding` attribute of the `<Document-Preference>` element. (In a left to right layout, subsequent pages appear to the right of the first page in the spread; in a right to left layout, pages appear to the left of the first page in the spread.)

**Note:** Page items do not appear as child elements of page elements, but are, instead, collected on the spread.

#### Schema Example 38. Page Schema

```
Page_Object = element Page {
    attribute Self { xsd:string },
    attribute Name { xsd:string }?,
    attribute AppliedTrapPreset { xsd:string }?,
    attribute AppliedMaster { xsd:string }?,
    attribute OverrideList { list { xsd:string * } }?,
    attribute TabOrder { list { xsd:string * } }?,
    attribute GridStartingPoint { GridStartingPointOptions_EnumValue }?,
    attribute UseMasterGrid { xsd:boolean }?,
    element Properties {
        element Descriptor { list_type, element ListItem {
            (string_type, xsd:string ) |
            (enum_type, PageNumberStyle_EnumValue ) |
            (bool_type, xsd:boolean ) |
            (long_type, xsd:int {minInclusive="1" maxInclusive="999999"} )
        }* }?&
        element Label { element KeyValuePair { KeyValuePair_TypeDef }* }
    }
    ?
    ,
}
MarginPreference_Object?&
```

```

    Guide_Object*&
    GridDataInformation_Object?
)
}

```

The following example shows a (simplified) <Spread> element containing two pages. In a left-to-right layout, the first page in the spread is the left hand (verso) page; the second page is the right hand (recto).

#### **IDML Example 32. Pages Within a Spread**

```

<Spread Self="uff" FlattenerOverride="Default" ShowMasterItems="true" Page-
Count="2" BindingLocation="1" AllowPageShuffle="true" ItemTransform="1 0 0 1 0
972" AppliedMaster="ucf">
  <Page Self="u104" Name="2" AppliedTrapPreset="TrapPreset\kkDefaultTrapStyleName"
AppliedMaster="ucf" OverrideList="" TabOrder="" GridStartingPoint="TopOutside"
UseMasterGrid="true">
    <MarginPreference ColumnCount="1" ColumnGutter="12" Top="36" Bottom="36"
Left="36" Right="36" ColumnDirection="Horizontal" ColumnsPositions="0 540"/>
  </Page>
  <Page Self="u105" Name="3" AppliedTrapPreset="TrapPreset\kkDefaultTrapStyleName"
AppliedMaster="ucf" OverrideList="" TabOrder="" GridStartingPoint="TopOutside"
UseMasterGrid="true">
    <MarginPreference ColumnCount="1" ColumnGutter="12" Top="36" Bottom="36"
Left="36" Right="36" ColumnDirection="Horizontal" ColumnsPositions="0 540"/>
  </Page>
</Spread>

```

---

**Table 44. Page Properties Represented as Attributes**

Attribute	Type	Req	Description
Name	string	no	The name of the page.
AppliedTrapPreset	string	no	The name of the trapping preset applied to the page.
AppliedMaster	string	no	The master spread applied to the page.
OverrideList	list of strings as a space-separated string	no	The overridden master page items on this page, as a series of references (using the value of the Self attribute of the overridden page items). For more on overriding master page items on document pages, refer to the InDesign online help.
TabOrder	string list of strings as a space-separated string	no	The order in which the focus in an exported PDF moves to different form fields when the tab button is pressed, as a series of references (using the value of the Self attribute of the page items).
GridStartingPoint	GridStarting PointOptions _EnumValue	no	The starting point for the grid. Can be Top-Outside, TopInside, BottomOutside, BottomInside, CenterVertical, Center-Horizontal, or CenterCompletely.
UseMasterGrid	boolean	no	If true, use the master grid.

**Table 45. Page Properties Represented as Elements**

Name	Type	Req	Description
Descriptor	ListItem elements	no	A collection of properties used by InCopy assignments.

### 10.3.6 Columns and Margins

The columns and margins of a page are defined by the <MarginPreference> object.

#### Schema Example 39. Margin Preference

```
MarginPreference_Object = element MarginPreference {
    attribute Self { xsd:string },
    attribute ColumnCount { xsd:int {minInclusive="1" maxInclusive="216"} }?,
    attribute ColumnGutter { xsd:double {minInclusive="0" maxInclusive="1440"} }?,
    attribute Top { xsd:double }?,
    attribute Bottom { xsd:double }?,
    attribute Left { xsd:double }?,
    attribute Right { xsd:double }?,
    attribute ColumnDirection { HorizontalOrVertical_EnumValue }?,
    attribute ColumnsPositions { list { xsd:double * } }?
}
```

#### IDML Example 33. Margin Preferences

```
<MarginPreference ColumnCount="1" ColumnGutter="12" Top="36" Bottom="36"
Left="36" Right="36" ColumnDirection="Horizontal" ColumnsPositions="0 540"/>
```

### 10.3.7 Guides

In addition to page items, InDesign spreads can contain guides. For more on using guides to align page items and to mark areas of the page, refer to the InDesign online help.

#### Schema Example 40. Guide

```
Guide_Object = element Guide {
    attribute Self { xsd:string },
    attribute Orientation { HorizontalOrVertical_EnumValue }?,
    attribute Location { xsd:double }?,
    attribute FitToPage { xsd:boolean }?,
    attribute ViewThreshold { xsd:double {minInclusive="5" maxInclusive="4000"} }?,
    attribute Locked { xsd:boolean }?,
    attribute ItemLayer { xsd:string }?,
    attribute PageIndex { xsd:short }?,
    element Properties {
        element GuideColor { InDesignUIColorType_TypeDef }?&
        element Label { element KeyValuePair { KeyValuePair_TypeDef }* }
    }
}
```

#### IDML Example 34. Guide

```
<Guide Self="ue4" Orientation="Horizontal" Location="72" FitToPage="true" View-
Threshold="5" Locked="false" ItemLayer="ub1" PageIndex="1">
<Properties>
```

```

<GuideColor type="enumeration">Cyan</GuideColor>
</Properties>
</Guide>

```

**Table 46. Guide Properties Represented as Attributes**

Name	Type	Req	Description
FitToPage	boolean	no	If true, horizontal orientation guides stop at the edges of the specified page. If false, the guides extends across the width of the spread and into the pasteboard area.
ItemLayer	string	no	The layer that the guide is on.
Location	double	no	The location at which to place the guide relative to the current ruler zero point.
Locked	boolean	no	If true, the guide is locked.
Orientation	HorizontalOrVertical_Enum_Value	no	The orientation of the guide. Can be Horizontal or Vertical.
PageIndex	short	no	The index of the page containing the guide.
ViewThreshold	double	no	The view magnification as a percentage below which guides are no longer displayed. (Range: 5.0 to 4000.0)

**Table 47. Guide Properties Represented as Elements**

Name	Type	Req	Description
GuideColor	InDesignUIColor-Type_TypeDef	no	The color of the guide, specified either as an array of three ListItem elements, each containing a value from 0 to 255 and representing R, G, and B values, or as a UIColor enumeration.

### 10.3.8 Transparency Flattener Settings

The appearance of transparency (including the Drop Shadow, Inner Shadow, Outer Glow, Inner Glow, Bevel and Emboss, Satin, and Feather effects) when you print or export an InDesign document depends on the flattener settings for the spread and/or the document (the exact appearance of the transparent objects on the printed pages or in the exported file depends on the capabilities of the printer or file format). The flattener settings of the document apply to all objects unless the objects exist on a spread which has been assigned its own flattener settings; in that case, the flattener settings of the spread override those of the document. In IDML, these settings are represented by the `<FlattenerPreference>` element.

#### Schema Example 41. FlattenerPreference

```

FlattenerPreference_Object = element FlattenerPreference {
    attribute Self { xsd:string },
    attribute LineArtAndTextResolution { xsd:double }?,
    attribute GradientAndMeshResolution { xsd:double }?,
    attribute ClipComplexRegions { xsd:boolean }?,
    attribute ConvertAllStrokesToOutlines { xsd:boolean }?,
    attribute ConvertAllTextToOutlines { xsd:boolean }?,
    element Properties {

```

```

element RasterVectorBalance {
    (enum_type, FlattenerLevel_EnumValue ) |
    (double_type, xsd:double {minInclusive="0" maxInclusive="100"} )
}?
}
?
}

```

The following example shows a custom flattener setting applied to a spread.

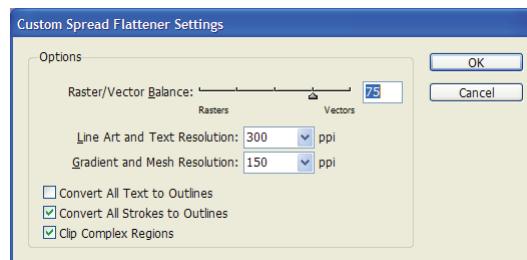
**IDML Example 35. FlattenerPreference**

```

<Spread Self="ub8" FlattenerOverride="Custom">
    <FlattenerPreference Self="ub8FlattenerPreference1"
        LineArtAndTextResolution="300" GradientAndMeshResolution="150"
        ClipComplexRegions="true" ConvertAllStrokesToOutlines="true"
        ConvertAllTextToOutlines="false">
        <Properties>
            <RasterVectorBalance type="double">75</RasterVectorBalance>
        </Properties>
    </FlattenerPreference>
</Spread>

```

*Figure 28. FlattenerPreference*



## 10.4 Stories

A story, or “text flow” is the basic text container in an InDesign document; all text exists inside stories. Stories are associated with at least one text frame or text path, and can span any number of linked text frames or text paths in a document. Text frames and text paths are page items, and are discussed in the section “Spreads and Master Spreads”.

In an IDML package, the XML documents representing stories are stored inside the “Stories”. Each file contains a single `<Story>` element. Story files use the naming convention described in “IDML Component Names.” The root element of a `Story.xml` file is the `<Story>` element, and each story file stores a single `<Story>` element.

The `<Story>` element is very complex. The schema of `<Story>` element describes more than 200 simple attributes and more than 40 complex attributes that can appear in the `<Properties>` element of the story. In addition, the `<Story>` element can contain other child elements, including elements corresponding to inline or anchored frames, tables, notes, hyperlinks, and footnotes.

That said, most of these attributes and elements of story are optional. You do not need to construct all of them to assemble a new story for use in an IDML package. A story can be as simple as the following example:

### IDML Example 36. Story File in an IDML Package

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<idPkg:Story xmlns:idPkg="http://ns.adobe.com/AdobeInDesign/idml/1.0/packaging">
  <Story Self="ucb">
    <ParagraphStyleRange>
      <CharacterStyleRange>
        <Content>Hello World</Content>
      </CharacterStyleRange>
    </ParagraphStyleRange>
  </Story>
</idPkg:Story>
```

A version of the same element elsewhere in an IDML package (in `designmap.xml`, for example) would be the same as the `<Story>` element in the example above.

The `<Story>` object can contain the default text formatting for the story. Individual instances of local text formatting (i.e., formatting which was not applied by paragraph and character styles) can also appear in the story file. This local formatting will always appear inside a `<Paragraph-StyleRange>` or `<CharacterStyleRange>` element see “Local Formatting vs. Styles.”

Paragraph and character style formatting is applied using references to the `Self` attribute of `<ParagraphStyle>` and `<CharacterStyle>` elements. In an IDML package, these elements are stored in the `Styles.xml` file inside the Resources folder. The formatting attributes of the styles are stored within that file, not within the `<Story>` element (see “Styles”).

### 10.4.1 Story vs. XMLStory

The `<XMLStory>` element (which is in the `BackingStory.xml` file in the XML folder of an IDML package) represents unplaced XML elements (i.e., an XML element in the XML structure of the InDesign document, that has not yet been placed in the layout) can have all of the same attributes and child elements that a `<Story>` element can have.

#### 10.4.2 Story vs. AssignedStory

An `<AssignedStory>` element represents an InCopy file, not a `<Story>` element. The `<AssignedStory>` element has an attribute called `AssignedStory`. The value of this attribute can contain a reference to a `<story>` element, or to a page item element such as a `<Rectangle>`.

#### 10.4.3 Referring to a Story

A `<Story>` element is referred to by its unique ID—the value of its `Self` attribute. `<TextFrame>` and `<TextPath>` elements refer to the story using the `ParentStory` attribute, `<Link>` elements refer to the story using the `Parent` attribute. In addition, `<Document>` elements use the `InstanceList` attribute to refer to an index story. Third party developers may create their own references to stories.

#### 10.4.4 Adding a Story

To add a new story to a IDML file or package, you add a `<story>` element. In an IDML package, this element can be stored in an XML file in the Stories folder. This file can have any valid XML file name—it does not have to match the naming convention that InDesign uses when you export IDML. A single XML file can contain multiple `<Story>` elements. When you refer to stories in other places in the IDML file, you use the `Self` attribute of the `<Story>` element in the file, not the file name.

When you add a story, you need to make certain that:

- One of the `Spread.xml` files in the IDML package contains a `<TextFrame>` or `<TextPath>` element that contains a reference to the story. If no `<TextFrame>` or `<TextPath>` element refers to the story, the story will not be imported, and will not appear in the layout.
- The paragraph styles, character styles, colors, and other references used in the story have already been defined in files in the Resources folder of the IDML package. If they have not been defined, you'll need to add these definition files to the package.

**Note:** In general, following the same naming scheme and storage approach as InDesign uses is good practice, but it's not required. Using the unique identifier of the story in the name of the file makes it easier to find the corresponding `<Story>` element, and storing a single `<Story>` per file makes it easier to change text content.

#### 10.4.5 Local Formatting vs. Styles

In InDesign, you can use the typesetting controls (in the Control panel, Character panel, or on the Type menu, for example) to apply formatting to text without using paragraph or character styles. When you do this, you're applying *local* formatting. InDesign represents local formatting using attributes and elements of the `<Story>`, `<ParagraphStyleRange>`, and `<CharacterStyleRange>` elements.

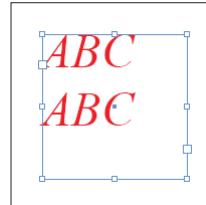
Most of the time, it's far better to use character and paragraph styles to apply text formatting. Using styles means that the formatting definitions can be changed in one place, rather than being spread throughout the text in a file, and applying a style is much faster than applying a series of local formatting changes. In addition, character and paragraph styles can be included in object, table, and cell styles.

In the following example, the first `<ParagraphStyleRange>` element uses local formatting (specified as part of the `<CharacterStyleRange>` element); the second applies the same formatting using a paragraph style (specified as the `AppliedParagraphStyle` attribute). In this example, we've applied only three local formatting attributes to illustrate the point, but completely replacing the formatting applied by a paragraph style could entail the specification of more than two hundred attributes and elements. The two paragraphs have the same formatting, but the second example is a much more efficient use of IDML. Local formatting persists inside a given `<ParagraphStyleRange>` element until the next `<ParagraphStyleRange>`, and within a `<CharacterStyleRange>` until the next `<CharacterStyleRange>` or `<ParagraphStyleRange>`.

#### IDML Example 37. Local Formatting vs. Styles

```
<ParagraphStyleRange AppliedParagraphStyle="ParagraphStyle\kNormalParagraphStyle">
  <CharacterStyleRange AppliedCharacterStyle="CharacterStyle\k[No character style]">
    FillColor="Color\cRed" FontStyle="Italic" PointSize="24">
      <Content>ABC</Content>
      <br/>
    </CharacterStyleRange>
  </ParagraphStyleRange>
<ParagraphStyleRange AppliedParagraphStyle="ParagraphStyle\cStyle1">
  <CharacterStyleRange AppliedCharacterStyle="CharacterStyle\k[No character
  style]">
    <Content>ABC</Content>
  </CharacterStyleRange>
</ParagraphStyleRange>
```

Figure 29. Local Formatting vs. Styles



#### 10.4.6 Common Text Properties

Just as all text objects in the InDesign scripting DOM share a large number of properties, all text elements in an IDML package share a large set of common attributes and elements. These attributes and properties can appear in a `<Story>`, `<ParagraphStyleRange>`, or `<Character-StyleRange>` element inside a `<Story>` element, and in a `<ParagraphStyle>` or `<Character-Style>` elements.

To avoid repeating these attributes and elements in every text-related element reference in this specification, we'll describe them in this section, and refer to these descriptions from other sections.

**Table 48. Common Text Properties Represented as Attributes**

Name	Type	Req	Description
AppliedCharacter-Style	string	no	The character style applied to the text.
AppliedConditions	string (space separated list of conditions)	no	The applied conditions.
AppliedLanguage	string	no	The language of the text. A reference to the <code>Self</code> attribute of a language.
AppliedParagraph-Style	string	no	The paragraph style applied to the text.
AutoLeading	double	no	The percent of the type size to use for auto leading. (Range: 0 to 500).
AutoTcy	short	no	The number of half-width characters at or below which the characters automatically run horizontally in vertical text.
AutoTcyInclude-Roman	boolean	no	If true, auto tcy includes Roman characters.
BaselineShift	double	no	The baseline shift applied to the text.
BulletsAlignment	ListAlignment_EnumValue	no	The alignment of the bullet character. Can be <code>LeftAlign</code> (Align left), <code>CenterAlign</code> (Align center), or <code>RightAlign</code> (Align right).
BulletsAnd-NumberingListType	ListType_EnumValue	no	List type for bullets and numbering. Can be <code>NoList</code> (No list), <code>BulletList</code> (Bullet list), or <code>NumberedList</code> (Numbered list).
BulletsTextAfter	string	no	The text after string expression for bullets.
BunriKinshi	boolean	no	If true, adds the double period (..), ellipse (...), and double hyphen (--) to the selected kinsoku set. Note: Valid only when a kinsoku set is in effect.
Capitalization	Capitalization_EnumValue	no	The capitalization scheme. Can be <code>Normal</code> (Do not change the capitalization of the text), <code>SmallCaps</code> (Use small caps for lowercase letters), <code>AllCaps</code> (Use all uppercase letters), or <code>CapToSmallCap</code> (Use OpenType small caps).

Name	Type	Req	Description
Character-Alignment	Character-Alignment_Enum-Value	no	The alignment of small characters to the largest character in the line. Can be AlignBaseline (Aligns small characters in a line to the large character), AlignEmTop (Aligns small characters in horizontal text to the top of the em box of large characters In vertical text, aligns characters to the right of the em box), AlignEmCenter (Aligns small characters to the center of the em box of large characters), AlignEmBottom (Aligns small characters in horizontal text to the bottom of the em box of large characters In vertical text, aligns characters to the left of the em box), AlignICFTop (Aligns small characters in horizontal text to the top of the ICF of large characters In vertical text, aligns characters to the right of the ICF), or AlignICFBottom (Aligns small characters in horizontal text to the bottom of the ICF of large characters In vertical text, aligns characters to the left of the ICF).
Character-Direction	Character-Direction_Enum-Value	no	The direction of the character. Can be Default-Direction (Default direction), LeftToRight-Direction (Left to right direction), or RightToLeftDirection (Right to left direction).
CharacterRotation	double	no	The rotation angle (in degrees) of individual characters. Note: The rotation is counterclockwise.
CjkGridTracking	boolean	no	If true, uses grid tracking to track non-Roman characters in CJK grids.
Composer	string	no	The text composer to use to compose the text.
DesiredGlyph-Scaling	double	no	The desired width (as a percentage) of individual characters. (Range: 50 to 200)
DesiredLetter-Spacing	double	no	The desired letter spacing, specified as a percentge of the built-in space between letters in the font. (Range: -100 to 500)
DesiredWord-Spacing	double	no	The desired word spacing, specified as a percentage of the font word space value. (Range: 0 to 1000)
DiacriticPosition	Diacritic-Position_Enum-Value	no	Position of diacritical characters. Can be DefaultPosition (Default position), LoosePosition (Loose position), MediumPosition (Medium position), TightPosition (Tight position), or OpentypePosition (OpenType position).
DigitsType	DigitsType_Enum-Value	no	The type of digits to use. Can be Default-Digits (Default digits), ArabicDigits (Arabic digits), HindiDigits (Hindi digits), or Farsi-Digits (Farsi digits).
DropCapCharacters	short	no	The number of characters to drop for a drop cap.
DropCapLines	short	no	The number of lines to drop for a drop cap.

Name	Type	Req	Description
DropcapDetail	int	no	The detailed size and positioning of the drop cap.
EndJoin	OutlineJoin_EnumValue	no	The stroke join type applied to the characters of the text. Can be MiterEndJoin (Miter end join), RoundEndJoin (Rounded end join), or BevelEndJoin (Beveled end join).
FillColor	string	no	The swatch (color, gradient, tint, or mixed ink) applied to the fill of the text, as a reference to the Self attribute of the swatch.
FillTint	double	no	The tint (as a percentage) of the fill color of the Paragraph. (To specify a tint percentage, use a number in the range of 0 to 100; to use the inherited or overridden value, use -1.)
FirstLineIndent	double	no	The amount to indent the first line.
FontStyle	string	no	The name of the font style.
GlyphForm	AlternateGlyphForms_EnumValue	no	The glyph variant to substitute for standard glyphs. Can be None (Does not use an alternate form), TraditionalForm (Uses the traditional variant), ExpertForm (Uses the expert variant), JIS78Form (Uses the JIS78 variant), JIS83Form (Uses the JIS83 variant), MonospacedHalfWidthForm (Uses the monospaced half-width variant), ThirdWidthForm (Uses the third-width variant), QuarterWidthForm (Uses the quarter-width variant), NLCForm (Uses the NLC variant), ProportionalWidthForm (Substitutes proportional glyphs for half-width and full-width glyphs), FullWidthForm (Uses the full-width variant), JIS04Form (Uses the JIS04 variant), or JIS90Form (Uses the JIS90 variant).
GotoNextX	GotoNextX_EnumValue	no	A break character that forces text to the next page or column.
GradientFillAngle	double	no	The angle of a linear gradient applied to the fill of the text. (Range: -180 to 180)
GradientFillLength	double	no	The length (for a linear gradient) or radius (for a radial gradient) applied to the fill of the text.
GradientFillStart	UnitPointType_TypeDef	no	The starting point (in page coordinates) of a gradient applied to the fill of the text, in the format [x, y].
GradientStrokeAngle	double	no	The angle of a linear gradient applied to the stroke of the text. (Range: -180 to 180)
GradientStrokeLength	double	no	The length (for a linear gradient) or radius (for a radial gradient) applied to the stroke of the text.
GradientStrokeStart	UnitPointType_TypeDef	no	The starting point (in page coordinates) of a gradient applied to the stroke of the text, in the format [x, y].
GridAlignFirstLineOnly	boolean	no	If true, aligns only the first line to the frame grid or baseline grid. If false, aligns all lines to the grid.

Name	Type	Req	Description
GridAlignment	GridAlignment_EnumValue	no	The alignment to the frame grid or baseline grid. Can be None (Lines are not aligned to the grid), AlignBaseline (Aligns the text baseline to the grid), AlignEmTop (Aligns the top of the em box to the grid), AlignEmCenter (Aligns the center of the em box to the grid), AlignEmBottom (Aligns the bottom of the em box to the grid), AlignICFTop (Aligns the top of the ICF box to the grid), or AlignICFBottom (Aligns the bottom of the ICF box to the grid).
GridGyoudori	short	no	The manual gyoudori setting.
HorizontalScale	double	no	The horizontal scaling applied to the text as a percentage of its current size. (Range: 1 to 1000)
HyphenWeight	short	no	The relative desirability of better spacing vs. fewer hyphens. A lower value results in greater use of hyphens. (Range: 0 to 100)
HyphenateAcross-Columns	boolean	no	If true, allows the last word in a text column to be hyphenated.
HyphenateAfter-First	short	no	The minimum number of letters at the beginning of a word that can be broken by a hyphen.
HyphenateBefore-Last	short	no	The minimum number of letters at the end of a word that can be broken by a hyphen.
Hyphenate-CapitalizedWords	boolean	no	If true, allows hyphenation of capitalized words.
HyphenateLadder-Limit	short	no	The maximum number of hyphens that can appear on consecutive lines. To specify unlimited consecutive lines, use zero.
HyphenateLastWord	boolean	no	If true, allows hyphenation in the last word in a paragraph. Note: Valid only when hyphenation is true.
HyphenateWords-LongerThan	short	no	The minimum number of letters a word must have in order to qualify for hyphenation.
Hyphenation	boolean	no	If true, allows hyphenation.
HyphenationZone	double	no	The amount of white space allowed at the end of a line of non-justified text before hyphenation begins. Note: Valid when composer is single-line composer.
IgnoreEdge-Alignment	boolean	no	If true, ignores optical edge alignment for the paragraph.
Jidori	short	no	The number of grid squares in which to arrange the text.

Name	Type	Req	Description
Justification	Justification_EnumValue	no	The paragraph alignment. Can be <code>LeftAlign</code> (Left aligns the text), <code>CenterAlign</code> (Center aligns the text), <code>RightAlign</code> (Right aligns the text), <code>LeftJustified</code> (Justifies the text and left aligns the last line in the paragraph), <code>RightJustified</code> (Justifies the text and right aligns the last line in the paragraph), <code>CenterJustified</code> (Justifies text text and center aligns the last line in the paragraph), <code>FullyJustified</code> (Justifies the text, including the last line in the paragraph), <code>ToBindingSide</code> (Aligns text to the binding spine of the page or spread), or <code>AwayFromBindingSide</code> (Aligns text to the side opposite the binding spine of the page).
Kashidas	Kashidas_EnumValue	no	Use of Kashidas for justificationCan be <code>DefaultKashidas</code> (Default kashidas), or <code>KashidasOff</code> (Kashidas off).
KeepAllLines-Together	boolean	no	If true, keeps all lines of the paragraph together. If false, allows paragraphs to break across pages or columns.
KeepFirstLines	short	no	The minimum number of lines to keep together in a paragraph before allowing a page break.
KeepLastLines	short	no	The minimum number of lines to keep together in a paragraph after a page break.
KeepLinesTogether	boolean	no	If true, keeps a specified number of lines together when the paragraph breaks across columns or text frames.
KeepRuleAboveIn-Frame	boolean	no	If true, forces the rule above the paragraph to remain in the frame bounds. Note: Valid only when rule above is true.
KeepWithNext	short	no	The minimum number of lines to keep with the next paragraph.
KntenAlignment	KntenAlignment_EnumValue	no	The alignment of knten characters relative to the parent characters. Can be <code>AlignKntenLeft</code> (Aligns knten with the left of parent characters), or <code>AlignKntenCenter</code> (Aligns knten with the center of parent characters).
KntenCharacter-Set	KntenCharacter-Set_EnumValue	no	The character set used for the custom knten character. Note: Valid only when knten kind is custom. Can be <code>CharacterInput</code> (Character input), <code>ShiftJIS</code> (Shift JIS), <code>JIS</code> (JIS), <code>Kuten</code> (Kuten), or <code>Unicode</code> (Unicode).
KntenCustom-Character	string	no	The character used for knten. Note: Valid only when knten kind is custom.
KntenFontSize	double	no	The size (in points) of knten characters.

Name	Type	Req	Description
KntenKind	KntenCharacter_EnumValue	no	The style of knten characters. Can be None (Does not use knten), KntenSesameDot (Uses the knten sesame dot), KntenWhiteSesameDot (Uses the knten white sesame dot), KntenBlackCircle (Uses the knten black circle), KntenWhiteCircle (Uses the knten white circle), KntenBlackTriangle (Uses the knten black triangle), KntenWhiteTriangle (Uses the knten white triangle), KntenBullseye (Uses the knten bullseye), KntenFisheye (Uses the knten fisheye), KntenSmallBlackCircle (Uses the knten small black circle), KntenSmallWhiteCircle (Uses the knten small white circle), or Custom (Uses a custom knten style).
KntenOverprint-Fill	Adornment-Overprint_EnumValue	no	The method of overprinting the knten fill. Can be Auto (Uses auto overprint), OverprintOn (Turns on overprint), or OverprintOff (Turns off overprint).
KntenOverprint-Stroke	Adornment-Overprint_EnumValue	no	The method of overprinting the knten stroke. Can be Auto (Uses auto overprint), OverprintOn (Turns on overprint), or OverprintOff (Turns off overprint).
KntenPlacement	double	no	The distance between knten characters and their parent characters.
KntenPosition	RubyKnten-Position_EnumValue	no	The knten position relative to the parent character. Can be AboveRight (Places knten or ruby to the right and above the parent character), or BelowLeft (Places ktenen or ruby to the left and below the parent character).
KntenStrokeTint	double	no	The stroke tint (as a percentage) of knten characters. (Range: 0 to 100)
KntenTint	double	no	The fill tint (as a percentage) of knten characters. (Range: 0 to 100)
KntenWeight	double	no	The stroke weight (in points) of knten characters.
KntenXScale	double	no	The horizontal size of knten characters as a percent of the original size.
KntenYScale	double	no	The vertical size of ktenen characters as a percent of the original size.
KerningMethod	string	no	The type of pair kerning.
KerningValue	double	no	The amount of space to add or remove between characters, specified in thousands of an em.
KeyboardDirection	Character-Direction_EnumValue	no	The keyboard direction of the characterCan be DefaultDirection (Default direction), LeftToRightDirection (Left to right direction), or RightToLeftDirection (Right to left direction).

Name	Type	Req	Description
KinsokuHangType	KinsokuHangTypes_EnumValue	no	The type of hanging punctuation to allow. Note: Valid only when a kinsoku set is in effect. Can be None (Disables hanging punctuation), KinsokuHangRegular (Enables hanging punctuation and allows punctuation marks to be placed on or outside the text frame but allows burasagari characters to hang as little as possible Note: Differs for justified and nonjustified text For information on justification, see line alignment), or KinsokuHangForce (Enables hanging punctuation but forces hanging punctuation outside the text frame and does not allow the punctuation to be placed on the text frame).
KinsokuType	KinsokuType_EnumValue	no	The type of kinsoku processing for preventing kinsoku characters from beginning or ending a line. Note: Valid only when a kinsoku set is defined. Can be KinsokuPushInFirst (Attempts to move characters to the previous line; if the push-in is not possible, pushes characters to the next line), KinsokuPushOutFirst (Attempts to move characters to the next line; if the push-out is not possible, pushes characters to the previous line), KinsokuPushOutOnly (Always moves characters to the next line Does not attempt a push-in), or KinsokuPrioritizeAdjustmentAmount (The kinsoku prioritize adjustment amount).
LastLineIndent	double	no	The amount to indent the last line in the Paragraph.
LeadingAki	double	no	The amount of space before each character.
LeadingModel	LeadingModel_EnumValue	no	The point from which leading is measured from line to line. Can be LeadingModelRoman (Measures the space between type baselines), LeadingModelAkiBelow (Measures the space between lines from the aki below), LeadingModelAkiAbove (Measures the space between lines from the aki above), LeadingModelCenter (Measures the space between the character center points), or LeadingModelCenterDown (Center down leading model).
LeftIndent	double	no	The width of the left indent.
Ligatures	boolean	no	If true, replaces specific character combinations (e.g., fl, fi) with ligature characters.
MaximumGlyphScaling	double	no	The maximum width (as a percentage) of individual characters. (Range: 50 to 200)
MaximumLetterSpacing	double	no	The maximum letter spacing, specified as a percentge of the built-in space between letters in the font. (Range: -100 to 500) Note: Valid only when text is justified.
MaximumWordSpacing	double	no	The maximum word spacing, specified as a percentage of the font word space value. Note: Valid only when text is justified. (Range: 0 to 1000)

Name	Type	Req	Description
MinimumGlyphScaling	double	no	The minimum width (as a percentage) of individual characters. (Range: 50 to 200)
MinimumLetterSpacing	double	no	The minimum letter spacing, specified as a percentage of the built-in space between letters in the font. (Range: -100 to 500) Note: Valid only when text is justified.
MinimumWordSpacing	double	no	The minimum word spacing, specified as a percentage of the font word space value. Note: Valid only when text is justified. (Range: 0 to 1000)
MiterLimit	double	no	The limit of the ratio of stroke width to miter length before a miter (pointed) join becomes a bevel (squared-off) join.
NoBreak	boolean	no	If true, keeps the text on the same line.
NumberingAlignment	ListAlignment_EnumValue	no	The alignment of the number. Can be LeftAlign (Align left), CenterAlign (Align center), or RightAlign (Align right).
NumberingApplyRestartPolicy	boolean	no	If true, apply the numbering restart policy.
NumberingContinue	boolean	no	Continue the numbering at this level.
NumberingExpression	string	no	The number string expression for numbering.
NumberingLevel	int	no	The level of the paragraph.
NumberingStartAt	int	no	Determines starting number in a numbered list.
OTFContextualAlternate	boolean	no	If true, uses contextual alternate forms in OpenType fonts.
OTFDiscordianaryLigature	boolean	no	If true, uses discretionary ligatures in OpenType fonts.
OTFFigureStyle	OTFFigureStyle_EnumValue	no	The figure style in OpenType fonts. Can be TabularLining (Use monspaced lining figures), ProportionalOldstyle (Use proportional width oldstyle figures), ProportionalLining (Use proportional width lining figures), TabularOldstyle (Use monospaced oldstyle figures), or Default (Use the default figure style for the font).
OTFFraction	boolean	no	If true, uses fractions in OpenType fonts.
OTFHVKana	boolean	no	If true, switches hiragana fonts, which have different glyphs for horizontal and vertical.
OTFHistorical	boolean	no	If true, use historical forms in OpenType fonts.
OTFJustificationAlternate	boolean	no	Whether to use justification alternate forms in OpenType fonts
OTFLocale	boolean	no	If true, uses localized forms in OpenType fonts.
OTFMark	boolean	no	If true, uses mark positioning in OpenType fonts.
OTFOrdinal	boolean	no	If true, uses ordinals in OpenType fonts.

Name	Type	Req	Description
OTFOverlapSwash	boolean	no	If true, use overlapping swash forms in OpenType fonts.
OTFProportional-Metrics	boolean	no	If true, kerns according to proportional CJK metrics in OpenType fonts.
OTFRomanItalics	boolean	no	If true, applies italics to half-width alphanumeric characters.
OTFSashedZero	boolean	no	If true, use a slashed zeroes in OpenType fonts.
OTFSretched-Alternate	boolean	no	Whether to use stretched alternate forms in OpenType fonts.
OTFStylistic-Alternate	boolean	no	If true, use stylistic alternate forms in OpenType fonts.
OTFStylisticSets	int	no	The stylistic sets to use in OpenType fonts.
OTFSwash	boolean	no	If true, uses swash forms in OpenType fonts.
OTFTitling	boolean	no	If true, uses titling forms in OpenType fonts.
OverprintFill	boolean	no	If true, the fill color of the characters will overprint.
OverprintStroke	boolean	no	If true, the stroke of the characters will overprint.
PageNumberType	PageNumberType_EnumValue	no	The type of the page number marker. Can be AutoPageNumber, NextPageNumber, or PreviousPageNumber.
Paragraph-Direction	Paragraph-Direction_EnumValue	no	The paragraph composition direction. Can be LeftToRightDirection (Left to Right paragraph direction), or RightToLeftDirection (Right to Left paragraph direction).
ParagraphGyoudori	boolean	no	If true, the gyoudori mode applies to the entire paragraph. If false, the gyoudori mode applies to each line in the paragraph.
Paragraph-Justification	Paragraph-Justification_EnumValue	no	Paragraph justificationCan be Default-Justification (Default justification), ArabicJustification (Arabic justification), or NaskhJustification (Naskh justification).
PointSize	double	no	The text size.

Name	Type	Req	Description
Position	Position_EnumValue	no	The text position relative to the baseline. Can be Normal (Normal position), Superscript (Superscripts the text), Subscript (Subscripts the text), OTSuperscript (For OpenType fonts, uses—if available—raised glyphs that are sized correctly relative to the surrounding characters), OTSubscript (For OpenType fonts, uses—if available—lowered glyphs that are sized correctly relative to the surrounding characters), OTNumerator (For OpenType fonts, shrinks the text but keeps the top of the characters aligned with the top of the main text Note: Valid only for numeric characters), or OTDenominator (For OpenType fonts, shrinks the text but keeps text on the main text baseline Note: Valid only for numeric characters).
PositionalForm	PositionalForms_EnumValue	no	The OpenType positional form. Can be None (None), Calculate (Calculated forms), Initial (Initial form), Medial (Medial form), Final (Final form), or Isolated (Isolated form).
Rensuuji	boolean	no	If true, disallows line breaks in numbers. If false, lines can break between digits in multi-digit numbers.
RightIndent	double	no	The width of the right indent.
RotateSingleByte-Characters	boolean	no	If true, rotates Roman characters in vertical text.
RubyAlignment	RubyAlignments_EnumValue	no	The ruby alignment. Can be RubyLeft (Aligns ruby with the left-most character in the parent text), RubyCenter (Centers ruby relative to the parent text), RubyRight (Aligns ruby with the right-most character in the parent text), RubyFullJustify (Justifies ruby across the parent text), RubyJIS (Ruby JIS), RubyEqualAki (Ruby equal aki), or Ruby1Aki (Ruby 1 aki).
RubyAutoAlign	boolean	no	If true, auto aligns ruby.
RubyAutoScaling	boolean	no	If true, automatically scales ruby to the specified percent of parent text size. For information on specifying a percent, see ruby parent scaling percent.
RubyAutoTcyAutoScale	boolean	no	If true, automatically scales glyphs in auto tcy (tate-chuu-yoko) in ruby to fit one em.
RubyAutoTcyDigits	short	no	The number of digits included in auto tcy (tate-chuu-yoko) in ruby.
RubyAutoTcy-IncludeRoman	boolean	no	If true, includes Roman characters in auto tcy (tate-chuu-yoko) in ruby.
RubyFlag	int	no	If true, ruby is on.
RubyFontSize	double	no	The size (in points) of ruby characters.
RubyOpenTypePro	boolean	no	If true, uses OpenType Pro fonts for ruby.

Name	Type	Req	Description
RubyOverhang	boolean	no	If true, constrains ruby overhang to the specified amount. For information on specifying an amount, see ruby parent overhang amount.
RubyOverprintFill	Adornment-Overprint_Enum-Value	no	The method of overprinting the ruby fill. Can be Auto (Uses auto overprint), OverprintOn (Turns on overprint), or OverprintOff (Turns off overprint).
RubyOverprint-Stroke	Adornment-Overprint_Enum-Value	no	The method of overprinting the ruby stroke. Can be Auto (Uses auto overprint), OverprintOn (Turns on overprint), or OverprintOff (Turns off overprint).
RubyParent-OverhangAmount	RubyOverhang_EnumValue	no	The amount by which ruby characters can overhang the parent text. Can be None (Does not allow ruby overhang), RubyOverhangOneRuby (Ruby overhang is one ruby), RubyOverhang-HalfRuby (Ruby overhang is one-half ruby), RubyOverhangOneChar (Ruby overhang is the size of one character), RubyOverhangHalfChar (Ruby is overhang one-half the size of one character), or RubyOverhangNoLimit (There is no ruby overhang size limit).
RubyParent-ScalePercent	double	no	The amount (as a percentage) to scale the parent text size to determine the ruby text size.
RubyParentSpacing	RubyParent-Space_EnumValue	no	The ruby spacing relative to the parent text. Can be RubyParentNoAdjustment (Does not base ruby spacing on parent text), RubyParent-BothSides (Ruby parent both sides), Ruby-Parent121Aki (Ruby parent 121 aki), Ruby-ParentEqualAki (Applies the parent text aki to the ruby characters), or RubyParentFull-Justify (Justifies ruby characters to both edges of the parent text).
RubyPosition	RubyKeten-Position_Enum-Value	no	The position of ruby characters relative to the parent text. Can be AboveRight (Places keten or ruby to the right and above the parent character), or BelowLeft (Places keten or ruby to the left and below the parent character).
RubyString	string	no	The ruby string contents.
RubyStrokeTint	double	no	The stroke tint (as a percentage) of ruby characters.
RubyTint	double	no	The tint (as a percentage) of the ruby fill color. (Range: 0 to 100)
RubyType	RubyTypes_Enum-Value	no	The ruby type. Can be GroupRuby (Provides ruby for a group of characters), or Per-CharacterRuby (Provides ruby for each individual character in the group).
RubyWeight	double	no	The stroke weight (in points) of ruby characters.
RubyXOffset	double	no	The amount of horizontal space between ruby and parent characters.

Name	Type	Req	Description
RubyXScale	double	no	The horizontal size of ruby characters, specified as a percent of the original size.
RubyYOffset	double	no	The amount of vertical space between ruby and parent characters.
RubyYScale	double	no	The vertical size of ruby characters, specified as a percent of the original size.
RuleAbove	boolean	no	If true, places a rule above the paragraph.
RuleAboveGap-Overprint	boolean	no	If true, the stroke gap of the paragraph rule above will overprint. Note: Valid only the rule above type is not solid.
RuleAboveGapTint	double	no	The tint (as a percentage) of the stroke gap color of the paragraph rule. (Range: 0 to 100) Note: Valid only when the rule above type is not solid.
RuleAboveLeft-Indent	double	no	The distance to indent the left edge of the paragraph rule above (based on either the text width or the column width of the first line in the paragraph).
RuleAboveLine-Weight	double	no	The line weight of the rule above
RuleAboveOffset	double	no	The amount to offset the paragraph rule above from the baseline of the first line the paragraph.
RuleAbove-Overprint	boolean	no	If true, the paragraph rule above will overprint.
RuleAboveRight-Indent	double	no	The distance to indent the right edge of the paragraph rule above (based on either the text width or the column width of the first line in the paragraph).
RuleAboveTint	double	no	The tint (as a percentage) of the paragraph rule above. (Range: 0 to 100)
RuleAboveWidth	RuleWidth_Enum-Value	no	The basis (text width or column width) used to calculate the width of the paragraph rule above. Can be TextWidth (Makes the paragraph rule above the width of the first line of text in the paragraph), or ColumnWidth (Makes the rule the width of the column).
RuleBelow	boolean	no	If true, applies a paragraph rule below.
RuleBelowGap-Overprint	boolean	no	If true, the gap color of the rule below will overprint.
RuleBelowGapTint	double	no	The tint (as a percentage) of the stroke gap color of the paragraph rule below. (Range: 0 to 100) Note: Valid only when the paragraph rule below type is not solid.
RuleBelowLeft-Indent	double	no	The distance to indent the left edge of the paragraph rule below (based on either the text width or the column width of the last line in the paragraph).

Name	Type	Req	Description
RuleBelowLine-Weight	double	no	The line weight of the rule below
RuleBelowOffset	double	no	The amount to offset the the paragraph rule below from the baseline of the last line of the paragraph.
RuleBelow-Overprint	boolean	no	If true, the rule below will overprint.
RuleBelowRight-Indent	double	no	The distance to indent the right edge of the paragraph rule below (based on either the text width or the column width of the last line in the paragraph).
RuleBelowTint	double	no	The tint (as a percentage) of the paragraph rule below. (Range: 0 to 100)
RuleBelowWidth	RuleWidth_Enum-Value	no	The basis (text width or column width) used to calculate the width of the paragraph rule below. Can be <code>TextWidth</code> (Makes the paragraph rule above the width of the first line of text in the paragraph), or <code>ColumnWidth</code> (Makes the rule the width of the column).
ScaleAffectsLine-Height	boolean	no	If true, the line changes size when characters are scaled.
ShataiAdjust-Rotation	boolean	no	If true, applies shatai rotation.
ShataiAdjustTsume	boolean	no	If true, adjusts shatai tsume.
ShataiDegreeAngle	double	no	The shatai lens angle (in degrees).
Shatai-Magnification	double	no	The amount (as a percentage) of shatai obliquing to apply.
SingleWord-Justification	SingleWord-Justification_EnumValue	no	The alignment to use for lines that contain a single word. Can be <code>LeftAlign</code> (Left aligns the word), <code>CenterAlign</code> (Center aligns the word), <code>RightAlign</code> (Right aligns the word), or <code>FullyJustified</code> (Fully justifies the word).
Skew	double	no	The skew angle of the text. (Range: -85 to 85)
SpaceAfter	double	no	The height of the paragraph space below.
SpaceBefore	double	no	The height of the paragraph space above.
StartParagraph	StartParagraph_EnumValue	no	The location at which to start the paragraph. Can be <code>Anywhere</code> (Starts in the next available space), <code>NextColumn</code> (Starts at the top of the next column), <code>NextFrame</code> (Starts at the top of the next text frame in the thread), <code>NextPage</code> (Starts at the top of the next page), <code>NextOddPage</code> (Starts at the top of the next odd-numbered page), or <code>NextEvenPage</code> (Starts at the top of the next even-numbered page).
StrikeThroughGap-Overprint	boolean	no	If true, the gap color of the strikethrough stroke will overprint. Note: Valid when strike through type is not solid.

Name	Type	Req	Description
StrikeThroughGap-Tint	double	no	The tint (as a percentage) of the strikethrough stroke gap color. (Range: 0 to 100) Note: Valid when strike through type is not solid.
StrikeThrough-Offset	double	no	The amount by which to offset the strikethrough stroke from the text baseline.
StrikeThrough-Overprint	boolean	no	If true, the strikethrough stroke will overprint.
StrikeThroughTint	double	no	The tint (as a percentage) of the strikethrough stroke. (Range: 0 to 100)
StrikeThrough-Weight	double	no	The stroke weight of the strikethrough stroke.
StrikeThru	boolean	no	If true, draws a strikethrough line through the text.
StrokeAlignment	TextStrokeAlign_EnumValue	no	The stroke alignment applied to the text. Can be CenterAlignment (The stroke straddles the path), or OutsideAlignment (The stroke is outside the path, like a picture frame).
StrokeColor	string	no	The swatch (color, gradient, tint, or mixed ink) applied to the stroke of the Paragraph.
StrokeTint	double	no	The tint (as a percentage) of the stroke color of the Paragraph. (To specify a tint percentage, use a number in the range of 0 to 100; to use the inherited or overridden value, use -1.)
StrokeWeight	double	no	The stroke weight applied to the characters of the text.
Tatechuyoko	boolean	no	If true, makes the character horizontal in vertical text.
Tatechuyoko-XOffset	double	no	The horizontal offset for horizontal characters in vertical text.
Tatechuyoko-YOffset	double	no	The vertical offset for horizontal characters in vertical text.
Tracking	double	no	The amount by which to loosen or tighten a block of text, specified in thousands of an em.
TrailingAki	double	no	The amount of space after each character.
TreatIdeographic-SpaceAsSpace	boolean	no	If true, ideographic spaces will not wrap to the next line like text characters.
Tsume	double	no	The amount of horizontal character compression.
Underline	boolean	no	If true, underlines the text.
UnderlineGap-Overprint	boolean	no	If true, the gap color of the underline stroke will overprint.
UnderlineGapTint	double	no	The tint (as a percentage) of the gap color of the underline stroke. (Range: 0 to 100) Note: Valid when underline type is not solid.
UnderlineOffset	double	no	The amount by which to offset the underline from the text baseline.

Name	Type	Req	Description
Underline-Overprint	boolean	no	If true, the underline stroke color will overprint.
UnderlineTint	double	no	The underline stroke tint (as a percentage). (Range: 0 to 100)
UnderlineWeight	double	no	The stroke weight of the underline stroke.
VerticalScale	double	no	The vertical scaling applied to the text as a percentage of its current size. (Range: 1 to 1000)
Warichu	boolean	no	If true, turns on warichu.
WarichuAlignment	WarichuAlignment_EnumValue	no	The warichu alignment. Can be Auto (Automatically aligns warichu characters), Left-Align (Aligns warichu on the left side of the text frame), CenterAlign (Aligns warichu in the center of the text frame), RightAlign (Warichu on the right side of the text frame), Fully-Justified (Justifies warichu lines and makes all lines of equal length), LeftJustified (Justifies warichu lines and left aligns the last line), CenterJustified (Justifies warichu lines and center aligns the last line), or RightJustified (Justifies warichu lines and right aligns the last line).
WarichuChars-AfterBreak	short	no	The minimum number of characters allowed after a line break.
WarichuChars-BeforeBreak	short	no	The minimum number of characters allowed before a line break.
WarichuLine-Spacing	double	no	The gap between lines of warichu characters.
WarichuLines	short	no	The number of lines of warichu within a single normal line.
WarichuSize	double	no	The amount (as a percentage) to scale parent text size to determine warichu size.
XOffsetDiacritic	double	no	The X offset for diacritic adjustment
YOffsetDiacritic	double	no	The Y offset for diacritic adjustment

**Table 49. Common Text Properties Represented as Elements**

Name	Type	Req	Description
AllGREPStyles	ListItem	no	A list of the grep styles in the text.
AllLineStyles	ListItem	no	A list of the line styles in the text.
AllNestedStyles	ListItem	no	A list of the nested styles in the text.
AppliedFont	string (a reference to a self attribute) or string	no	The font applied to the text, specified as either a font object or the name of font family.

Name	Type	Req	Description
AppliedNumbering-List	string (a reference to a self attribute) or string	no	The list to be part of.
BalanceRagged-Lines	boolean or BalanceLines-Style_EnumValue	no	If true or set to an enumeration value, balances ragged lines in the paragraph(s) of the Paragraph. Note: Not valid with a single-line text composer. For information, see composer. Can be NoBalancing (Does not balance lines), VeeShape (Prefers shorter last lines), Fully-Balanced (Balances lines equally), or Pyramid-Shape (Prefers longer last lines).
BulletChar	undefined	no	Bullet character.
BulletsCharacter-Style	string (a reference to a self attribute) or string	no	The character style to be used for the text after string.
BulletsFont	string (a reference to a self attribute) or string or AutoEnum_EnumValue	no	The font used for bullet characters.
BulletsFontStyle	string or NothingEnum_EnumValue or Auto-Enum_EnumValue	no	The font style used for bullet characters.
CustomGlyph	long or string	no	A custom glyph.
KentenFillColor	string (a reference to a self attribute) or string	no	The swatch (color, gradient, tint, or mixed ink) applied to the fill of kenten characters.
KentenFont	string (a reference to a self attribute) or string	no	The font to use for kenten characters.
KentenFontStyle	string or NothingEnum_EnumValue	no	The font style of kenten characters.
KentenStrokeColor	string (a reference to a self attribute) or string	no	The swatch (color, gradient, tint, or mixed ink) applied to the stroke of kenten characters.

Name	Type	Req	Description
KinsokuSet	string (a reference to a self attribute) or KinsokuSet_EnumValue or string	no	The kinsoku set that determines legitimate line breaks. Can return: KinsokuTable, KinsokuSet enumerator or String. Can be Nothing (Does not use a kinsoku set), HardKinsoku (Uses the hard or maximum kinsoku set, which includes all Japanese characters that should not begin or end a line), SoftKinsoku (Uses the soft or weak kinsoku set, which omits from the hard kinsoku set long vowel symbols and small hiragana and katakana characters), KoreanKinsoku (Uses the Korean kinsoku set), SimplifiedChineseKinsoku (Uses the simplified Chinese kinsoku set), or TraditionalChineseKinsoku (Uses the traditional Chinese kinsoku set).
Leading	double or Leading_EnumValue	no	The leading applied to the text. Can return: Unit or Leading enumerator.
Mojikumi	string (a reference to a self attribute) or string or MojikumiTable-Defaults_EnumValue	no	The mojikumi table. For information, see mojikumi table defaults. Can be Nothing (Turns off mojikumi), LineEndAllOneHalfEmEnum (Uses half-width spacing for all characters), OneEm-IndentLineEndUkeOneHalfEmEnum (Indents lines one space and uses line end uke one half space), OneOrOneHalfEmIndentLineEnd-UkeOneHalfEmEnum (Indents lines one full or half space and uses line end uke one half space), OneOrOneHalfEmIndentLineEndAllOneEm-Enum (Uses full-width spacing for all characters except the last character in the line, which uses either full- or half-width spacing), OneEm-IndentLineEndAllOneEmEnum (Indents lines one full space and uses full-width spacing for all characters), OneEmIndentLineEndAllNo-FloatEnum (Indents lines one full space and uses no float for all characters), OneEmIndentLineEndUkeNoFloatEnum (Indents lines one full space and uses line end uke no float), One-OrOneHalfEmIndentLineEndUkeNoFloat-Enum (Indents lines one half space or one full space and uses line end uke no float), OneEm-IndentLineEndAllOneHalfEmEnum (Indents lines one full space and uses half-width spacing for all characters), LineEndAllOneEmEnum (Uses full-width spacing for all characters), LineEndUkeNoFloatEnum (Uses line end uke no float), OneOrOneHalfEmIndentLine-EndPeriodOneEmEnum (Indents lines one or one-half space and uses full-width spacing for punctuation and for the last character in the line), OneEmIndentLineEndPeriodOneEmEnum (Indents lines one full space and uses full-width spacing for punctuation and for the last character in the line), or LineEndPeriodOneEmEnum (Uses full-width spacing for punctuation).

Name	Type	Req	Description
Numbering-CharacterStyle	string (a reference to a self attribute) or string	no	The character style to be used for the number string.
NumberingFormat	NumberingStyle_EnumValue or string	no	Numbering format options. Can be Upper-Roman (Upper roman), LowerRoman (Lower roman), UpperLetters (Upper letters), Lower-Letters (Lower letters), Arabic (Arabic), KatakanaModern (Katakana (a, i, u, e, o)), KatakanaTraditional (Katakana (i, ro, ha, ni)), FormatNone (Do not add characters), SingleLeadingZeros (Add single leading zeros), Kanji (Kanji), DoubleLeadingZeros (Add double leading zeros), or Triple-LeadingZeros (Add triple leading zeros).
NumberingRestart-Policies	undefined	no	Numbering restart policies.
OpenTypeFeatures	ListItem	no	OpenType features.
RubyFill	string (a reference to a self attribute) or string	no	The swatch (color, gradient, tint, or mixed ink) applied to the fill of ruby characters.
RubyFont	string (a reference to a self attribute) or string	no	The font applied to ruby characters.
RubyFontStyle	string or NothingEnum_EnumValue	no	The font style of ruby characters.
RubyStroke	string (a reference to a self attribute) or string	no	The swatch (color, gradient, tint, or mixed ink) applied to the stroke of ruby characters.
RuleAboveColor	string (a reference to a self attribute) or string	no	The swatch (color, gradient, tint, or mixed ink) applied to the paragraph rule above.
RuleAboveGapColor	string (a reference to a self attribute) or string	no	The swatch (color, gradient, tint, or mixed ink) applied to the stroke gap of the paragraph rule above. Note: Valid only when the paragraph rule above type is not solid.
RuleAboveType	string (a reference to a self attribute) or string	no	The stroke type of the rule above the paragraph.
RuleBelowColor	string (a reference to a self attribute) or string	no	The swatch (color, gradient, tint, or mixed ink) applied to the paragraph rule below. Can return: Swatch or String.

Name	Type	Req	Description
RuleBelowGapColor	string (a reference to a self attribute) or string	no	The swatch (color, gradient, tint, or mixed ink) applied to the stroke gap of the paragraph rule below. Note: Valid only when the paragraph rule below type is not solid.
RuleBelowType	string (a reference to a self attribute) or string	no	The stroke type of the rule below the paragraph.
StrikeThrough-Color	string (a reference to a self attribute) or string	no	The swatch (color, gradient, tint, or mixed ink) applied to the strikethrough stroke.
StrikeThroughGap-Color	string (a reference to a self attribute) or string	no	The swatch (color, gradient, tint, or mixed ink) applied to the gap of the strikethrough stroke.
StrikeThroughType	string (a reference to a self attribute) or string	no	The stroke type of the strikethrough stroke. Can return: StrokeStyle or String.
TabList	ListItem	no	A list of all of the properties of all of the paragraph's tab stops. Can return: Array of Arrays of Property Name/Value Pairs.
UnderlineColor	string (a reference to a self attribute) or string	no	The swatch (color, gradient, tint, or mixed ink) applied to the underline stroke. Can return: Swatch or String.
UnderlineGapColor	string (a reference to a self attribute) or string	no	The swatch (color, gradient, tint, or mixed ink) applied to the gap of the underline stroke. Note: Valid when underline type is not solid. Can return: Swatch or String.
UnderlineType	string (a reference to a self attribute) or string	no	The stroke type of the underline stroke. Can return: StrokeStyle or String.

### Attributes and Elements Related to Japanese Features

Attributes such as RubyFont, Mojikumi, and WarichuLines are specific to the Japanese feature set. IDML recognizes these settings because InDesign supports a number of different platforms and locales. For example, an IDML package create by Japanese version of InDesign can be opened without error in English version of InDesign (just as a binary file created by one language version can be opened in another language version).

#### 10.4.7 Story Schema

The attributes and elements of a `<Story>` element define text content and text formatting of the story. InDesign stories can contain page items, such as text frames or graphics, notes, footnotes, hyperlinks, and other objects. The `<Story>` element can contain everything that can appear in a story in an InDesign document.

Most of the attributes and elements defined in the schema for a story concern the text formatting defaults and preference settings of the story. Story attributes such as `AppliedParagraphStyle`, `AppliedCharacterStyle`, and `FontStyle` define the default text formatting of the story. They can be overridden by related attributes in text child elements that appear inside the story, such as the `<ParagraphStyleRange>` element.

##### Schema Example 42. Story Schema

```
Story_Object = element Story {
    attribute Self { xsd:string },
    attribute AppliedTOCStyle { xsd:string }?,
    attribute FirstLineIndent { xsd:double }?,
    attribute LeftIndent { xsd:double }?,
    attribute RightIndent { xsd:double }?,
    attribute SpaceBefore { xsd:double }?,
    attribute SpaceAfter { xsd:double }?,
    attribute Justification { Justification_EnumValue }?,
    attribute SingleWordJustification { SingleWordJustification_EnumValue }?,
    attribute AutoLeading { xsd:double }?,
    attribute DropCapLines { xsd:short {minInclusive="0" maxInclusive="25"} }?,
    attribute DropCapCharacters { xsd:short {minInclusive="0" maxInclusive="150"} }?,
    attribute KeepLinesTogether { xsd:boolean }?,
    attribute KeepAllLinesTogether { xsd:boolean }?,
    attribute KeepWithNext { xsd:short {minInclusive="0" maxInclusive="5"} }?,
    attribute KeepFirstLines { xsd:short {minInclusive="1" maxInclusive="50"} }?,
    attribute KeepLastLines { xsd:short {minInclusive="1" maxInclusive="50"} }?,
    attribute StartParagraph { StartParagraph_EnumValue }?,
    attribute Composer { xsd:string }?,
    attribute MinimumWordSpacing { xsd:double }?,
    attribute MaximumWordSpacing { xsd:double }?,
    attribute DesiredWordSpacing { xsd:double }?,
    attribute MinimumLetterSpacing { xsd:double }?,
    attribute MaximumLetterSpacing { xsd:double }?,
    attribute DesiredLetterSpacing { xsd:double }?,
    attribute MinimumGlyphScaling { xsd:double }?,
    attribute MaximumGlyphScaling { xsd:double }?,
    attribute DesiredGlyphScaling { xsd:double }?,
    attribute RuleAbove { xsd:boolean }?,
    attribute RuleAboveOverprint { xsd:boolean }?,
    attribute RuleAboveLineWidth { xsd:double }?,
    attribute RuleAboveTint { xsd:double }?,
    attribute RuleAboveOffset { xsd:double }?,
    attribute RuleAboveLeftIndent { xsd:double }?,
    attribute RuleAboveRightIndent { xsd:double }?,
    attribute RuleAboveWidth { RuleWidth_EnumValue }?,
    attribute RuleAboveGapTint { xsd:double }?,
    attribute RuleAboveGapOverprint { xsd:boolean }?,
    attribute RuleBelow { xsd:boolean }?,
```

```
        attribute RuleBelowLineWeight { xsd:double }?,
        attribute RuleBelowTint { xsd:double }?,
        attribute RuleBelowOffset { xsd:double }?,
        attribute RuleBelowLeftIndent { xsd:double }?,
        attribute RuleBelowRightIndent { xsd:double }?,
        attribute RuleBelowWidth { RuleWidth_EnumValue }?,
        attribute RuleBelowGapTint { xsd:double }?,
        attribute HyphenateCapitalizedWords { xsd:boolean }?,
        attribute Hyphenation { xsd:boolean }?,
        attribute HyphenateBeforeLast {xsd:short {minInclusive="1" maxInclusive="15"} }?,
        attribute HyphenateAfterFirst {xsd:short {minInclusive="1" maxInclusive="15"} }?,
        attribute HyphenateWordsLongerThan {xsd:short {minInclusive="3"
maxInclusive="25"} }?,
        attribute HyphenateLadderLimit {xsd:short {minInclusive="0"
maxInclusive="25"} }?,
        attribute HyphenationZone { xsd:double }?,
        attribute HyphenWeight { xsd:short {minInclusive="0" maxInclusive="10"} }?,
        attribute AppliedParagraphStyle { xsd:string }?,
        attribute AppliedCharacterStyle { xsd:string }?,
        attribute FontStyle { xsd:string }?,
        attribute PointSize { xsd:double }?,
        attribute KerningMethod { xsd:string }?,
        attribute Tracking { xsd:double }?,
        attribute Capitalization { Capitalization_EnumValue }?,
        attribute Position { Position_EnumValue }?,
        attribute Underline { xsd:boolean }?,
        attribute StrikeThru { xsd:boolean }?,
        attribute Ligatures { xsd:boolean }?,
        attribute NoBreak { xsd:boolean }?,
        attribute HorizontalScale { xsd:double }?,
        attribute VerticalScale { xsd:double }?,
        attribute BaselineShift { xsd:double }?,
        attribute Skew { xsd:double }?,
        attribute FillTint { xsd:double }?,
        attribute StrokeTint { xsd:double }?,
        attribute StrokeWeight { xsd:double }?,
        attribute OverprintStroke { xsd:boolean }?,
        attribute OverprintFill { xsd:boolean }?,
        attribute OTFFigureStyle { OTFFigureStyle_EnumValue }?,
        attribute OTFOrdinal { xsd:boolean }?,
        attribute OTFFraction { xsd:boolean }?,
        attribute OTFDiscretionaryLigature { xsd:boolean }?,
        attribute OTFTitling { xsd:boolean }?,
        attribute OTFContextualAlternate { xsd:boolean }?,
        attribute OTFSwash { xsd:boolean }?,
        attribute UnderlineTint { xsd:double }?,
        attribute UnderlineGapTint { xsd:double }?,
        attribute UnderlineOverprint { xsd:boolean }?,
        attribute UnderlineGapOverprint { xsd:boolean }?,
        attribute UnderlineOffset { xsd:double }?,
        attribute UnderlineWeight { xsd:double }?,
        attribute StrikeThroughTint { xsd:double }?,
        attribute StrikeThroughGapTint { xsd:double }?,
        attribute StrikeThroughOverprint { xsd:boolean }?,
```

```

attribute StrikeThroughGapOverprint { xsd:boolean }?,
attribute StrikeThroughOffset { xsd:double }?,
attribute StrikeThroughWeight { xsd:double }?,
attribute FillColor { xsd:string }?,
attribute StrokeColor { xsd:string }?,
attribute AppliedLanguage { xsd:string }?,
attribute LastLineIndent { xsd:double }?,
attribute HyphenateLastWord { xsd:boolean }?,
attribute OTFSashedZero { xsd:boolean }?,
attribute OTFHistorical { xsd:boolean }?,
attribute OTFStylisticSets { xsd:int }?,
attribute GradientFillLength { xsd:double }?,
attribute GradientFillAngle { xsd:double }?,
attribute GradientStrokeLength { xsd:double }?,
attribute GradientStrokeAngle { xsd:double }?,
attribute GradientFillStart { UnitPointType_TypeDef }?,
attribute GradientStrokeStart { UnitPointType_TypeDef }?,
attribute RuleBelowOverprint { xsd:boolean }?,
attribute RuleBelowGapOverprint { xsd:boolean }?,
attribute DropcapDetail { xsd:int }?,
attribute HyphenateAcrossColumns { xsd:boolean }?,
attribute KeepRuleAboveInFrame { xsd:boolean }?,
attribute IgnoreEdgeAlignment { xsd:boolean }?,
attribute OTFMark { xsd:boolean }?,
attribute OTFLocale { xsd:boolean }?,
attribute PositionalForm { PositionalForms_EnumValue }?,
attribute ParagraphDirection { ParagraphDirection_EnumValue }?,
attribute ParagraphJustification { ParagraphJustification_EnumValue }?,
attribute MiterLimit { xsd:double {minInclusive="0" maxInclusive="1000"} }?,
attribute StrokeAlignment { TextStrokeAlign_EnumValue }?,
attribute EndJoin { OutlineJoin_EnumValue }?,
attribute OTFOverlapSwash { xsd:boolean }?,
attribute OTFStylisticAlternate { xsd:boolean }?,
attribute OTFJustificationAlternate { xsd:boolean }?,
attribute OTFStretchedAlternate { xsd:boolean }?,
attribute CharacterDirection { CharacterDirection_EnumValue }?,
attribute KeyboardDirection { CharacterDirection_EnumValue }?,
attribute DigitsType { DigitsType_EnumValue }?,
attribute Kashidas { Kashidas_EnumValue }?,
attribute DiacriticPosition { DiacriticPosition_EnumValue }?,
attribute XOffsetDiacritic { xsd:double }?,
attribute YOffsetDiacritic { xsd:double }?,
attribute GotoNextX { GotoNextX_EnumValue }?,
attribute PageNumberType { PageNumberType_EnumValue }?,
attribute TrackChanges { xsd:boolean }?,
attribute StoryTitle { xsd:string }?,
attribute AppliedNamedGrid { xsd:string }?,
attribute CharacterAlignment { CharacterAlignment_EnumValue }?,
attribute Tsume { xsd:double }?,
attribute LeadingAki { xsd:double }?,
attribute TrailingAki { xsd:double }?,
attribute CharacterRotation { xsd:double }?,
attribute Jidori { xsd:short }?,
attribute ShataiMagnification { xsd:double }?,

```

```

        attribute ShataiDegreeAngle { xsd:double }?,
        attribute ShataiAdjustRotation { xsd:boolean }?,
        attribute ShataiAdjustTsume { xsd:boolean }?,
        attribute Tatechuyoko { xsd:boolean }?,
        attribute TatechuyokoXOffset { xsd:double }?,
        attribute TatechuyokoYOffset { xsd:double }?,
        attribute KentenTint { xsd:double }?,
        attribute KentenStrokeTint { xsd:double }?,
        attribute KentenWeight { xsd:double }?,
        attribute KentenOverprintFill { AdornmentOverprint_EnumValue }?,
        attribute KentenOverprintStroke { AdornmentOverprint_EnumValue }?,
        attribute KentenKind { KentenCharacter_EnumValue }?,
        attribute KentenPlacement { xsd:double }?,
        attribute KentenAlignment { KentenAlignment_EnumValue }?,
        attribute KentenPosition { RubyKentenPosition_EnumValue }?,
        attribute KentenFontSize { xsd:double }?,
        attribute KentenXScale { xsd:double }?,
        attribute KentenYScale { xsd:double }?,
        attribute KentenCustomCharacter { xsd:string }?,
        attribute KentenCharacterSet { KentenCharacterSet_EnumValue }?,
        attribute RubyTint { xsd:double }?,
        attribute RubyWeight { xsd:double }?,
        attribute RubyOverprintFill { AdornmentOverprint_EnumValue }?,
        attribute RubyOverprintStroke { AdornmentOverprint_EnumValue }?,
        attribute RubyStrokeTint { xsd:double }?,
        attribute RubyFontSize { xsd:double }?,
        attribute RubyOpenTypePro { xsd:boolean }?,
        attribute RubyXScale { xsd:double }?,
        attribute RubyYScale { xsd:double }?,
        attribute RubyType { RubyTypes_EnumValue }?,
        attribute RubyAlignment { RubyAlignments_EnumValue }?,
        attribute RubyPosition { RubyKentenPosition_EnumValue }?,
        attribute RubyXOffset { xsd:double }?,
        attribute RubyYOffset { xsd:double }?,
        attribute RubyParentSpacing { RubyParentSpacing_EnumValue }?,
        attribute RubyAutoAlign { xsd:boolean }?,
        attribute RubyOverhang { xsd:boolean }?,
        attribute RubyAutoScaling { xsd:boolean }?,
        attribute RubyParentScalingPercent { xsd:double }?,
        attribute RubyParentOverhangAmount { RubyOverhang_EnumValue }?,
        attribute Warichu { xsd:boolean }?,
        attribute WarichuSize { xsd:double }?,
        attribute WarichuLines { xsd:short }?,
        attribute WarichuLineSpacing { xsd:double }?,
        attribute WarichuAlignment { WarichuAlignment_EnumValue }?,
        attribute WarichuCharsAfterBreak { xsd:short }?,
        attribute WarichuCharsBeforeBreak { xsd:short }?,
        attribute OTFProportionalMetrics { xsd:boolean }?,
        attribute OTFHVVKana { xsd:boolean }?,
        attribute OTFRomanItalics { xsd:boolean }?,
        attribute ScaleAffectsLineHeight { xsd:boolean }?,
        attribute CjkGridTracking { xsd:boolean }?,
        attribute GlyphForm { AlternateGlyphForms_EnumValue }?,
        attribute RubyFlag { xsd:int }?,

```

```

attribute RubyString { xsd:string }?,
attribute GridAlignFirstLineOnly { xsd:boolean }?,
attribute GridAlignment { GridAlignment_EnumValue }?,
attribute GridGyoudori { xsd:short }?,
attribute AutoTcy { xsd:short }?,
attribute AutoTcyIncludeRoman { xsd:boolean }?,
attribute KinsokuType { KinsokuType_EnumValue }?,
attribute KinsokuHangType { KinsokuHangTypes_EnumValue }?,
attribute BunriKinshi { xsd:boolean }?,
attribute Rensuuji { xsd:boolean }?,
attribute RotateSingleByteCharacters { xsd:boolean }?,
attribute LeadingModel { LeadingModel_EnumValue }?,
attribute RubyAutoTcyDigits { xsd:short }?,
attribute RubyAutoTcyIncludeRoman { xsd:boolean }?,
attribute RubyAutoTcyAutoScale { xsd:boolean }?,
attribute TreatIdeographicSpaceAsSpace { xsd:boolean }?,
attribute AllowArbitraryHyphenation { xsd:boolean }?,
attribute ParagraphGyoudori { xsd:boolean }?,
attribute BulletsAndNumberingListType { ListType_EnumValue }?,
attribute NumberingExpression { xsd:string }?,
attribute BulletsTextAfter { xsd:string }?,
attribute NumberingLevel { xsd:int }?,
attribute NumberingContinue { xsd:boolean }?,
attribute NumberingStartAt { xsd:int }?,
attribute NumberingApplyRestartPolicy { xsd:boolean }?,
attribute BulletsAlignment { ListAlignment_EnumValue }?,
attribute NumberingAlignment { ListAlignment_EnumValue }?,
element Properties {
    element ExcelImportPreferences { list_type, element ListItem {
        (enum_type, AlignmentStyleOptions_EnumValue ) |
        (long_type, xsd:int ) |
        (bool_type, xsd:boolean ) |
        (enum_type, TableFormattingOptions_EnumValue ) |
        (string_type, xsd:string )
    }*
} ?&
    element WordRTFImportPreferences { list_type, element ListItem {
        (bool_type, xsd:boolean ) |
        (enum_type, ConvertPageBreaks_EnumValue ) |
        (enum_type, ConvertTablesOptions_EnumValue ) |
        (enum_type, ResolveStyleClash_EnumValue ) |
        (long_type, xsd:int )
    }*
} ?&
    element TextImportPreferences { list_type, element ListItem {
        (bool_type, xsd:boolean ) |
        (long_type, xsd:int ) |
        (enum_type, TextImportCharacterSet_EnumValue ) |
        (enum_type, ImportPlatform_EnumValue ) |
        (short_type, xsd:short )
    }*
} ?&
    element StyleMappingPreferences { list_type,
        element ListItem {

```

```

        list_type, element ListItem { string_type, xsd:string }*
    },
    element ListItem {
        list_type, element ListItem { string_type, xsd:string }*
    }
} ?&
element BalanceRaggedLines {
    (bool_type, xsd:boolean ) |
    (enum_type, BalanceLinesStyle_EnumValue )
} ?&
element RuleAboveColor {
    (object_type, xsd:string ) |
    (string_type, xsd:string )
} ?&
element RuleAboveGapColor {
    (object_type, xsd:string ) |
    (string_type, xsd:string )
} ?&
element RuleAboveType {
    (object_type, xsd:string ) |
    (string_type, xsd:string )
} ?&
element RuleBelowColor {
    (object_type, xsd:string ) |
    (string_type, xsd:string )
} ?&
element RuleBelowGapColor {
    (object_type, xsd:string ) |
    (string_type, xsd:string )
} ?&
element RuleBelowType {
    (object_type, xsd:string ) |
    (string_type, xsd:string )
} ?&
element AllNestedStyles { list_type, element ListItem {
    record_type,
    (
        element AppliedCharacterStyle { object_type, xsd:string } &
        element Delimiter {
            (string_type, xsd:string ) |
            (enum_type, NestedStyleDelimiters_EnumValue )
        } &
        element Repetition { long_type, xsd:int } &
        element Inclusive { bool_type, xsd:boolean })
    } *
} ?&
element TabList { list_type, element ListItem {
    record_type,
    (
        element Alignment { enum_type, TabStopAlignment_EnumValue } &
        element AlignmentCharacter { string_type, xsd:string } &
        element Leader { string_type, xsd:string } &
        element Position { unit_type, xsd:double })
    } *
}

```

```

} ?&
element AppliedFont {
  (object_type, xsd:string ) |
  (string_type, xsd:string )
} ?&
element Leading {
  (unit_type, xsd:double ) |
  (enum_type, Leading_EnumValue )
} ?&
element UnderlineColor {
  (object_type, xsd:string ) |
  (string_type, xsd:string )
} ?&
element UnderlineGapColor {
  (object_type, xsd:string ) |
  (string_type, xsd:string )
} ?&
element UnderlineType {
  (object_type, xsd:string ) |
  (string_type, xsd:string )
} ?&
element StrikeThroughColor {
  (object_type, xsd:string ) |
  (string_type, xsd:string )
} ?&
element StrikeThroughGapColor {
  (object_type, xsd:string ) |
  (string_type, xsd:string )
} ?&
element StrikeThroughType {
  (object_type, xsd:string ) |
  (string_type, xsd:string )
} ?&
element AllLineStyles { list_type, element ListItem {
  record_type,
  (
    element AppliedCharacterStyle { object_type, xsd:string } &
    element LineCount { long_type, xsd:int } &
    element RepeatLast { long_type, xsd:int })
  } *
} ?&
element AllGREPStyles { list_type, element ListItem {
  record_type,
  (
    element AppliedCharacterStyle { object_type, xsd:string } &
    element GrepExpression { string_type, xsd:string })
  } *
} ?&
element OpenTypeFeatures { list_type, element ListItem {
  list_type,
  element ListItem {
    (string_type, xsd:string ) |
    (long_type, xsd:int )
  },
}

```

```

element ListItem {
    (string_type, xsd:string) |
    (long_type, xsd:int)
}
}*
}?
element KentenFillColor {
    (object_type, xsd:string) |
    (string_type, xsd:string)
}?
element KentenStrokeColor {
    (object_type, xsd:string) |
    (string_type, xsd:string)
}?
element KentenFont {
    (object_type, xsd:string) |
    (string_type, xsd:string)
}?
element KentenFontStyle {
    (string_type, xsd:string) |
    (enum_type, NothingEnum_EnumValue)
}?
element RubyFill {
    (object_type, xsd:string) |
    (string_type, xsd:string)
}?
element RubyStroke {
    (object_type, xsd:string) |
    (string_type, xsd:string)
}?
element RubyFont {
    (object_type, xsd:string) |
    (string_type, xsd:string)
}?
element RubyFontStyle {
    (string_type, xsd:string) |
    (enum_type, NothingEnum_EnumValue)
}?
element KinsokuSet {
    (object_type, xsd:string) |
    (enum_type, KinsokuSet_EnumValue) |
    (string_type, xsd:string)
}?
element Mojikumi {
    (object_type, xsd:string) |
    (string_type, xsd:string) |
    (enum_type, MojikumiTableDefaults_EnumValue)
}?
element BulletChar {
    attribute BulletCharacterType { BulletCharacterType_EnumValue },
    attribute BulletCharacterValue { xsd:int }
}?
element BulletsFont {
    (object_type, xsd:string) |

```

```

        (string_type, xsd:string ) |
        (enum_type, AutoEnum_EnumValue )
    }?&
    element BulletsFontStyle {
        (string_type, xsd:string ) |
        (enum_type, NothingEnum_EnumValue ) |
        (enum_type, AutoEnum_EnumValue )
    }?&
    element BulletsCharacterStyle {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element NumberingCharacterStyle {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element AppliedNumberingList {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element NumberingFormat {
        (enum_type, NumberingStyle_EnumValue ) |
        (string_type, xsd:string )
    }?&
    element NumberingRestartPolicies {
        attribute RestartPolicy { RestartPolicy_EnumValue },
        attribute LowerLevel { xsd:int },
        attribute UpperLevel { xsd:int }
    }?&
    element Label { element KeyValuePair { KeyValuePair_TypeDef }* }
    }?
}
?
,
(
    GridDataInformation_Object?,
    (StoryPreference_Object?&
    MetadataPacketPreference_Object?&
    InCopyExportOption_Object?),
    (GaijiOwnedItemObject_Object*&
    Footnote_Object*&
    TextVariableInstance_Object*&
    ParagraphStyleRange_Object*&
    CharacterStyleRange_Object*&
    XMLElement_Object*&
    Table_Object*&
    Link_Object*&
    Change_Object*&
    Note_Object*&
    TextFrame_Object*&
    Oval_Object*&
    Rectangle_Object*&
    GraphicLine_Object*&
    Polygon_Object&
)

```

```

Group_Object*&
EPSText_Object*&
FormField_Object*&
Button_Object*&
HiddenText_Object*&
element Content {text}*&
element Br {empty}*)
)
}

```

Most of the properties of a story that are expressed as attributes can be found in the listing of common text element attributes (see “[Common Text Properties](#)”).

---

**Table 50. Story Properties Represented as Attributes**

Name	Type	Req	Description
AppliedNamedGrid	string	no	The named grid in use.
AppliedTOCStyle	string	no	The applied TOC style.
StoryTitle	string	no	Title for this story
TrackChanges	boolean	no	If true, track changes is turned on.

Most of the properties of a story that are expressed as elements can be found in the listing of common text properties elements (see “[Common Text Properties](#)”). Stories can also contain a number of elements that are unique to the `<Story>` element. These elements are described in the following table.

---

**Table 51. Story Properties Represented as Elements**

Name	Type	Req	Description
ExcelImport-Preferences	ListItem	no	A series of <code>ListItem</code> elements defining the properties of the Excel import preferences. InDesign exports these properties to maintain round-trip accuracy of the data in the IDML document; there is no need to include them in IDML documents you create yourself.
StyleMapping-Preferences	ListItem	no	A series of <code>ListItem</code> elements defining the properties of the style mapping import preferences. InDesign exports these properties to maintain round-trip accuracy of the data in the IDML document; there is no need to include them in IDML documents you create yourself.
TextImport-Preferences	ListItem	no	A series of <code>ListItem</code> elements defining the properties of the text import preferences. InDesign exports these properties to maintain round-trip accuracy of the data in the IDML document; there is no need to include them in IDML documents you create yourself.

Name	Type	Req	Description
WordRTFImport-Preferences	ListItem	no	A series of <code>ListItem</code> elements defining the properties of the Word/RTF import preferences. InDesign exports these properties to maintain round-trip accuracy of the data in the IDML document; there is no need to include them in IDML documents you create yourself.

#### 10.4.8 Text Child Elements

Text child elements represent unique objects within the story, such as ranges of text, inline or anchored objects (groups or frames), notes, tables, and hyperlinks. These fall into two categories based on their representation in the IDML format: inline elements or text range elements.

Some inline text child elements may contain text content themselves—a note, for example, will contain further `<ParagraphStyleRange>` and `<CharacterStyleRange>` elements, which, in turn, can also contain anchored or inline items. An anchored text frame will contain a reference to another `<Story>` element, which, in turn, can contain `<ParagraphStyleRange>` and `<CharacterStyleRange>` elements and text child elements of its own.

##### ***Text Range Elements***

Text range elements are the XML elements that define a range of text in a story. In general, a text range element contains a continuous “run” of text formatting. These objects are further broken up into `<ParagraphStyleRange>` elements, which contain ranges of continuous paragraph formatting. `<ParagraphStyleRange>` elements contain `<CharacterStyleRange>` elements, which define a continuous range of character formatting. All of the text in a `<Story>` element is contained inside `<Content>` elements within `<CharacterStyleRange>` elements.

##### **Schema Example 43. ParagraphStyleRange**

```
ParagraphStyleRange_Object = element ParagraphStyleRange {
    attribute FirstLineIndent { xsd:double }?,
    attribute LeftIndent { xsd:double }?,
    attribute RightIndent { xsd:double }?,
    attribute SpaceBefore { xsd:double }?,
    attribute SpaceAfter { xsd:double }?,
    attribute Justification { Justification_EnumValue }?,
    attribute SingleWordJustification { SingleWordJustification_EnumValue }?,
    attribute AutoLeading { xsd:double }?,
    attribute DropCapLines { xsd:short {minInclusive="0" maxInclusive="25"} }?,
    attribute DropCapCharacters { xsd:short {minInclusive="0" maxInclusive="150"} }?,
    attribute KeepLinesTogether { xsd:boolean }?,
    attribute KeepAllLinesTogether { xsd:boolean }?,
    attribute KeepWithNext { xsd:short {minInclusive="0" maxInclusive="5"} }?,
    attribute KeepFirstLines { xsd:short {minInclusive="1" maxInclusive="50"} }?,
    attribute KeepLastLines { xsd:short {minInclusive="1" maxInclusive="50"} }?,
    attribute StartParagraph { StartParagraph_EnumValue }?,
    attribute Composer { xsd:string }?,
    attribute MinimumWordSpacing { xsd:double }?,
    attribute MaximumWordSpacing { xsd:double }?,
}
```

```

        attribute DesiredWordSpacing { xsd:double }?,
        attribute MinimumLetterSpacing { xsd:double }?,
        attribute MaximumLetterSpacing { xsd:double }?,
        attribute DesiredLetterSpacing { xsd:double }?,
        attribute MinimumGlyphScaling { xsd:double }?,
        attribute MaximumGlyphScaling { xsd:double }?,
        attribute DesiredGlyphScaling { xsd:double }?,
        attribute RuleAbove { xsd:boolean }?,
        attribute RuleAboveOverprint { xsd:boolean }?,
        attribute RuleAboveLineWeight { xsd:double }?,
        attribute RuleAboveTint { xsd:double }?,
        attribute RuleAboveOffset { xsd:double }?,
        attribute RuleAboveLeftIndent { xsd:double }?,
        attribute RuleAboveRightIndent { xsd:double }?,
        attribute RuleAboveWidth { RuleWidth_EnumValue }?,
        attribute RuleAboveGapTint { xsd:double }?,
        attribute RuleAboveGapOverprint { xsd:boolean }?,
        attribute RuleBelow { xsd:boolean }?,
        attribute RuleBelowLineWeight { xsd:double }?,
        attribute RuleBelowTint { xsd:double }?,
        attribute RuleBelowOffset { xsd:double }?,
        attribute RuleBelowLeftIndent { xsd:double }?,
        attribute RuleBelowRightIndent { xsd:double }?,
        attribute RuleBelowWidth { RuleWidth_EnumValue }?,
        attribute RuleBelowGapTint { xsd:double }?,
        attribute HyphenateCapitalizedWords { xsd:boolean }?,
        attribute Hyphenation { xsd:boolean }?,
        attribute HyphenateBeforeLast { xsd:short {minInclusive="1" maxInclusive="15"} }?,
        attribute HyphenateAfterFirst { xsd:short {minInclusive="1" maxInclusive="15"} }?,
        attribute HyphenateWordsLongerThan { xsd:short {minInclusive="3" maxInclusive="25"} }?,
        attribute HyphenateLadderLimit { xsd:short {minInclusive="0" maxInclusive="25"} }?,
        attribute HyphenationZone { xsd:double }?,
        attribute HyphenWeight { xsd:short {minInclusive="0" maxInclusive="10"} }?,
        attribute AppliedParagraphStyle { xsd:string }?,
        attribute AppliedCharacterStyle { xsd:string }?,
        attribute KerningValue { xsd:double }?,
        attribute FontStyle { xsd:string }?,
        attribute PointSize { xsd:double }?,
        attribute KerningMethod { xsd:string }?,
        attribute Tracking { xsd:double }?,
        attribute Capitalization { Capitalization_EnumValue }?,
        attribute Position { Position_EnumValue }?,
        attribute Underline { xsd:boolean }?,
        attribute StrikeThru { xsd:boolean }?,
        attribute Ligatures { xsd:boolean }?,
        attribute NoBreak { xsd:boolean }?,
        attribute HorizontalScale { xsd:double }?,
        attribute VerticalScale { xsd:double }?,
        attribute BaselineShift { xsd:double }?,
        attribute Skew { xsd:double }?
    
```

```

attribute FillTint { xsd:double }?,
attribute StrokeTint { xsd:double }?,
attribute StrokeWeight { xsd:double }?,
attribute OverprintStroke { xsd:boolean }?,
attribute OverprintFill { xsd:boolean }?,
attribute OTFFigureStyle { OTFFigureStyle_EnumValue }?,
attribute OTFOrdinal { xsd:boolean }?,
attribute OTFFraction { xsd:boolean }?,
attribute OTFDiscretionaryLigature { xsd:boolean }?,
attribute OTFTitling { xsd:boolean }?,
attribute OTFContextualAlternate { xsd:boolean }?,
attribute OTFSwash { xsd:boolean }?,
attribute UnderlineTint { xsd:double }?,
attribute UnderlineGapTint { xsd:double }?,
attribute UnderlineOverprint { xsd:boolean }?,
attribute UnderlineGapOverprint { xsd:boolean }?,
attribute UnderlineOffset { xsd:double }?,
attribute UnderlineWeight { xsd:double }?,
attribute StrikeThroughTint { xsd:double }?,
attribute StrikeThroughGapTint { xsd:double }?,
attribute StrikeThroughOverprint { xsd:boolean }?,
attribute StrikeThroughGapOverprint { xsd:boolean }?,
attribute StrikeThroughOffset { xsd:double }?,
attribute StrikeThroughWeight { xsd:double }?,
attribute FillColor { xsd:string }?,
attribute StrokeColor { xsd:string }?,
attribute AppliedLanguage { xsd:string }?,
attribute LastLineIndent { xsd:double }?,
attribute HyphenateLastWord { xsd:boolean }?,
attribute OTFSashedZero { xsd:boolean }?,
attribute OTFHistorical { xsd:boolean }?,
attribute OTFStylisticSets { xsd:int }?,
attribute GradientFillLength { xsd:double }?,
attribute GradientFillAngle { xsd:double }?,
attribute GradientStrokeLength { xsd:double }?,
attribute GradientStrokeAngle { xsd:double }?,
attribute GradientFillStart { UnitPointType_TypeDef }?,
attribute GradientStrokeStart { UnitPointType_TypeDef }?,
attribute RuleBelowOverprint { xsd:boolean }?,
attribute RuleBelowGapOverprint { xsd:boolean }?,
attribute DropcapDetail { xsd:int }?,
attribute HyphenateAcrossColumns { xsd:boolean }?,
attribute KeepRuleAboveInFrame { xsd:boolean }?,
attribute IgnoreEdgeAlignment { xsd:boolean }?,
attribute OTFMark { xsd:boolean }?,
attribute OTFLocale { xsd:boolean }?,
attribute PositionalForm { PositionalForms_EnumValue }?,
attribute ParagraphDirection { ParagraphDirection_EnumValue }?,
attribute ParagraphJustification { ParagraphJustification_EnumValue }?,
attribute MiterLimit { xsd:double {minInclusive="0" maxInclusive="1000"} }?,
attribute StrokeAlignment { TextStrokeAlign_EnumValue }?,
attribute EndJoin { OutlineJoin_EnumValue }?,
attribute OTFOverlapSwash { xsd:boolean }?,
attribute OTFStylisticAlternate { xsd:boolean }?,

```

```

        attribute OTFJustificationAlternate { xsd:boolean }?,
        attribute OTFSretchedAlternate { xsd:boolean }?,
        attribute CharacterDirection { CharacterDirection_EnumValue }?,
        attribute KeyboardDirection { CharacterDirection_EnumValue }?,
        attribute DigitsType { DigitsType_EnumValue }?,
        attribute Kashidas { Kashidas_EnumValue }?,
        attribute DiacriticPosition { DiacriticPosition_EnumValue }?,
        attribute XOffsetDiacritic { xsd:double }?,
        attribute YOffsetDiacritic { xsd:double }?,
        attribute GotoNextX { GotoNextX_EnumValue }?,
        attribute PageNumberType { PageNumberType_EnumValue }?,
        attribute AppliedConditions { list { xsd:string * } }?,
        attribute CharacterAlignment { CharacterAlignment_EnumValue }?,
        attribute Tsume { xsd:double }?,
        attribute LeadingAki { xsd:double }?,
        attribute TrailingAki { xsd:double }?,
        attribute CharacterRotation { xsd:double }?,
        attribute Jidori { xsd:short }?,
        attribute ShataiMagnification { xsd:double }?,
        attribute ShataiDegreeAngle { xsd:double }?,
        attribute ShataiAdjustRotation { xsd:boolean }?,
        attribute ShataiAdjustTsume { xsd:boolean }?,
        attribute Tatechuyoko { xsd:boolean }?,
        attribute TatechuyokoXOffset { xsd:double }?,
        attribute TatechuyokoYOffset { xsd:double }?,
        attribute KentenTint { xsd:double }?,
        attribute KentenStrokeTint { xsd:double }?,
        attribute KentenWeight { xsd:double }?,
        attribute KentenOverprintFill { AdornmentOverprint_EnumValue }?,
        attribute KentenOverprintStroke { AdornmentOverprint_EnumValue }?,
        attribute KentenKind { KentenCharacter_EnumValue }?,
        attribute KentenPlacement { xsd:double }?,
        attribute KentenAlignment { KentenAlignment_EnumValue }?,
        attribute KentenPosition { RubyKentenPosition_EnumValue }?,
        attribute KentenFontSize { xsd:double }?,
        attribute KentenXScale { xsd:double }?,
        attribute KentenYScale { xsd:double }?,
        attribute KentenCustomCharacter { xsd:string }?,
        attribute KentenCharacterSet { KentenCharacterSet_EnumValue }?,
        attribute RubyTint { xsd:double }?,
        attribute RubyWeight { xsd:double }?,
        attribute RubyOverprintFill { AdornmentOverprint_EnumValue }?,
        attribute RubyOverprintStroke { AdornmentOverprint_EnumValue }?,
        attribute RubyStrokeTint { xsd:double }?,
        attribute RubyFontSize { xsd:double }?,
        attribute RubyOpenTypePro { xsd:boolean }?,
        attribute RubyXScale { xsd:double }?,
        attribute RubyYScale { xsd:double }?,
        attribute RubyType { RubyTypes_EnumValue }?,
        attribute RubyAlignment { RubyAlignments_EnumValue }?,
        attribute RubyPosition { RubyKentenPosition_EnumValue }?,
        attribute RubyXOffset { xsd:double }?,
        attribute RubyYOffset { xsd:double }?,
        attribute RubyParentSpacing { RubyParentSpacing_EnumValue }?,

```

```

attribute RubyAutoAlign { xsd:boolean }?,
attribute RubyOverhang { xsd:boolean }?,
attribute RubyAutoScaling { xsd:boolean }?,
attribute RubyParentScalingPercent { xsd:double }?,
attribute RubyParentOverhangAmount { RubyOverhang_EnumValue }?,
attribute Warichu { xsd:boolean }?,
attribute WarichuSize { xsd:double }?,
attribute WarichuLines { xsd:short }?,
attribute WarichuLineSpacing { xsd:double }?,
attribute WarichuAlignment { WarichuAlignment_EnumValue }?,
attribute WarichuCharsAfterBreak { xsd:short }?,
attribute WarichuCharsBeforeBreak { xsd:short }?,
attribute OTFProportionalMetrics { xsd:boolean }?,
attribute OTFHVKana { xsd:boolean }?,
attribute OTFRomanItalics { xsd:boolean }?,
attribute ScaleAffectsLineHeight { xsd:boolean }?,
attribute CjkGridTracking { xsd:boolean }?,
attribute GlyphForm { AlternateGlyphForms_EnumValue }?,
attribute RubyFlag { xsd:int }?,
attribute RubyString { xsd:string }?,
attribute GridAlignFirstLineOnly { xsd:boolean }?,
attribute GridAlignment { GridAlignment_EnumValue }?,
attribute GridGyoudori { xsd:short }?,
attribute AutoTcy { xsd:short }?,
attribute AutoTcyIncludeRoman { xsd:boolean }?,
attribute KinsokuType { KinsokuType_EnumValue }?,
attribute KinsokuHangType { KinsokuHangTypes_EnumValue }?,
attribute BunriKinshi { xsd:boolean }?,
attribute Rensuuji { xsd:boolean }?,
attribute RotateSingleByteCharacters { xsd:boolean }?,
attribute LeadingModel { LeadingModel_EnumValue }?,
attribute RubyAutoTcyDigits { xsd:short }?,
attribute RubyAutoTcyIncludeRoman { xsd:boolean }?,
attribute RubyAutoTcyAutoScale { xsd:boolean }?,
attribute TreatIdeographicSpaceAsSpace { xsd:boolean }?,
attribute AllowArbitraryHyphenation { xsd:boolean }?,
attribute ParagraphGyoudori { xsd:boolean }?,
attribute BulletsAndNumberingListType { ListType_EnumValue }?,
attribute NumberingExpression { xsd:string }?,
attribute BulletsTextAfter { xsd:string }?,
attribute NumberingLevel { xsd:int }?,
attribute NumberingContinue { xsd:boolean }?,
attribute NumberingStartAt { xsd:int }?,
attribute NumberingApplyRestartPolicy { xsd:boolean }?,
attribute BulletsAlignment { ListAlignment_EnumValue }?,
attribute NumberingAlignment { ListAlignment_EnumValue }?,
element Properties {
    element BalanceRaggedLines {
        (bool_type, xsd:boolean ) |
        (enum_type, BalanceLinesStyle_EnumValue )
    }?&
    element RuleAboveColor {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }
}

```

```

} ?&
element RuleAboveGapColor {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element RuleAboveType {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element RuleBelowColor {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element RuleBelowGapColor {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element RuleBelowType {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element AllNestedStyles { list_type, element ListItem {
  record_type,
  (
    element AppliedCharacterStyle { object_type, xsd:string } &
    element Delimiter {
      (string_type, xsd:string) |
      (enum_type, NestedStyleDelimiters_EnumValue)
    } &
    element Repetition { long_type, xsd:int } &
    element Inclusive { bool_type, xsd:boolean })
  } *
} ?&
element TabList { list_type, element ListItem {
  record_type,
  (
    element Alignment { enum_type, TabStopAlignment_EnumValue } &
    element AlignmentCharacter { string_type, xsd:string } &
    element Leader { string_type, xsd:string } &
    element Position { unit_type, xsd:double })
  } *
} ?&
element AppliedFont {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element Leading {
  (unit_type, xsd:double) |
  (enum_type, Leading_EnumValue)
} ?&
element UnderlineColor {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&

```

```

element UnderlineGapColor {
    (object_type, xsd:string) |
    (string_type, xsd:string)
} ?&
element UnderlineType {
    (object_type, xsd:string) |
    (string_type, xsd:string)
} ?&
element StrikeThroughColor {
    (object_type, xsd:string) |
    (string_type, xsd:string)
} ?&
element StrikeThroughGapColor {
    (object_type, xsd:string) |
    (string_type, xsd:string)
} ?&
element StrikeThroughType {
    (object_type, xsd:string) |
    (string_type, xsd:string)
} ?&
element CustomGlyph {
    (long_type, xsd:int) |
    (string_type, xsd:string)
} ?&
element AllLineStyles { list_type, element ListItem {
    record_type,
    (
        element AppliedCharacterStyle { object_type, xsd:string } &
        element LineCount { long_type, xsd:int } &
        element RepeatLast { long_type, xsd:int })
    } *
} ?&
element AllGREPstyles { list_type, element ListItem {
    record_type,
    (
        element AppliedCharacterStyle { object_type, xsd:string } &
        element GrepExpression { string_type, xsd:string })
    } *
} ?&
element OpenTypeFeatures { list_type, element ListItem {
    list_type,
    element ListItem {
        (string_type, xsd:string) |
        (long_type, xsd:int)
    },
    element ListItem {
        (string_type, xsd:string) |
        (long_type, xsd:int)
    }
} *
} ?&
element KentenFillColor {
    (object_type, xsd:string) |
    (string_type, xsd:string)
}

```

```

} ?&
element KentenStrokeColor {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element KentenFont {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element KentenFontStyle {
  (string_type, xsd:string) |
  (enum_type, NothingEnum_EnumValue)
} ?&
element RubyFill {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element RubyStroke {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element RubyFont {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element RubyFontStyle {
  (string_type, xsd:string) |
  (enum_type, NothingEnum_EnumValue)
} ?&
element KinsokuSet {
  (object_type, xsd:string) |
  (enum_type, KinsokuSet_EnumValue) |
  (string_type, xsd:string)
} ?&
element Mojikumi {
  (object_type, xsd:string) |
  (string_type, xsd:string) |
  (enum_type, MojikumiTableDefaults_EnumValue)
} ?&
element BulletChar {
  attribute BulletCharacterType { BulletCharacterType_EnumValue },
  attribute BulletCharacterValue { xsd:int }
} ?&
element BulletsFont {
  (object_type, xsd:string) |
  (string_type, xsd:string) |
  (enum_type, AutoEnum_EnumValue)
} ?&
element BulletsFontStyle {
  (string_type, xsd:string) |
  (enum_type, NothingEnum_EnumValue) |
  (enum_type, AutoEnum_EnumValue)
} ?&
element BulletsCharacterStyle {

```

```

(object_type, xsd:string) |
(string_type, xsd:string)
} ?&
element NumberingCharacterStyle {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element AppliedNumberingList {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element NumberingFormat {
  (enum_type, NumberingStyle_EnumValue) |
  (string_type, xsd:string)
} ?&
element NumberingRestartPolicies {
  attribute RestartPolicy { RestartPolicy_EnumValue },
  attribute LowerLevel { xsd:int },
  attribute UpperLevel { xsd:int }
} ?
}
?
,
(
  CharacterStyleRange_Object*&
  Note_Object*&
  Table_Object*&
  TextVariableInstance_Object*&
  Footnote_Object*&
  TextFrame_Object*&
  Oval_Object*&
  Rectangle_Object*&
  GraphicLine_Object*&
  Polygon_Object*&
  Group_Object*&
  EPSText_Object*&
  FormField_Object*&
  Button_Object*&
  HyperlinkTextDestination_Object*&
  ParagraphDestination_Object*&
  Change_Object*&
  XMLElement_Object*&
  XMLComment_Object*&
  XMLInstruction_Object1*&
  DTD_Object*&
  HiddenText_Object*&
  HyperlinkTextSource_Object*&
  CrossReferenceSource_Object*&
  element Content {text}*&
  element Br {empty}*
)
}

```

### **ParagraphStyleRange Attributes**

The properties of a <ParagraphStyleRange> that are expressed as attributes can be found in the listing of common text element attributes (see “[Common Text Properties](#)”).

### **ParagraphStyleRange Elements**

The properties of a <ParagraphStyleRange> that are expressed as elements can be found in the listing of common text elements (see “[Common Text Properties](#)”).

### **Character Style Range**

Inside a <ParagraphStyleRange> element, you’ll find one or more <CharacterStyleRange> elements. <CharacterStyleRange> elements represent a continuous run of text formatting.

#### **Schema Example 44. CharacterStyleRange**

```
CharacterStyleRange_Object = element CharacterStyleRange {
    attribute FirstLineIndent { xsd:double }?,
    attribute LeftIndent { xsd:double }?,
    attribute RightIndent { xsd:double }?,
    attribute SpaceBefore { xsd:double }?,
    attribute SpaceAfter { xsd:double }?,
    attribute Justification { Justification_EnumValue }?,
    attribute SingleWordJustification { SingleWordJustification_EnumValue }?,
    attribute AutoLeading { xsd:double }?,
    attribute DropCapLines { xsd:short {minInclusive="0" maxInclusive="25"} }?,
    attribute DropCapCharacters { xsd:short {minInclusive="0" maxInclusive="150"} }?,
    attribute KeepLinesTogether { xsd:boolean }?,
    attribute KeepAllLinesTogether { xsd:boolean }?,
    attribute KeepWithNext { xsd:short {minInclusive="0" maxInclusive="5"} }?,
    attribute KeepFirstLines { xsd:short {minInclusive="1" maxInclusive="50"} }?,
    attribute KeepLastLines { xsd:short {minInclusive="1" maxInclusive="50"} }?,
    attribute StartParagraph { StartParagraph_EnumValue }?,
    attribute Composer { xsd:string }?,
    attribute MinimumWordSpacing { xsd:double }?,
    attribute MaximumWordSpacing { xsd:double }?,
    attribute DesiredWordSpacing { xsd:double }?,
    attribute MinimumLetterSpacing { xsd:double }?,
    attribute MaximumLetterSpacing { xsd:double }?,
    attribute DesiredLetterSpacing { xsd:double }?,
    attribute MinimumGlyphScaling { xsd:double }?,
    attribute MaximumGlyphScaling { xsd:double }?,
    attribute DesiredGlyphScaling { xsd:double }?,
    attribute RuleAbove { xsd:boolean }?,
    attribute RuleAboveOverprint { xsd:boolean }?,
    attribute RuleAboveLineWidth { xsd:double }?,
    attribute RuleAboveTint { xsd:double }?,
    attribute RuleAboveOffset { xsd:double }?,
    attribute RuleAboveLeftIndent { xsd:double }?,
    attribute RuleAboveRightIndent { xsd:double }?,
    attribute RuleAboveWidth { RuleWidth_EnumValue }?,
```

```

        attribute RuleAboveGapTint { xsd:double }?,
        attribute RuleAboveGapOverprint { xsd:boolean }?,
        attribute RuleBelow { xsd:boolean }?,
        attribute RuleBelowLineWeight { xsd:double }?,
        attribute RuleBelowTint { xsd:double }?,
        attribute RuleBelowOffset { xsd:double }?,
        attribute RuleBelowLeftIndent { xsd:double }?,
        attribute RuleBelowRightIndent { xsd:double }?,
        attribute RuleBelowWidth { RuleWidth_EnumValue }?,
        attribute RuleBelowGapTint { xsd:double }?,
        attribute HyphenateCapitalizedWords { xsd:boolean }?,
        attribute Hyphenation { xsd:boolean }?,
        attribute HyphenateBeforeLast { xsd:short {minInclusive="1" maxInclusive="15"} }?,
        attribute HyphenateAfterFirst { xsd:short {minInclusive="1" maxInclusive="15"} }?,
        attribute HyphenateWordsLongerThan { xsd:short {minInclusive="3" maxInclusive="25"} }?,
        attribute HyphenateLadderLimit { xsd:short {minInclusive="0" maxInclusive="25"} }?,
        attribute HyphenationZone { xsd:double }?,
        attribute HyphenWeight { xsd:short {minInclusive="0" maxInclusive="10"} }?,
        attribute AppliedParagraphStyle { xsd:string }?,
        attribute AppliedCharacterStyle { xsd:string }?,
        attribute KerningValue { xsd:double }?,
        attribute FontStyle { xsd:string }?,
        attribute PointSize { xsd:double }?,
        attribute KerningMethod { xsd:string }?,
        attribute Tracking { xsd:double }?,
        attribute Capitalization { Capitalization_EnumValue }?,
        attribute Position { Position_EnumValue }?,
        attribute Underline { xsd:boolean }?,
        attribute StrikeThru { xsd:boolean }?,
        attribute Ligatures { xsd:boolean }?,
        attribute NoBreak { xsd:boolean }?,
        attribute HorizontalScale { xsd:double }?,
        attribute VerticalScale { xsd:double }?,
        attribute BaselineShift { xsd:double }?,
        attribute Skew { xsd:double }?,
        attribute FillTint { xsd:double }?,
        attribute StrokeTint { xsd:double }?,
        attribute StrokeWeight { xsd:double }?,
        attribute OverprintStroke { xsd:boolean }?,
        attribute OverprintFill { xsd:boolean }?,
        attribute OTFFigureStyle { OTFFigureStyle_EnumValue }?,
        attribute OTFOrdinal { xsd:boolean }?,
        attribute OTFFraction { xsd:boolean }?,
        attribute OTFDiscretionaryLigature { xsd:boolean }?,
        attribute OTFTitling { xsd:boolean }?,
        attribute OTFContextualAlternate { xsd:boolean }?,
        attribute OTFSwash { xsd:boolean }?,
        attribute UnderlineTint { xsd:double }?,
        attribute UnderlineGapTint { xsd:double }?,
        attribute UnderlineOverprint { xsd:boolean }?,

```

```

        attribute UnderlineGapOverprint { xsd:boolean }?,
        attribute UnderlineOffset { xsd:double }?,
        attribute UnderlineWeight { xsd:double }?,
        attribute StrikeThroughTint { xsd:double }?,
        attribute StrikeThroughGapTint { xsd:double }?,
        attribute StrikeThroughOverprint { xsd:boolean }?,
        attribute StrikeThroughGapOverprint { xsd:boolean }?,
        attribute StrikeThroughOffset { xsd:double }?,
        attribute StrikeThroughWeight { xsd:double }?,
        attribute FillColor { xsd:string }?,
        attribute StrokeColor { xsd:string }?,
        attribute AppliedLanguage { xsd:string }?,
        attribute LastLineIndent { xsd:double }?,
        attribute HyphenateLastWord { xsd:boolean }?,
        attribute OTFSashedZero { xsd:boolean }?,
        attribute OTFHistorical { xsd:boolean }?,
        attribute OTFStylisticSets { xsd:int }?,
        attribute GradientFillLength { xsd:double }?,
        attribute GradientFillAngle { xsd:double }?,
        attribute GradientStrokeLength { xsd:double }?,
        attribute GradientStrokeAngle { xsd:double }?,
        attribute GradientFillStart { UnitPointType_TypeDef }?,
        attribute GradientStrokeStart { UnitPointType_TypeDef }?,
        attribute RuleBelowOverprint { xsd:boolean }?,
        attribute RuleBelowGapOverprint { xsd:boolean }?,
        attribute DropcapDetail { xsd:int }?,
        attribute HyphenateAcrossColumns { xsd:boolean }?,
        attribute KeepRuleAboveInFrame { xsd:boolean }?,
        attribute IgnoreEdgeAlignment { xsd:boolean }?,
        attribute OTFMark { xsd:boolean }?,
        attribute OTFLocale { xsd:boolean }?,
        attribute PositionalForm { PositionalForms_EnumValue }?,
        attribute ParagraphDirection { ParagraphDirection_EnumValue }?,
        attribute ParagraphJustification { ParagraphJustification_EnumValue }?,
        attribute MiterLimit { xsd:double {minInclusive="0" maxInclusive="1000"} }?,
        attribute StrokeAlignment { TextStrokeAlign_EnumValue }?,
        attribute EndJoin { OutlineJoin_EnumValue }?,
        attribute OTFOverlapSwash { xsd:boolean }?,
        attribute OTFStylisticAlternate { xsd:boolean }?,
        attribute OTFJustificationAlternate { xsd:boolean }?,
        attribute OTFSretchedAlternate { xsd:boolean }?,
        attribute CharacterDirection { CharacterDirection_EnumValue }?,
        attribute KeyboardDirection { CharacterDirection_EnumValue }?,
        attribute DigitsType { DigitsType_EnumValue }?,
        attribute Kashidas { Kashidas_EnumValue }?,
        attribute DiacriticPosition { DiacriticPosition_EnumValue }?,
        attribute XOffsetDiacritic { xsd:double }?,
        attribute YOffsetDiacritic { xsd:double }?,
        attribute GotoNextX { GotoNextX_EnumValue }?,
        attribute PageNumberType { PageNumberType_EnumValue }?,
        attribute AppliedConditions { list { xsd:string * } }?,
        attribute CharacterAlignment { CharacterAlignment_EnumValue }?,
        attribute Tsume { xsd:double }?,
        attribute LeadingAki { xsd:double }?,

```

```

attribute TrailingAki { xsd:double }?,
attribute CharacterRotation { xsd:double }?,
attribute Jidori { xsd:short }?,
attribute ShataiMagnification { xsd:double }?,
attribute ShataiDegreeAngle { xsd:double }?,
attribute ShataiAdjustRotation { xsd:boolean }?,
attribute ShataiAdjustTsume { xsd:boolean }?,
attribute Tatechuyoko { xsd:boolean }?,
attribute TatechuyokoXOffset { xsd:double }?,
attribute TatechuyokoYOffset { xsd:double }?,
attribute KentenTint { xsd:double }?,
attribute KentenStrokeTint { xsd:double }?,
attribute KentenWeight { xsd:double }?,
attribute KentenOverprintFill { AdornmentOverprint_EnumValue }?,
attribute KentenOverprintStroke { AdornmentOverprint_EnumValue }?,
attribute KentenKind { KentenCharacter_EnumValue }?,
attribute KentenPlacement { xsd:double }?,
attribute KentenAlignment { KentenAlignment_EnumValue }?,
attribute KentenPosition { RubyKentenPosition_EnumValue }?,
attribute KentenFontSize { xsd:double }?,
attribute KentenXScale { xsd:double }?,
attribute KentenYScale { xsd:double }?,
attribute KentenCustomCharacter { xsd:string }?,
attribute KentenCharacterSet { KentenCharacterSet_EnumValue }?,
attribute RubyTint { xsd:double }?,
attribute RubyWeight { xsd:double }?,
attribute RubyOverprintFill { AdornmentOverprint_EnumValue }?,
attribute RubyOverprintStroke { AdornmentOverprint_EnumValue }?,
attribute RubyStrokeTint { xsd:double }?,
attribute RubyFontSize { xsd:double }?,
attribute RubyOpenTypePro { xsd:boolean }?,
attribute RubyXScale { xsd:double }?,
attribute RubyYScale { xsd:double }?,
attribute RubyType { RubyTypes_EnumValue }?,
attribute RubyAlignment { RubyAlignments_EnumValue }?,
attribute RubyPosition { RubyKentenPosition_EnumValue }?,
attribute RubyXOffset { xsd:double }?,
attribute RubyYOffset { xsd:double }?,
attribute RubyParentSpacing { RubyParentSpacing_EnumValue }?,
attribute RubyAutoAlign { xsd:boolean }?,
attribute RubyOverhang { xsd:boolean }?,
attribute RubyAutoScaling { xsd:boolean }?,
attribute RubyParentScalingPercent { xsd:double }?,
attribute RubyParentOverhangAmount { RubyOverhang_EnumValue }?,
attribute Warichu { xsd:boolean }?,
attribute WarichuSize { xsd:double }?,
attribute WarichuLines { xsd:short }?,
attribute WarichuLineSpacing { xsd:double }?,
attribute WarichuAlignment { WarichuAlignment_EnumValue }?,
attribute WarichuCharsAfterBreak { xsd:short }?,
attribute WarichuCharsBeforeBreak { xsd:short }?,
attribute OTFProportionalMetrics { xsd:boolean }?,
attribute OTFHVKana { xsd:boolean }?,
attribute OTFRomanItalics { xsd:boolean }?,

```

```

        attribute ScaleAffectsLineHeight { xsd:boolean }?,
        attribute CjkGridTracking { xsd:boolean }?,
        attribute GlyphForm { AlternateGlyphForms_EnumValue }?,
        attribute RubyFlag { xsd:int }?,
        attribute RubyString { xsd:string }?,
        attribute GridAlignFirstLineOnly { xsd:boolean }?,
        attribute GridAlignment { GridAlignment_EnumValue }?,
        attribute GridGyoudori { xsd:short }?,
        attribute AutoTcy { xsd:short }?,
        attribute AutoTcyIncludeRoman { xsd:boolean }?,
        attribute KinsokuType { KinsokuType_EnumValue }?,
        attribute KinsokuHangType { KinsokuHangTypes_EnumValue }?,
        attribute BunriKinshi { xsd:boolean }?,
        attribute Rensuuji { xsd:boolean }?,
        attribute RotateSingleByteCharacters { xsd:boolean }?,
        attribute LeadingModel { LeadingModel_EnumValue }?,
        attribute RubyAutoTcyDigits { xsd:short }?,
        attribute RubyAutoTcyIncludeRoman { xsd:boolean }?,
        attribute RubyAutoTcyAutoScale { xsd:boolean }?,
        attribute TreatIdeographicSpaceAsSpace { xsd:boolean }?,
        attribute AllowArbitraryHyphenation { xsd:boolean }?,
        attribute ParagraphGyoudori { xsd:boolean }?,
        attribute BulletsAndNumberingListType { ListType_EnumValue }?,
        attribute NumberingExpression { xsd:string }?,
        attribute BulletsTextAfter { xsd:string }?,
        attribute NumberingLevel { xsd:int }?,
        attribute NumberingContinue { xsd:boolean }?,
        attribute NumberingStartAt { xsd:int }?,
        attribute NumberingApplyRestartPolicy { xsd:boolean }?,
        attribute BulletsAlignment { ListAlignment_EnumValue }?,
        attribute NumberingAlignment { ListAlignment_EnumValue }?,
        element Properties {
            element BalanceRaggedLines {
                (bool_type, xsd:boolean ) |
                (enum_type, BalanceLinesStyle_EnumValue )
            }?&
            element RuleAboveColor {
                (object_type, xsd:string ) |
                (string_type, xsd:string )
            }?&
            element RuleAboveGapColor {
                (object_type, xsd:string ) |
                (string_type, xsd:string )
            }?&
            element RuleAboveType {
                (object_type, xsd:string ) |
                (string_type, xsd:string )
            }?&
            element RuleBelowColor {
                (object_type, xsd:string ) |
                (string_type, xsd:string )
            }?&
            element RuleBelowGapColor {
                (object_type, xsd:string ) |

```

```

        (string_type, xsd:string )
    }?&
    element RuleBelowType {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element AllNestedStyles { list_type, element ListItem {
        record_type,
        (
            element AppliedCharacterStyle { object_type, xsd:string }&
            element Delimiter {
                (string_type, xsd:string ) |
                (enum_type, NestedStyleDelimiters_EnumValue )
            }&
            element Repetition { long_type, xsd:int }&
            element Inclusive { bool_type, xsd:boolean })
        }*
    }?&
    element TabList { list_type, element ListItem {
        record_type,
        (
            element Alignment { enum_type, TabStopAlignment_EnumValue }&
            element AlignmentCharacter { string_type, xsd:string }&
            element Leader { string_type, xsd:string }&
            element Position { unit_type, xsd:double })
        }*
    }?&
    element AppliedFont {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element Leading {
        (unit_type, xsd:double ) |
        (enum_type, Leading_EnumValue )
    }?&
    element UnderlineColor {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element UnderlineGapColor {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element UnderlineType {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element StrikeThroughColor {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element StrikeThroughGapColor {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }

```

```

} ?&
element StrikeThroughType {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element CustomGlyph {
  (long_type, xsd:int) |
  (string_type, xsd:string)
} ?&
element AllLineStyles { list_type, element ListItem {
  record_type,
  (
    element AppliedCharacterStyle { object_type, xsd:string } &
    element LineCount { long_type, xsd:int } &
    element RepeatLast { long_type, xsd:int })
  } *
} ?&
element AllGREPstyles { list_type, element ListItem {
  record_type,
  (
    element AppliedCharacterStyle { object_type, xsd:string } &
    element GrepExpression { string_type, xsd:string })
  } *
} ?&
element OpenTypeFeatures { list_type, element ListItem {
  list_type,
  element ListItem {
    (string_type, xsd:string) |
    (long_type, xsd:int)
  },
  element ListItem {
    (string_type, xsd:string) |
    (long_type, xsd:int)
  }
} *
} ?&
element KentenFillColor {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element KentenStrokeColor {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element KentenFont {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element KentenFontStyle {
  (string_type, xsd:string) |
  (enum_type, NothingEnum_EnumValue)
} ?&
element RubyFill {
  (object_type, xsd:string) |

```

```

(string_type, xsd:string )
)?&
element RubyStroke {
  (object_type, xsd:string ) |
  (string_type, xsd:string )
)?&
element RubyFont {
  (object_type, xsd:string ) |
  (string_type, xsd:string )
)?&
element RubyFontStyle {
  (string_type, xsd:string ) |
  (enum_type, NothingEnum_EnumValue )
)?&
element KinsokuSet {
  (object_type, xsd:string ) |
  (enum_type, KinsokuSet_EnumValue ) |
  (string_type, xsd:string )
)?&
element Mojikumi {
  (object_type, xsd:string ) |
  (string_type, xsd:string ) |
  (enum_type, MojikumiTableDefaults_EnumValue )
)?&
element BulletChar {
  attribute BulletCharacterType { BulletCharacterType_EnumValue },
  attribute BulletCharacterValue { xsd:int }
)?&
element BulletsFont {
  (object_type, xsd:string ) |
  (string_type, xsd:string ) |
  (enum_type, AutoEnum_EnumValue )
)?&
element BulletsFontStyle {
  (string_type, xsd:string ) |
  (enum_type, NothingEnum_EnumValue ) |
  (enum_type, AutoEnum_EnumValue )
)?&
element BulletsCharacterStyle {
  (object_type, xsd:string ) |
  (string_type, xsd:string )
)?&
element NumberingCharacterStyle {
  (object_type, xsd:string ) |
  (string_type, xsd:string )
)?&
element AppliedNumberingList {
  (object_type, xsd:string ) |
  (string_type, xsd:string )
)?&
element NumberingFormat {
  (enum_type, NumberingStyle_EnumValue ) |
  (string_type, xsd:string )
)?&

```

```

        element NumberingRestartPolicies {
            attribute RestartPolicy { RestartPolicy_EnumValue },
            attribute LowerLevel { xsd:int },
            attribute UpperLevel { xsd:int }
        }?
    }
?
'
(
    PageReference_Object*&
    Note_Object*&
    Table_Object*&
    TextVariableInstance_Object*&
    Footnote_Object*&
    TextFrame_Object*&
    Oval_Object*&
    Rectangle_Object*&
    GraphicLine_Object*&
    Polygon_Object*&
    Group_Object*&
    EPSText_Object*&
    FormField_Object*&
    Button_Object*&
    HyperlinkTextDestination_Object*&
    ParagraphDestination_Object*&
    Change_Object*&
    XMLElement_Object*&
    XMLComment_Object*&
    XMLInstruction_Object1*&
    DTD_Object*&
    HiddenText_Object*&
    HyperlinkTextSource_Object*&
    CrossReferenceSource_Object*&
    element Content {text}*&
    element Br {empty}*
)
}

```

### **CharacterStyleRange Attributes**

The properties of a <CharacterStyleRange> that are expressed as attributes can be found in the listing of common text element attributes (see “Common Text Properties”).

### **CharacterStyleRange Elements**

Most of the properties of a <CharacterStyleRange> that are expressed as elements can be found in the listing of common text elements (see “Common Text Properties”). A <CharacterStyleRange> can also contain an element that is unique. This element is described in the following table.

**Table 52. CharacterStyleRange Elements**

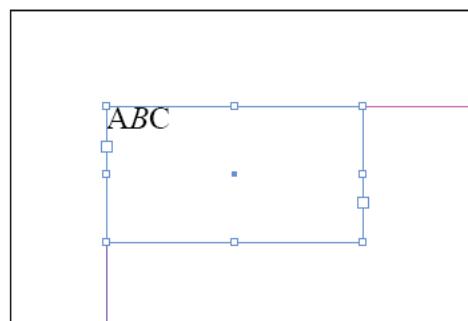
Name	Type	Req	Description
Content	string	yes	The content of the <CharacterStyleRange>. Can be a string of text, or a reference to an inline element.

In the following example, the story contains three characters, “A,” “B”, and “C.” The middle character has slightly different formatting than the surrounding characters (it’s been formatted using the font style “Italic”), and so occupies a different character style range. The story has only one <ParagraphStyleRange>, and three <CharacterStyleRange> elements.

#### IDML Example 38. ParagraphStyleRange and CharacterStyleRange

```
<Story Self="ucb" AppliedTOCStyle="n" TrackChanges="false"
StoryTitle="$ID/" AppliedNamedGrid="n">
<StoryPreference Self="ucbStoryPreference1" OpticalMarginAlignment="false"
OpticalMarginSize="12" FrameType="TextFrameType" StoryOrientation="Horizontal"
StoryDirection="LeftToRightDirection"/>
<ParagraphStyleRange AppliedParagraphStyle=
"ParagraphStyle\kNormalParagraphStyle">
<CharacterStyleRange AppliedCharacterStyle=
"CharacterStyle\k[No character style]">
<Content>A</Content>
</CharacterStyleRange>
<CharacterStyleRange AppliedCharacterStyle=
"CharacterStyle\k[No character style]" FontStyle="Italic">
<Content>B</Content>
</CharacterStyleRange>
<CharacterStyleRange AppliedCharacterStyle=
"CharacterStyle\k[No character style]">
<Content>C</Content>
</CharacterStyleRange>
</ParagraphStyleRange>
</Story>
```

Figure 30. Minimal Story with Multiple Character Formats

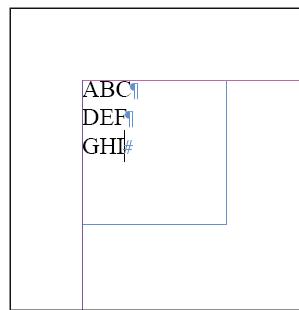


Multiple paragraphs can appear in a single <ParagraphStyleRange>. The following example shows how the text in the single <CharacterStyleRange> in a <ParagraphStyleRange> is broken into separate <Content> elements using the <br/> element. An empty <br/> element represents a return character; optional attributes attached to the element can specify other break characters. We’ve omitted the minimal <Story> element in this example.

**IDML Example 39. Paragraph Breaks in a CharacterStyleRange**

```
<ParagraphStyleRange AppliedParagraphStyle="ParagraphStyle\NormalParagraphStyle">
  <CharacterStyleRange AppliedCharacterStyle="CharacterStyle\[No character style]">
    <Content>ABC</Content>
    <br/>
    <Content>DEF</Content>
    <br/>
    <Content>GHI</Content>
  </CharacterStyleRange>
</ParagraphStyleRange>
```

*Figure 31. Breaking <Content> elements with <br/>*



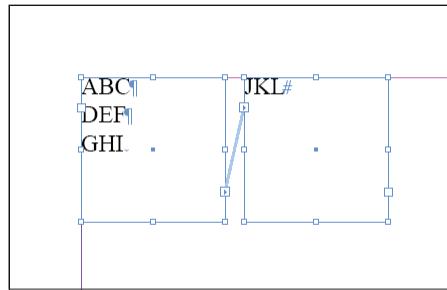
The following example demonstrates the use of special “break” characters in a `<CharacterStyleRange>` element. We’ve inserted a column break after the third example paragraph (“GHI”), which forces the fourth paragraph (“JKL”) to the next column or text frame. The break character is applied as an attribute of the `<CharacterStyleRange>` element. The `<Content>` element of that `<CharacterStyleRange>` element contains only a `<br/>` element.

The meaning of the `<br/>` element is defined by the `GoToNextX` attribute of the `<CharacterStyleRange>` element. This attribute can be `Anywhere` (the default, which can be omitted), `NextColumn`, `NextFrame`, `NextPage`, `NextOddPage`, or `NextEvenPage`.

**IDML Example 40. ParagraphStyleRange Elements and Column Breaks**

```
<ParagraphStyleRange AppliedParagraphStyle="ParagraphStyle\NormalParagraphStyle">
  <CharacterStyleRange AppliedCharacterStyle="CharacterStyle\[No character style]">
    <Content>ABC</Content>
    <br/>
    <Content>DEF</Content>
    <br/>
    <Content>GHI</Content>
  </CharacterStyleRange>
  <CharacterStyleRange AppliedCharacterStyle="CharacterStyle\[No character style]">
    GotoNextX="NextColumn">
      <br/>
    </CharacterStyleRange>
    <CharacterStyleRange AppliedCharacterStyle="CharacterStyle\[No character style]">
      <Content>JKL</Content>
    </CharacterStyleRange>
  </CharacterStyleRange>
</ParagraphStyleRange>
```

Figure 32. Column Break



**Note:** In general, it is better to use paragraph style properties (the `StartParagraph` attribute of the `<ParagraphStyle>` element) to force paragraphs to new columns or pages instead of using break characters.

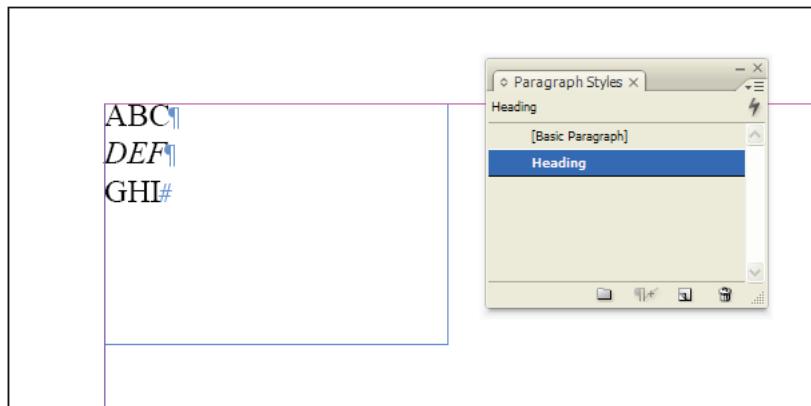
The following example shows an example of multiple `<ParagraphStyleRange>` elements. In this example, the second paragraph ("DEF") has different formatting from the first ("ABC") and third ("GHI") paragraphs—it has a paragraph style named "Heading" applied to it.

#### IDML Example 41. Multiple ParagraphStyleRange Elements

```

<ParagraphStyleRange AppliedParagraphStyle="ParagraphStyle\NormalParagraphStyle">
  <CharacterStyleRange AppliedCharacterStyle=
    "CharacterStyle\[No character style]">
    <Content>ABC</Content>
    <br/>
  </CharacterStyleRange>
</ParagraphStyleRange>
<ParagraphStyleRange AppliedParagraphStyle="ParagraphStyle\Heading">
  <CharacterStyleRange AppliedCharacterStyle="CharacterStyle\[No character style]">
    <Content>DEF</Content>
    <br/>
  </CharacterStyleRange>
</ParagraphStyleRange>
<ParagraphStyleRange AppliedParagraphStyle="ParagraphStyle\NormalParagraphStyle">
  <CharacterStyleRange AppliedCharacterStyle="CharacterStyle\[No character style]">
    <Content>GHI</Content>
  </CharacterStyleRange>
</ParagraphStyleRange>

```

Figure 33. Minimal Story with Multiple `<ParagraphStyleRange>` Elements

#### 10.4.9 XML Elements in Text

All XML text elements that have been placed in an InDesign layout appear as text ranges in the `<Story>` element they are associated with. Only unplaced XML elements (i.e., XML elements that have not been associated with a page item or story) appear in the `BackingStory.xml` file in the XML folder in the IDML package.

The text in `<XMLElement>` elements is contained by `<ParagraphStyleRange>` elements and `<CharacterStyleRange>` elements inside the `<XMLElement>` elements. For more on the `<XMLElement>` element, refer to the "XML."

##### IDML Example 42. XML Element

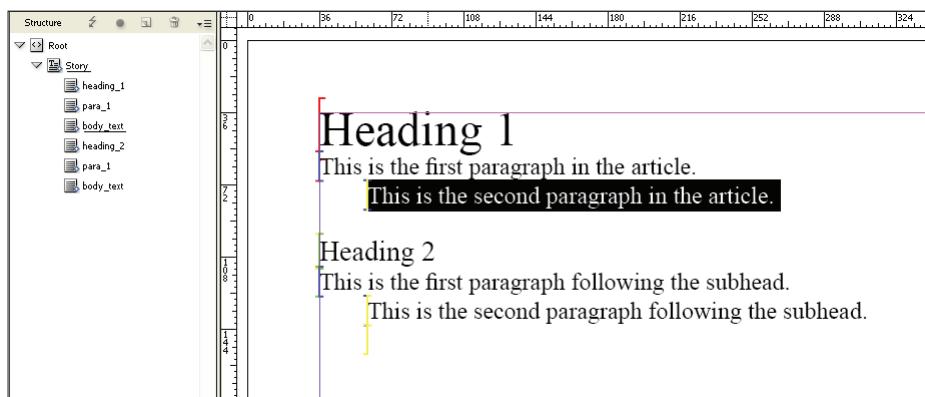
```
<Story Self="ud8">
  <XMLElement Self="di2i3" MarkupTag="XMLTag\cStory" XMLContent="ud8">
    <ParagraphStyleRange AppliedParagraphStyle="ParagraphStyle\cheading_1">
      <CharacterStyleRange AppliedCharacterStyle=
        "CharacterStyle\k[No character style]" />
    </ParagraphStyleRange>
    <XMLElement Self="di2i3i6" MarkupTag="XMLTag\cheading_1">
      <ParagraphStyleRange AppliedParagraphStyle="ParagraphStyle\cheading_1">
        <CharacterStyleRange AppliedCharacterStyle=
          "CharacterStyle\k[No character style]" />
          <Content>Heading 1</Content>
          <br/>
        </CharacterStyleRange>
      </ParagraphStyleRange>
    </XMLElement>
    <ParagraphStyleRange AppliedParagraphStyle="ParagraphStyle\cpa_1">
      <CharacterStyleRange AppliedCharacterStyle=
        "CharacterStyle\k[No character style]" />
    </ParagraphStyleRange>
    <XMLElement Self="di2i3i5" MarkupTag="XMLTag\cpa_1">
      <ParagraphStyleRange AppliedParagraphStyle="ParagraphStyle\cpa_1">
        <CharacterStyleRange AppliedCharacterStyle=
          "CharacterStyle\k[No character style]" />
          <Content>This is the first paragraph in the article.</Content>
          <br/>
      </ParagraphStyleRange>
    </XMLElement>
  </XMLElement>
</Story>
```

```

        </CharacterStyleRange>
    </ParagraphStyleRange>
</XMLElement>
<ParagraphStyleRange AppliedParagraphStyle="ParagraphStyle\cbody_text">
    <CharacterStyleRange AppliedCharacterStyle=
        "CharacterStyle\k[No character style]">
    </CharacterStyleRange>
<XMLElement Self="di2i3i4" MarkupTag="XMLTag\cbody_text">
    <ParagraphStyleRange AppliedParagraphStyle="ParagraphStyle\cbody_text">
        <CharacterStyleRange AppliedCharacterStyle=
            "CharacterStyle\k[No character style]">
            <Content>This is the second paragraph in the article.</Content>
            <br/>
        </CharacterStyleRange>
    </ParagraphStyleRange>
</XMLElement>
<ParagraphStyleRange AppliedParagraphStyle="ParagraphStyle\cheading_2">
    <CharacterStyleRange AppliedCharacterStyle=
        "CharacterStyle\k[No character style]">
    </CharacterStyleRange>
<XMLElement Self="di2i3i3" MarkupTag="XMLTag\cheading_2">
    <ParagraphStyleRange AppliedParagraphStyle="ParagraphStyle\cheading_2">
        <CharacterStyleRange AppliedCharacterStyle=
            "CharacterStyle\k[No character style]">
            <Content>Heading 2</Content>
            <br/>
        </CharacterStyleRange>
    </ParagraphStyleRange>
</XMLElement>
<ParagraphStyleRange AppliedParagraphStyle="ParagraphStyle\cpara_1">
    <CharacterStyleRange AppliedCharacterStyle=
        "CharacterStyle\k[No character style]">
    </CharacterStyleRange>
<XMLElement Self="di2i3i2" MarkupTag="XMLTag\cpara_1">
    <ParagraphStyleRange AppliedParagraphStyle="ParagraphStyle\cpara_1">
        <CharacterStyleRange AppliedCharacterStyle=
            "CharacterStyle\k[No character style]">
            <Content>This is the first paragraph following the subhead.</Content>
            <br/>
        </CharacterStyleRange>
    </ParagraphStyleRange>
</XMLElement>
<ParagraphStyleRange AppliedParagraphStyle="ParagraphStyle\cbody_text">
    <CharacterStyleRange AppliedCharacterStyle=
        "CharacterStyle\k[No character style]">
        <XMLElement Self="di2i3i1" MarkupTag="XMLTag\cbody_text">
            <Content>This is the second paragraph following the subhead.</Content>
            <br/>
        </XMLElement>
    </CharacterStyleRange>
</ParagraphStyleRange>
</XMLElement>
</Story>

```

Figure 34. XML Elements in Text



#### 10.4.10 Rules for Breaking Text Range Elements

In a complex InDesign layout, situations can arise in which the logical boundaries for creating text range elements overlap. What happens when an `<XMLElement>` element overlaps a `<CharacterStyleRange>` element? Or when a `<CharacterStyleRange>` element appears in the middle of a `<HyperlinkTextSource>` element?

InDesign relies on a hierarchy of rules for deciding where to break text style ranges as it exports IDML. The following list shows the order in which InDesign will break text style range elements, from strongest (never break) to weakest.

1. `<XML>`
2. `<Hyperlink>`
3. `<ChangedText>`
4. `<ParagraphStyleRange>`
5. `<CharacterStyleRange>`
6. `<Content>`

#### 6.4.1 Inline Elements

InDesign stories can contain page items, such as rectangles containing imported graphics or text frames containing text. Tables, footnotes and form fields are other examples of inline objects. These inline (or anchored) element are represented as text child elements of a `<CharacterStyleRange>` element, and appear as siblings of the `<Content>` element. The following sections discuss the different types of inline elements that can appear as text child elements of a `<CharacterStyleRange>` element.

##### **Tables**

InDesign tables contain rows, columns, and cells, and each cell can contain text, imported graphics, or another table. In IDML, the `<Table>` element contains `<Row>`, `<Column>`, and `<Cell>`

elements, which can, in turn, contain other elements. For more on InDesign tables, refer to the online help.

#### **Schema Example 45. Table**

```
Table_Object = element Table {
    attribute Self { xsd:string },
    attribute HeaderRowCount { xsd:int {minInclusive="0" maxInclusive="25"} }?,
    attribute FooterRowCount { xsd:int {minInclusive="0" maxInclusive="25"} }?,
    attribute TopBorderStrokeWeight { xsd:double }?,
    attribute TopBorderStrokeType { xsd:string }?,
    attribute TopBorderStrokeColor { xsd:string }?,
    attribute TopBorderStrokeTint { xsd:double }?,
    attribute TopBorderStrokeOverprint { xsd:boolean }?,
    attribute TopBorderStrokeGapColor { xsd:string }?,
    attribute TopBorderStrokeGapTint { xsd:double }?,
    attribute TopBorderStrokeGapOverprint { xsd:boolean }?,
    attribute LeftBorderStrokeWeight { xsd:double }?,
    attribute LeftBorderStrokeType { xsd:string }?,
    attribute LeftBorderStrokeColor { xsd:string }?,
    attribute LeftBorderStrokeTint { xsd:double }?,
    attribute LeftBorderStrokeOverprint { xsd:boolean }?,
    attribute LeftBorderStrokeGapColor { xsd:string }?,
    attribute LeftBorderStrokeGapTint { xsd:double }?,
    attribute LeftBorderStrokeGapOverprint { xsd:boolean }?,
    attribute BottomBorderStrokeWeight { xsd:double }?,
    attribute BottomBorderStrokeType { xsd:string }?,
    attribute BottomBorderStrokeColor { xsd:string }?,
    attribute BottomBorderStrokeTint { xsd:double }?,
    attribute BottomBorderStrokeOverprint { xsd:boolean }?,
    attribute BottomBorderStrokeGapColor { xsd:string }?,
    attribute BottomBorderStrokeGapTint { xsd:double }?,
    attribute BottomBorderStrokeGapOverprint { xsd:boolean }?,
    attribute RightBorderStrokeWeight { xsd:double }?,
    attribute RightBorderStrokeType { xsd:string }?,
    attribute RightBorderStrokeColor { xsd:string }?,
    attribute RightBorderStrokeTint { xsd:double }?,
    attribute RightBorderStrokeOverprint { xsd:boolean }?,
    attribute RightBorderStrokeGapColor { xsd:string }?,
    attribute RightBorderStrokeGapTint { xsd:double }?,
    attribute RightBorderStrokeGapOverprint { xsd:boolean }?,
    attribute SpaceBefore { xsd:double }?,
    attribute SpaceAfter { xsd:double }?,
    attribute SkipFirstAlternatingStrokeRows { xsd:int }?,
    attribute SkipLastAlternatingStrokeRows { xsd:int }?,
    attribute StartRowStrokeCount { xsd:int }?,
    attribute StartRowStrokeColor { xsd:string }?,
    attribute StartRowStrokeWeight { xsd:double }?,
    attribute StartRowStrokeType { xsd:string }?,
    attribute StartRowStrokeTint { xsd:double }?,
    attribute StartRowStrokeGapOverprint { xsd:boolean }?,
    attribute StartRowStrokeGapColor { xsd:string }?,
    attribute StartRowStrokeGapTint { xsd:double }?,
    attribute StartRowStrokeOverprint { xsd:boolean }?,
    attribute EndRowStrokeCount { xsd:int }?,
```

```

        attribute EndRowStrokeColor { xsd:string }?,
        attribute EndRowStrokeWeight { xsd:double }?,
        attribute EndRowStrokeType { xsd:string }?,
        attribute EndRowStrokeTint { xsd:double }?,
        attribute EndRowStrokeOverprint { xsd:boolean }?,
        attribute EndRowStrokeGapColor { xsd:string }?,
        attribute EndRowStrokeGapTint { xsd:double }?,
        attribute EndRowStrokeGapOverprint { xsd:boolean }?,
        attribute SkipFirstAlternatingStrokeColumns { xsd:int }?,
        attribute SkipLastAlternatingStrokeColumns { xsd:int }?,
        attribute StartColumnStrokeCount { xsd:int }?,
        attribute StartColumnStrokeColor { xsd:string }?,
        attribute StartColumnStrokeWeight { xsd:double }?,
        attribute StartColumnStrokeType { xsd:string }?,
        attribute StartColumnStrokeTint { xsd:double }?,
        attribute StartColumnStrokeOverprint { xsd:boolean }?,
        attribute StartColumnStrokeGapColor { xsd:string }?,
        attribute StartColumnStrokeGapTint { xsd:double }?,
        attribute StartColumnStrokeGapOverprint { xsd:boolean }?,
        attribute EndColumnStrokeCount { xsd:int }?,
        attribute EndColumnStrokeColor { xsd:string }?,
        attribute EndColumnStrokeWeight { xsd:double }?,
        attribute EndColumnLineStyle { xsd:string }?,
        attribute EndColumnStrokeTint { xsd:double }?,
        attribute EndColumnStrokeOverprint { xsd:boolean }?,
        attribute EndColumnStrokeGapColor { xsd:string }?,
        attribute EndColumnStrokeGapTint { xsd:double }?,
        attribute EndColumnStrokeGapOverprint { xsd:boolean }?,
        attribute ColumnFillsPriority { xsd:boolean }?,
        attribute SkipFirstAlternatingFillRows { xsd:int }?,
        attribute SkipLastAlternatingFillRows { xsd:int }?,
        attribute StartRowFillColor { xsd:string }?,
        attribute StartRowCount { xsd:int }?,
        attribute StartRowFillTint { xsd:double }?,
        attribute StartRowFillOverprint { xsd:boolean }?,
        attribute EndRowCount { xsd:int }?,
        attribute EndRowFillColor { xsd:string }?,
        attribute EndRowFillTint { xsd:double }?,
        attribute EndRowFillOverprint { xsd:boolean }?,
        attribute SkipFirstAlternatingFillColumns { xsd:int }?,
        attribute SkipLastAlternatingFillColumns { xsd:int }?,
        attribute StartColumnRowCount { xsd:int }?,
        attribute StartColumnFillColor { xsd:string }?,
        attribute StartColumnFillTint { xsd:double }?,
        attribute StartColumnFillOverprint { xsd:boolean }?,
        attribute EndColumnRowCount { xsd:int }?,
        attribute EndColumnFillColor { xsd:string }?,
        attribute EndColumnFillTint { xsd:double }?,
        attribute EndColumnFillOverprint { xsd:boolean }?,
        attribute BreakHeaders { HeaderFooterBreakTypes_EnumValue }?,
        attribute BreakFooters { HeaderFooterBreakTypes_EnumValue }?,
        attribute SkipFirstHeader { xsd:boolean }?,
        attribute SkipLastFooter { xsd:boolean }?,
        attribute StrokeOrder { StrokeOrderTypes_EnumValue }?,

```

```

        attribute DefaultRowStrokeWeight { xsd:double }?,
        attribute DefaultRowStrokeType { xsd:string }?,
        attribute DefaultRowStrokeColor { xsd:string }?,
        attribute DefaultRowStrokeTint { xsd:double }?,
        attribute DefaultRowStrokeOverprint { xsd:boolean }?,
        attribute DefaultRowStrokeGapColor { xsd:string }?,
        attribute DefaultRowStrokeGapTint { xsd:double }?,
        attribute DefaultRowStrokeGapOverprint { xsd:boolean }?,
        attribute DefaultColumnStrokeWeight { xsd:double }?,
        attribute DefaultColumnStrokeType { xsd:string }?,
        attribute DefaultColumnStrokeColor { xsd:string }?,
        attribute DefaultColumnStrokeTint { xsd:double }?,
        attribute DefaultColumnStrokeOverprint { xsd:boolean }?,
        attribute DefaultColumnStrokeGapColor { xsd:string }?,
        attribute DefaultColumnStrokeGapTint { xsd:double }?,
        attribute DefaultColumnStrokeGapOverprint { xsd:boolean }?,
        attribute TopInset { xsd:double }?,
        attribute LeftInset { xsd:double }?,
        attribute BottomInset { xsd:double }?,
        attribute RightInset { xsd:double }?,
        attribute FillColor { xsd:string }?,
        attribute FillTint { xsd:double }?,
        attribute OverprintFill { xsd:boolean }?,
        attribute TopLeftDiagonalLine { xsd:boolean }?,
        attribute TopRightDiagonalLine { xsd:boolean }?,
        attribute DiagonalLineInFront { xsd:boolean }?,
        attribute DiagonalLineStrokeWeight { xsd:double }?,
        attribute DiagonalLineStrokeType { xsd:string }?,
        attribute DiagonalLineStrokeColor { xsd:string }?,
        attribute DiagonalLineStrokeTint { xsd:double }?,
        attribute DiagonalLineStrokeOverprint { xsd:boolean }?,
        attribute DiagonalLineStrokeGapColor { xsd:string }?,
        attribute DiagonalLineStrokeGapTint { xsd:double }?,
        attribute DiagonalLineStrokeGapOverprint { xsd:boolean }?,
        attribute ClipContentToCell { xsd:boolean }?,
        attribute FirstBaselineOffset { FirstBaseline_EnumValue }?,
        attribute VerticalJustification { VerticalJustification_EnumValue }?,
        attribute ParagraphSpacingLimit { xsd:double }?,
        attribute MinimumFirstBaselineOffset { xsd:double {minInclusive="0" max-
Inclusive="8640"} }?,
        attribute RotationAngle { xsd:double }?,
        attribute WritingDirection { xsd:boolean }?,
        attribute MinimumHeight { xsd:double }?,
        attribute MaximumHeight { xsd:double }?,
        attribute KeepWithNextRow { xsd:boolean }?,
        attribute StartRow { StartParagraph_EnumValue }?,
        attribute AutoGrow { xsd:boolean }?,
        attribute BodyRowCount { xsd:int {minInclusive="1" maxInclusive="10000"} }?,
        attribute ColumnCount { xsd:int {minInclusive="1" maxInclusive="200"} }?,
        attribute SingleRowHeight { xsd:double }?,
        attribute SingleColumnWidth { xsd:double }?,
        attribute AppliedTableStyle { xsd:string }?,
        attribute TableDirection { TableDirection_EnumValue }?,
        attribute DisplayCollapsed { xsd:boolean }?,

```

```

        attribute DisplayOrder { DisplayOrderOptions_EnumValue }?,
        element Properties {
            element Label { element KeyValuePair { KeyValuePair_TypeDef }*
                }?
            ?
            ,
            (
                Cell_Object*&
                Row_Object*&
                Column_Object*
            )
        }
    }
}

```

**Table 53. Table Properties Represented as Attributes**

Name	Type	Req	Description
AppliedTableStyle	string	no	The table style applied to the table.
AutoGrow	boolean	no	If true, increase the height of a cell to fit the content of the cell.
BodyRowCount	int	no	The number of body rows.
BottomBorder-StrokeColor	string	no	The color, specified as a swatch (color, gradient, tint, or mixed ink), of the bottom border stroke.
BottomBorder-StrokeGapColor	string	no	The gap color, specified as a swatch (color, gradient, tint, or mixed ink), of the bottom border stroke. Note: Valid only when bottom border stroke type is not solid.
BottomBorder-StrokeGap-Overprint	boolean	no	If true, the gap of the bottom border stroke will overprint. Note: Valid only when bottom border stroke type is not solid.
BottomBorder-StrokeGapTint	double	no	The tint (as a percentage) of the gap color of the bottom border stroke. (Range: 0 to 100) Note: Valid only when bottom border stroke type is not solid.
BottomBorder-StrokeOverprint	boolean	no	If true, the bottom border stroke will overprint.
BottomBorder-StrokeTint	double	no	The tint (as a percentage) of the bottom border stroke. (Range: 0 to 100)
BottomBorder-StrokeType	string	no	The stroke type of the bottom border.
BottomBorder-StrokeWeight	double	no	The stroke weight of the bottom border stroke.
BreakFooters	HeaderFooter-BreakTypes_Enum-Value	no	The footer placement. Can be InAllText-Columns (Places headers or footers in each text column), OncePerTextFrame (Repeats headers or footers in each text frame), or OncePerPage (Places one instance of headers or footers per page).

Name	Type	Req	Description
BreakHeaders	HeaderFooter-BreakTypes_Enum-Value	no	The header placement. Can be InAllText-Columns (Places headers or footers in each text column), OncePerTextFrame (Repeats headers or footers in each text frame), or OncePerPage (Places one instance of headers or footers per page).
ColumnCount	int	no	The number of columns.
ColumnFills-Priority	boolean	no	If true, hides alternating row fills. If false, hides alternating column fills.
DefaultColumn-StrokeColor	string	no	The default stroke color for cells in new columns.
DefaultColumn-StrokeGapColor	string	no	The default stroke gap color for cells in new columns.
DefaultColumn-StrokeGap-Overprint	boolean	no	The default stroke gap overprint setting for cells in new columns.
DefaultColumn-StrokeGapTint	double	no	The default stroke gap tint for cells in new columns.
DefaultColumn-StrokeOverprint	boolean	no	The default stroke overprint setting for cells in new columns.
DefaultColumn-StrokeTint	double	no	The default stroke tint for cells in new columns.
DefaultColumn-StrokeType	string	no	The default stroke type for cells in new columns.
DefaultColumn-StrokeWeight	double	no	The default stroke weight for cells in new columns.
DefaultRowStroke-Color	string	no	The default stroke color for cells in new rows.
DefaultRowStroke-GapColor	string	no	The default stroke gap color for cells in new rows.
DefaultRowStroke-GapOverprint	boolean	no	The default stroke gap overprint setting for cells in new rows.
DefaultRowStroke-GapTint	double	no	The default stroke gap tint for cells in new rows.
DefaultRowStroke-Overprint	boolean	no	The default stroke overprint setting for cells in new rows.
DefaultRowStroke-Tint	double	no	The default stroke tint for cells in new rows.
DefaultRowStroke-Type	string	no	The default stroke type for cells in new rows.
DefaultRowStroke-Weight	double	no	The default stroke weight for cells in new columns.
DisplayCollapsed	boolean	no	If true, then the table will show collapsed in story and galley views.

Name	Type	Req	Description
DisplayOrder	DisplayOrder-Options_EnumValue	no	Specifies the order the table cells will display in when viewing in story and galley views. Can be OrderByRows (Order by rows), or OrderByColumns (Order by columns).
EndColumnFill-Color	string	no	The fill color, specified as a swatch (color, gradient, tint, or mixed ink), of columns in the second alternating fill group. Note: Valid when alternating fills are defined for table columns.
EndColumnFill-Count	int	no	The number of columns in the second alternating fills group. Note: Valid when alternating fills are defined for table columns.
EndColumnFill-Overprint	boolean	no	If true, the columns in the second alternating fills group will overprint. Note: Valid when alternating fills are defined for table columns.
EndColumnFillTint	double	no	The tint (as a percentage) of the columns in the second alternating fills group. (Range: 0 to 100) Note: Valid when alternating fills are defined for table columns.
EndColumnLine-Style	string	no	The stroke type of columns in the second alternating strokes group.
EndColumnStroke-Color	string	no	The stroke color, specified as a swatch (color, gradient, tint, or mixed ink), of column borders in the second alternating column strokes group. Note: Valid when end column stroke count is 1 or greater.
EndColumnStroke-Count	int	no	The number of columns in the second alternating column strokes group.
EndColumnStroke-GapColor	string	no	The stroke gap color, specified as a swatch (color, gradient, tint, or mixed ink), of column borders in the second alternating column strokes group. Note: Valid when end column stroke count is 1 or greater.
EndColumnStroke-GapOverprint	boolean	no	If true, the gap of the column border stroke in the second alternating column strokes group will overprint. Note: Valid when end column stroke count is 1 or greater.
EndColumnStroke-GapTint	double	no	The tint (as a percentage) of the gap color of column borders in the second alternating column strokes group. (Range: 0 to 100) Note: Valid when end column stroke count is 1 or greater.
EndColumnStroke-Overprint	boolean	no	If true, the column borders in the second alternating column strokes group will overprint. Note: Valid when end column stroke count is 1 or greater.
EndColumnStroke-Tint	double	no	The tint (as a percentage) of column borders in the second alternating column strokes group. (Range: 0 to 100) Note: Valid when end column stroke count is 1 or greater.

Name	Type	Req	Description
EndColumnStroke-Weight	double	no	The stroke weight of column borders in the second alternating column strokes group. Note: Valid when end column stroke count is 1 or greater.
EndRowFillColor	string	no	The fill color, specified as a swatch (color, gradient, tint, or mixed ink), of rows in the second alternating fills group. Note: Valid when alternating fills are defined for table rows.
EndRowFillCount	int	no	The number of rows in the second alternating fills group. Note: Valid when alternating fills are defined for table rows.
EndRowFill-Overprint	boolean	no	If true, the rows in the second alternating fills group will overprint. Note: Valid when alternating fills are defined for table rows.
EndRowFillTint	double	no	The tint (as a percentage) of the rows in the second alternating fills group. (Range: 0 to 100) Note: Valid when alternating fills are defined for table rows.
EndRowStrokeColor	string	no	The stroke color, specified as a swatch (color, gradient, tint, or mixed ink), of row borders in the second alternating row strokes group. Note: Valid when end row stroke count is 1 or greater.
EndRowStrokeCount	int	no	The number of rows in the second alternating row strokes group.
EndRowStrokeGap-Color	string	no	The gap color, specified as a swatch (color, gradient, tint, or mixed ink), of row borders in the second alternating rows group. Note: Valid when end row stroke count is 1 or greater.
EndRowStrokeGap-Overprint	boolean	no	If true, the gap of the row borders in the second alternating rows group will overprint. Note: Valid when end row stroke count is 1 or greater.
EndRowStrokeGap-Tint	double	no	The tint (as a percentage) of the gap color of rows in the second alternating strokes group. (Range: 0 to 100) Note: Valid when end row stroke count is 1 or greater and end row stroke type is not solid.
EndRowStroke-Overprint	boolean	no	If true, the rows in the second alternating rows group will overprint. Note: Valid when end row stroke count is 1 or greater.
EndRowStrokeTint	double	no	The tint (as a percentage) of the row borders in the second alternating strokes group. (Range: 0 to 100) Note: Valid when end row stroke count is 1 or greater.
EndRowStrokeType	string	no	The stroke type of rows in the second alternating strokes group.
EndRowStroke-Weight	double	no	The stroke weight of row borders in the second alternating row strokes group. Note: Valid when end row stroke count is 1 or greater.
FooterRowCount	int	no	The number of footer rows.

Name	Type	Req	Description
HeaderRowCount	int	no	The number of header rows.
KeepWithNextRow	boolean	no	If true, keep the row with the next row.
LeftBorderStroke-Color	string	no	The color, specified as a swatch (color, gradient, tint, or mixed ink), of the left border stroke.
LeftBorderStroke-GapColor	string	no	The gap color, specified as a swatch (color, gradient, tint, or mixed ink), of the left border stroke. Note: Valid only when left border stroke type is not solid.
LeftBorderStroke-GapOverprint	boolean	no	If true, the gap of the left border stroke will overprint. Note: Valid only when left border stroke type is not solid.
LeftBorderStroke-GapTint	double	no	The tint (as a percentage) of the gap color of the left border stroke. (Range: 0 to 100) Note: Valid only when left border stroke type is not solid.
LeftBorderStroke-Overprint	boolean	no	If true, the left border stroke will overprint.
LeftBorderStroke-Tint	double	no	The tint (as a percentage) of the left border stroke. (Range: 0 to 100)
LeftBorderStroke-Type	string	no	The stroke type of the left border.
LeftBorderStroke-Weight	double	no	The stroke weight of the left border stroke.
MaximumHeight	double	no	The maximum height of the cell.
MinimumHeight	double	no	The minimum height of the cell.
RightBorder-StrokeColor	string	no	The color, specified as a swatch (color, gradient, tint, or mixed ink), of the right border stroke.
RightBorder-StrokeGapColor	string	no	The gap color, specified as a swatch (color, gradient, tint, or mixed ink), of the right border stroke. Note: Valid only when right border stroke type is not solid.
RightBorder-StrokeGap-Overprint	boolean	no	If true, the gap color of the right border stroke will overprint. Note: Valid only when right border stroke type is not solid.
RightBorder-StrokeGapTint	double	no	The tint (as a percentage) of the gap color of the right border stroke. (Range: 0 to 100) Note: Valid only when right border stroke type is not solid.
RightBorder-StrokeOverprint	boolean	no	If true, the right border stroke will overprint.
RightBorder-StrokeTint	double	no	The tint (as a percentage) of the right border stroke. (Range: 0 to 100)
RightBorder-StrokeType	string	no	The stroke type of the right border.
RightBorder-StrokeWeight	double	no	The stroke weight of the right border stroke.
SingleColumnWidth	double	no	The width of a column.

Name	Type	Req	Description
SingleRowHeight	double	no	The height of a row.
SkipFirst-AlternatingFill-Columns	int	no	The number of columns on the left side of the table to skip before applying the column fill color. Note: Valid when alternating fills are defined for table columns.
SkipFirst-AlternatingFill-Rows	int	no	The number of body rows at the beginning of the table to skip before applying the row fill color. Note: Valid when alternating fills are defined for table rows.
SkipFirst-Alternating-StrokeColumns	int	no	The number of columns on the left of the table in which to skip border stroke formatting. Note: Valid when start column stroke count is 1 or greater and/or end column stroke count is 1 or greater.
SkipFirst-Alternating-StrokeRows	int	no	The number of body rows at the beginning of the table in which to skip border stroke formatting. Note: Valid when start row stroke count is 1 or greater and/or end row stroke count is 1 or greater.
SkipFirstHeader	boolean	no	If true, skips the first occurrence of header rows.
SkipLast-AlternatingFill-Columns	int	no	The number columns on the right side of the table in which to not apply the column fill color. Note: Valid when alternating fills are defined for table columns.
SkipLast-AlternatingFill-Rows	int	no	The number of body rows at the end of the table in which to not apply the row fill color. Note: Valid when alternating fills are defined for table rows.
SkipLast-Alternating-StrokeColumns	int	no	The number of columns on the right side of the table in which to skip border stroke formatting. Note: Valid when start column stroke count is 1 or greater and/or end column stroke count is 1 or greater.
SkipLast-Alternating-StrokeRows	int	no	The number of body rows at the end of the table in which to skip border stroke formatting. Note: Valid when start row stroke count is 1 or greater and/or end row stroke count is 1 or greater.
SkipLastFooter	boolean	no	If true, skips the last occurrence of footer rows.
SpaceAfter	double	no	The space below the table.
SpaceBefore	double	no	The space above the table.
StartColumnFill-Color	string	no	The fill color, specified as a swatch (color, gradient, tint, or mixed ink), of columns in the first alternating fills group. Note: Valid when alternating fills are defined for table columns.
StartColumnFill-Count	int	no	The number of columns in the first alternating fills group. Note: Valid when alternating fills are defined for table columns.

Name	Type	Req	Description
StartColumnFill-Overprint	boolean	no	If true, the columns in the first alternating fills group will overprint. Note: Valid when alternating fills are defined for table columns.
StartColumnFill-Tint	double	no	The tint (as a percentage) of the columns in the first alternating fills group. (Range: 0 to 100) Note: Valid when alternating fills are defined for table columns.
StartColumn-StrokeColor	string	no	The stroke color, specified as a swatch (color, gradient, tint, or mixed ink), of column borders in the first alternating column strokes group.
StartColumn-StrokeCount	int	no	The number of columns in the first alternating column strokes group.
StartColumn-StrokeGapColor	string	no	The stroke gap color, specified as a swatch (color, gradient, tint, or mixed ink), of column borders in the first alternating column strokes group. Note: Valid when start column stroke count is 1 or greater.
StartColumn-StrokeGap-Overprint	boolean	no	If true, the gap of the column borders in the first alternating column strokes group will overprint. Note: Valid when start column stroke count is 1 or greater.
StartColumn-StrokeGapTint	double	no	The tint (as a percentage) of the gap color of column borders in the first alternating column strokes group. (Range: 0 to 100) Note: Valid when start column stroke count is 1 or greater.
StartColumn-StrokeOverprint	boolean	no	If true, the column borders in the first alternating column strokes group will overprint. Note: Valid when start column stroke count is 1 or greater.
StartColumn-StrokeTint	double	no	The tint (as a percentage) of column borders in the first alternating column strokes group. (Range: 0 to 100) Note: Valid when start column stroke count is 1 or greater.
StartColumn-StrokeType	string	no	The stroke type of columns in the first alternating strokes group.
StartColumn-StrokeWeight	double	no	The stroke weight of column borders in the first alternating column strokes group. Note: Valid when start column stroke count is 1 or greater.
StartRow	StartParagraph_EnumValue	no	Can be Anywhere (Starts in the next available space), NextColumn (Starts at the top of the next column), NextFrame (Starts at the top of the next text frame in the thread), NextPage (Starts at the top of the next page), NextOddPage (Starts at the top of the next odd-numbered page), or NextEvenPage (Starts at the top of the next even-numbered page).
StartRowFillColor	string	no	The fill color, specified as a swatch (color, gradient, tint, or mixed ink), of rows in the first alternating fills group. Note: Valid when alternating fills are defined for table rows.

Name	Type	Req	Description
StartRowFillCount	int	no	The number of rows in the first alternating fills group. Note: Valid when alternating fills are defined for table rows.
StartRowFill-Overprint	boolean	no	If true, the rows in the first alternating fills group will overprint. Note: Valid when alternating fills are defined for table rows.
StartRowFillTint	double	no	The tint (as a percentage) of the rows in the first alternating fills group. (Range: 0 to 100) Note: Valid when alternating fills are defined for table rows.
StartRowStroke-Color	string	no	The color, specified as a swatch (color, gradient, tint, or mixed ink), of row borders in the first alternating row strokes group. Note: Valid when start row stroke count is 1 or greater.
StartRowStroke-Count	int	no	The number of rows in the first alternating row strokes group.
StartRowStroke-GapColor	string	no	The stroke gap color of row borders in the first alternating row strokes group, specified as a swatch (color, gradient, tint, or mixed ink). Note: Valid when start row stroke count is 1 or greater.
StartRowStroke-GapOverprint	boolean	no	If true, the gap color of the row border stroke in the first alternating row strokes group will overprint. Note: Valid when start row stroke count is 1 or greater.
StartRowStroke-GapTint	double	no	The tint (as a percentage) of the gap color of row borders in the first alternating rows group. (Range: 0 to 100) Note: Valid when start row stroke count is 1 or greater.
StartRowStroke-Overprint	boolean	no	If true, the row borders in the first alternating row strokes group will overprint. Note: Valid when start row stroke count is 1 or greater.
StartRowStroke-Tint	double	no	The tint (as a percentage) of the borders in the first alternating row strokes group. (Range: 0 to 100) Note: Valid when start row stroke count is 1 or greater.
StartRowStroke-Type	string	no	The stroke type of rows in the first alternating strokes group.
StartRowStroke-Weight	double	no	The stroke weight of row borders in the first alternating row strokes group. Note: Valid when start row stroke count is 1 or greater.

Name	Type	Req	Description
StrokeOrder	StrokeOrderTypes_EnumValue	no	The order in which to display row and column strokes at corners. Can be RowOnTop (Places row strokes in front of column strokes), ColumnOnTop (Places column strokes in front of row strokes), BestJoins (Places row strokes in front of column strokes when row and column strokes are different colors; joins striped strokes and connects crossing points), or InDesign2Compatibility (Places row strokes in front when row and column strokes are different colors; joins striped strokes only at points where strokes cross in a T-shape).
TableDirection	TableDirection_EnumValue	no	The direction of the the table. Can be LeftToRightDirection (Set left to right table direction), or RightToLeftDirection (Set right to left table direction).
TopBorderStroke-Color	string	no	The color, specified as a swatch (color, gradient, tint, or mixed ink), of the table's top border stroke.
TopBorderStroke-GapColor	string	no	The gap color, specified as a swatch (color, gradient, tint, or mixed ink), of the table's top border stroke. Note: Valid only when top border stroke type is not solid.
TopBorderStroke-GapOverprint	boolean	no	If true, the gap of the top border stroke will overprint. Note: Valid only when top border stroke type is not solid.
TopBorderStroke-GapTint	double	no	The tint (as a percentage) of the gap color of the table's top border stroke. (Range: 0 to 100) Note: Valid only when top border stroke type is not solid.
TopBorderStroke-Overprint	boolean	no	If true, the top border strokes will overprint.
TopBorderStroke-Tint	double	no	The tint (as a percentage) of the table's top border stroke. (Range: 0 to 100)
TopBorderStroke-Type	string	no	The stroke type of the top border.
TopBorderStroke-Weight	double	no	The stroke weight of the table's top border stroke.

A simple table (one row, three columns) would appear as follows (again, we've omitted the details of the `<Story>` element for clarity). Note that the `<Cell>` elements contain `<ParagraphStyle-Range>` and `<CharacterStyleRange>` elements that follow the same pattern as they do in the body of the `<Story>` element.

#### IDML Example 43. Table

```
<ParagraphStyleRange AppliedParagraphStyle="ParagraphStyle\NormalParagraphStyle">
  <CharacterStyleRange AppliedCharacterStyle=
    "CharacterStyle\[No character style]">
      <Table Self="uddie2" StoryOffset="ucbInsertionPoint0" HeaderRowCount="0"
        FooterRowCount="0" BodyRowCount="1" ColumnCount="3"
        AppliedTableStyle="TableStyle\[Basic Table]">
```

```

TableDirection="LeftToRightDirection">
  <Row Self="uddie2Row0" Name="0" SingleRowHeight="16.3203125"/>
  <Column Self="uddie2Column0" Name="0" SingleColumnWidth="180"/>
  <Column Self="uddie2Column1" Name="1" SingleColumnWidth="180"/>
  <Column Self="uddie2Column2" Name="2" SingleColumnWidth="180"/>
  <Cell Self="uddie2i0" Name="0:0" RowSpan="1" ColumnSpan="1"
    AppliedCellStyle="CellStyle\[None\]" AppliedCellStylePriority="0">
    <ParagraphStyleRange AppliedParagraphStyle=
      "ParagraphStyle\NormalParagraphStyle">
      <CharacterStyleRange AppliedCharacterStyle=
        "CharacterStyle\[No character style]">
        <Content>ABC</Content>
      </CharacterStyleRange>
    </ParagraphStyleRange>
  </Cell>
  <Cell Self="uddie2i1" Name="1:0" RowSpan="1" ColumnSpan="1"
    AppliedCellStyle="CellStyle\[None\]" AppliedCellStylePriority="0">
    <ParagraphStyleRange AppliedParagraphStyle=
      "ParagraphStyle\NormalParagraphStyle">
      <CharacterStyleRange AppliedCharacterStyle=
        "CharacterStyle\[No character style]">
        <Content>DEF</Content>
      </CharacterStyleRange>
    </ParagraphStyleRange>
  </Cell>
  <Cell Self="uddie2i2" Name="2:0" RowSpan="1" ColumnSpan="1"
    AppliedCellStyle="CellStyle\[None\]" AppliedCellStylePriority="0">
    <ParagraphStyleRange AppliedParagraphStyle=
      "ParagraphStyle\NormalParagraphStyle">
      <CharacterStyleRange AppliedCharacterStyle=
        "CharacterStyle\[No character style]">
        <Content>GHI</Content>
      </CharacterStyleRange>
    </ParagraphStyleRange>
  </Cell>
</Table>
</CharacterStyleRange>
</ParagraphStyleRange>

```

Figure 35. Simple Table

ABC	DEF	GHI
-----	-----	-----

## Cell

### Schema Example 46. Cell

```

Cell_Object = element Cell {
  attribute Self { xsd:string },
  attribute Name { xsd:string },
  attribute RowSpan { xsd:int }?,

```

```

        attribute ColumnSpan { xsd:int }?,
        attribute TopInset { xsd:double }?,
        attribute LeftInset { xsd:double }?,
        attribute BottomInset { xsd:double }?,
        attribute RightInset { xsd:double }?,
        attribute FillColor { xsd:string }?,
        attribute FillTint { xsd:double }?,
        attribute OverprintFill { xsd:boolean }?,
        attribute TopLeftDiagonalLine { xsd:boolean }?,
        attribute TopRightDiagonalLine { xsd:boolean }?,
        attribute DiagonalLineInFront { xsd:boolean }?,
        attribute DiagonalLineStrokeWeight { xsd:double }?,
        attribute DiagonalLineStrokeType { xsd:string }?,
        attribute DiagonalLineStrokeColor { xsd:string }?,
        attribute DiagonalLineStrokeTint { xsd:double }?,
        attribute DiagonalLineStrokeOverprint { xsd:boolean }?,
        attribute DiagonalLineStrokeGapColor { xsd:string }?,
        attribute DiagonalLineStrokeGapTint { xsd:double }?,
        attribute DiagonalLineStrokeGapOverprint { xsd:boolean }?,
        attribute ClipContentToCell { xsd:boolean }?,
        attribute FirstBaselineOffset { FirstBaseline_EnumValue }?,
        attribute VerticalJustification { VerticalJustification_EnumValue }?,
        attribute ParagraphSpacingLimit { xsd:double }?,
        attribute MinimumFirstBaselineOffset { xsd:double {minInclusive="0" max-
Inclusive="8640"} }?,
        attribute RotationAngle { xsd:double }?,
        attribute LeftEdgeStrokeWeight { xsd:double }?,
        attribute LeftEdgeStrokeType { xsd:string }?,
        attribute LeftEdgeStrokeColor { xsd:string }?,
        attribute LeftEdgeStrokeTint { xsd:double }?,
        attribute LeftEdgeStrokeOverprint { xsd:boolean }?,
        attribute LeftEdgeStrokeGapColor { xsd:string }?,
        attribute LeftEdgeStrokeGapTint { xsd:double }?,
        attribute LeftEdgeStrokeGapOverprint { xsd:boolean }?,
        attribute TopEdgeStrokeWeight { xsd:double }?,
        attribute TopEdgeStrokeType { xsd:string }?,
        attribute TopEdgeStrokeColor { xsd:string }?,
        attribute TopEdgeStrokeTint { xsd:double }?,
        attribute TopEdgeStrokeOverprint { xsd:boolean }?,
        attribute TopEdgeStrokeGapColor { xsd:string }?,
        attribute TopEdgeStrokeGapTint { xsd:double }?,
        attribute TopEdgeStrokeGapOverprint { xsd:boolean }?,
        attribute RightEdgeStrokeWeight { xsd:double }?,
        attribute RightEdgeStrokeType { xsd:string }?,
        attribute RightEdgeStrokeColor { xsd:string }?,
        attribute RightEdgeStrokeTint { xsd:double }?,
        attribute RightEdgeStrokeOverprint { xsd:boolean }?,
        attribute RightEdgeStrokeGapColor { xsd:string }?,
        attribute RightEdgeStrokeGapTint { xsd:double }?,
        attribute RightEdgeStrokeGapOverprint { xsd:boolean }?,
        attribute BottomEdgeStrokeWeight { xsd:double }?,
        attribute BottomEdgeStrokeType { xsd:string }?,
        attribute BottomEdgeStrokeColor { xsd:string }?,
        attribute BottomEdgeStrokeTint { xsd:double }?
    
```

```

        attribute BottomEdgeStrokeOverprint { xsd:boolean }?,
        attribute BottomEdgeStrokeGapColor { xsd:string }?,
        attribute BottomEdgeStrokeGapTint { xsd:double }?,
        attribute BottomEdgeStrokeGapOverprint { xsd:boolean }?,
        attribute InnerRowStrokeWeight { xsd:double }?,
        attribute InnerRowStrokeType { xsd:string }?,
        attribute InnerRowStrokeColor { xsd:string }?,
        attribute InnerRowStrokeTint { xsd:double }?,
        attribute InnerRowStrokeOverprint { xsd:boolean }?,
        attribute InnerRowStrokeGapColor { xsd:string }?,
        attribute InnerRowStrokeGapTint { xsd:double }?,
        attribute InnerRowStrokeGapOverprint { xsd:boolean }?,
        attribute InnerColumnStrokeWeight { xsd:double }?,
        attribute InnerColumnStrokeType { xsd:string }?,
        attribute InnerColumnStrokeColor { xsd:string }?,
        attribute InnerColumnStrokeTint { xsd:double }?,
        attribute InnerColumnStrokeOverprint { xsd:boolean }?,
        attribute InnerColumnStrokeGapColor { xsd:string }?,
        attribute InnerColumnStrokeGapTint { xsd:double }?,
        attribute InnerColumnStrokeGapOverprint { xsd:boolean }?,
        attribute TopEdgeStrokePriority { xsd:int }?,
        attribute LeftEdgeStrokePriority { xsd:int }?,
        attribute BottomEdgeStrokePriority { xsd:int }?,
        attribute RightEdgeStrokePriority { xsd:int }?,
        attribute AppliedCellStyle { xsd:string }?,
        attribute WritingDirection { xsd:boolean }?,
        attribute AppliedCellStylePriority { xsd:int }?,
        element Properties {
            element AllCellGradientAttrList { list_type, element ListItem {
                (double_type, xsd:double ) |
                (list_type,
                element ListItem { unit_type, xsd:double },
                element ListItem { unit_type, xsd:double })
            }*
        }?&
        element Label { list_type, element ListItem {
            list_type,
            element ListItem { string_type, xsd:string },
            element ListItem { string_type, xsd:string }
        }*
    }?
}
?
'
(
    TextVariableInstance_Object*&
    Table_Object*&
    ParagraphStyleRange_Object*&
    CharacterStyleRange_Object*&
    Change_Object*&
    Note_Object*&
    TextFrame_Object*&
    Oval_Object*&
    Rectangle_Object*&

```

```

GraphicLine_Object*&
Polygon_Object*&
Group_Object*&
EPSText_Object*&
FormField_Object*&
Button_Object*&
HiddenText_Object*
)
}

```

**Table 54. Cell Properties Represented as Attributes**

Name	Type	Req	Description
AppliedCellStyle	string	no	The cell style applied to the cell.
AppliedCellStyle- Priority	int	no	
BottomEdgeStroke- Color	string	no	The swatch (color, gradient, tint, or mixed ink) applied to the bottom edge border stroke.
BottomEdgeStroke- GapColor	string	no	The swatch (color, gradient, tint, or mixed ink) applied to the gap of the bottom edge border stroke. Note: Not valid when bottom edge stroke type is solid.
BottomEdgeStroke- GapOverprint	boolean	no	If true, the gap color of the bottom edge border stroke will overprint. Note: Not valid when bottom edge stroke type is solid.
BottomEdgeStroke- GapTint	double	no	The tint (as a percentage) of the bottom edge border stroke gap color. (Range: 0 to 100) Note: Not valid when bottom edge stroke type is solid.
BottomEdgeStroke- Overprint	boolean	no	If true, the bottom edge border stroke will overprint.
BottomEdgeStroke- Priority	int	no	The priority of a stroke determines the order in which it will be drawn, relative to the other strokes on the cell. Higher values equal higher priority.
BottomEdgeStroke- Tint	double	no	The tint (as a percentage) of the bottom edge border stroke.
BottomEdgeStroke- Type	string	no	The stroke type of the bottom edge.
BottomEdgeStroke- Weight	double	no	The stroke weight of the bottom edge border stroke.
BottomInset	double	no	The bottom inset of the cell.
ClipContentToCell	boolean	no	If true, clips the cell's content to width and height of the cell.
ColumnSpan	int	no	The number of columns that the cell spans.
DiagonalLineIn- Front	boolean	no	If true, draws the diagonal line in front of cell contents.
DiagonalLine- StrokeColor	string	no	The diagonal line color, specified as a swatch.

Name	Type	Req	Description
DiagonalLine-StrokeGapColor	string	no	The swatch (color, gradient, tint, or mixed ink) applied to the gap of the diagonal line stroke. Note: Not valid when diagonal line stroke type is solid.
DiagonalLine-StrokeGap-Overprint	boolean	no	If true, the stroke gap of the diagonal line will overprint. Note: Not valid when diagonal line stroke type is solid.
DiagonalLine-StrokeGapTint	double	no	The tint (as a percentage) of the diagonal line stroke gap color. Note: Not valid when diagonal line stroke type is solid.
DiagonalLine-StrokeOverprint	boolean	no	If true, the diagonal line stroke will overprint.
DiagonalLine-StrokeTint	double	no	The diagonal line tint (as a percentage). (Range: 0 to 100)
DiagonalLine-StrokeType	string	no	The stroke type of the diagonal line(s).
DiagonalLine-StrokeWeight	double	no	The diagonal line stroke weight.
FillColor	string	no	The swatch (color, gradient, tint, or mixed ink) applied to the fill of the cell.
FillTint	double	no	The tint (as a percentage) of the fill of the cell.
FirstBaseline-Offset	FirstBaseline_EnumValue	no	The distance between the baseline of the text and the top inset of the cell. Can be Ascent-Offset (The tallest character in the font falls below the top inset of the object), CapHeight (The tops of upper case letters touch the top inset of the object), LeadingOffset (The text leading value defines the distance between the baseline of the text and the top inset of the object), EmboxHeight (The text em box height is the distance between the baseline of the text and the top inset of the object), xHeight (The tops of lower case letters touch the top inset of the object), or FixedHeight (Uses the value specified for minimum first baseline offset as the distance between the baseline of the text and the top inset of the object).
InnerColumn-StrokeColor	string	no	The color, specified as a swatch, of the inner column border stroke.
InnerColumn-StrokeGapColor	string	no	The swatch (color, gradient, tint, or mixed ink) applied to the gap of the inner column border stroke. Note: Not valid when inner column stroke type is solid.
InnerColumn-StrokeGap-Overprint	boolean	no	If true, the gap color of the inner column border stroke will overprint. Note: Not valid when inner column stroke type is solid.

Name	Type	Req	Description
InnerColumn-StrokeGapTint	double	no	The tint (as a percentage) of the inner column border stroke gap color. (Range: 0 to 100) Note: Not valid when inner column stroke type is solid.
InnerColumn-StrokeOverprint	boolean	no	If true, the inner column border stroke will overprint.
InnerColumn-StrokeTint	double	no	The tint (as a percentage) of the inner column border stroke. (Range: 0 to 100)
InnerColumn-StrokeType	string	no	The stroke type of the inner column.
InnerColumn-StrokeWeight	double	no	The stroke weight of the inner column border stroke.
InnerRowStroke-Color	string	no	The color, specified as a swatch, of the inner row border stroke.
InnerRowStroke-GapColor	string	no	The swatch (color, gradient, tint, or mixed ink) applied to the gap of the inner row border stroke. Note: Not valid when inner row stroke type is solid.
InnerRowStroke-GapOverprint	boolean	no	If true, the gap color of the inner row border stroke will overprint. Note: Not valid when inner row stroke type is solid.
InnerRowStroke-GapTint	double	no	The tint (as a percentage) of the inner row border stroke. (Range: 0 to 100) Note: Not valid when inner row stroke type is solid.
InnerRowStroke-Overprint	boolean	no	If true, the inner row border stroke will overprint.
InnerRowStroke-Tint	double	no	The tint (as a percentage) of the inner row border stroke. (Range: 0 to 100)
InnerRowStroke-Type	string	no	The stroke type of the inner row.
InnerRowStroke-Weight	double	no	The stroke weight of the inner row border strokes.
LeftEdgeStroke-Color	string	no	The swatch (color, gradient, tint, or mixed ink) applied to the left edge border stroke.
LeftEdgeStroke-GapColor	string	no	The swatch (color, gradient, tint, or mixed ink) applied to the gap of the left edge border stroke. Note: Not valid when left edge stroke type is solid.
LeftEdgeStroke-GapOverprint	boolean	no	If true, the gap color of the left edge border stroke will overprint. Note: Not valid when left edge stroke type is solid.
LeftEdgeStroke-GapTint	double	no	The tint (as a percentage) of the left edge border stroke gap color. (Range: 0 to 100) Note: Not valid when left edge stroke type is solid.
LeftEdgeStroke-Overprint	boolean	no	If true, the left edge border stroke will overprint.

Name	Type	Req	Description
LeftEdgeStroke-Priority	int	no	The priority of a stroke determines the order in which it will be drawn, relative to the other strokes on the cell. Higher values equal higher priority.
LeftEdgeStroke-Tint	double	no	The tint (as a percentage) of the left edge border stroke. (Range: 0 to 100)
LeftEdgeStroke-Type	string	no	The stroke type of the left edge.
LeftEdgeStroke-Weight	double	no	The stroke weight of the left edge border stroke.
LeftInset	double	no	The left inset of the cell.
MinimumFirst-BaselineOffset	double	no	The space between the baseline of the text and the top inset of the frame or cell.
OverprintFill	boolean	no	If true, the fill of the cell will overprint.
ParagraphSpacing-Limit	double	no	The maximum space that can be added between paragraphs in a cell. Note: Valid only when vertical justification is justified.
RightEdgeStroke-Color	string	no	The color, specified as a swatch, of the right edge border stroke.
RightEdgeStroke-GapColor	string	no	The swatch (color, gradient, tint, or mixed ink) applied to the gap of the right edge border stroke. Note: Not valid when right edge stroke type is solid.
RightEdgeStroke-GapOverprint	boolean	no	If true, the gap color of the right edge border stroke will overprint. Note: Not valid when right edge stroke type is solid.
RightEdgeStroke-GapTint	double	no	The tint (as a percentage) of the right edge border stroke gap color. (Range: 0 to 100) Note: Not valid when right edge stroke type is solid.
RightEdgeStroke-Overprint	boolean	no	If true, the right edge border stroke will overprint.
RightEdgeStroke-Priority	int	no	The priority of a stroke determines the order in which it will be drawn, relative to the other strokes on the cell. Higher values equal higher priority.
RightEdgeStroke-Tint	double	no	The tint (as a percentage) of the right edge border stroke. (Range: 0 to 100)
RightEdgeStroke-Type	string	no	The stroke type of the right edge.
RightEdgeStroke-Weight	double	no	The stroke weight of the right edge border stroke.
RightInset	double	no	The right inset of the cell.
RotationAngle	double	no	The rotation angle (in degrees) of the cell, specified as one of the following values: 0, 90, 180, or 270.
RowSpan	int	no	The number of rows that the cell spans.

Name	Type	Req	Description
TopEdgeStroke-Color	string	no	The swatch (color, gradient, tint, or mixed ink) applied to the top edge border stroke.
TopEdgeStrokeGap-Color	string	no	The swatch (color, gradient, tint, or mixed ink) applied to the gap of the top edge border stroke. Note: Not valid when top edge stroke type is solid.
TopEdgeStrokeGap-Overprint	boolean	no	If true, the gap color of the top edge border stroke will overprint. Note: Not valid when top edge stroke type is solid.
TopEdgeStrokeGap-Tint	double	no	The tint (as a percentage) of the top edge border stroke gap color. (Range: 0 to 100) Note: Not valid when top edge stroke type is solid.
TopEdgeStroke-Overprint	boolean	no	If true, the top edge border stroke will overprint.
TopEdgeStroke-Priority	int	no	The priority of a stroke determines the order in which it will be drawn, relative to the other strokes on the cell. Higher values equal higher priority.
TopEdgeStrokeTint	double	no	The tint (as a percentage) of the top edge border stroke. (Range: 0 to 100)
TopEdgeStrokeType	string	no	The stroke type of the top edge.
TopEdgeStroke-Weight	double	no	The stroke weight of the top edge border stroke.
TopInset	double	no	The top inset of the cell.
TopLeftDiagonal-Line	boolean	no	If true, draws a diagonal line starting from the top left.
TopRightDiagonal-Line	boolean	no	If true, draws a diagonal line starting from the top right.
Vertical-Justification	Vertical-Justification_EnumValue	no	The vertical alignment of cell. Can be Top-Align (Text is aligned at the top of the object), CenterAlign (Text is center aligned vertically in the object), BottomAlign (Text is aligned at the bottom of the object), or JustifyAlign (Lines of text are evenly distributed vertically between the top and bottom of the object).
WritingDirection	boolean	no	The direction of the text in the cell.

**Table 55. Cell Properties Represented as Elements**

AllCellGradient-AttrList	List Item or double	no	
--------------------------	---------------------	----	--

**Column**

```
Column_Object = element Column {
    attribute Self { xsd:string },
    attribute Name { xsd:string },
    attribute TopInset { xsd:double }?,
```

```

        attribute LeftInset { xsd:double }?,
        attribute BottomInset { xsd:double }?,
        attribute RightInset { xsd:double }?,
        attribute FillColor { xsd:string }?,
        attribute FillTint { xsd:double }?,
        attribute OverprintFill { xsd:boolean }?,
        attribute TopLeftDiagonalLine { xsd:boolean }?,
        attribute TopRightDiagonalLine { xsd:boolean }?,
        attribute DiagonalLineInFront { xsd:boolean }?,
        attribute DiagonalLineStrokeWeight { xsd:double }?,
        attribute DiagonalLineStrokeType { xsd:string }?,
        attribute DiagonalLineStrokeColor { xsd:string }?,
        attribute DiagonalLineStrokeTint { xsd:double }?,
        attribute DiagonalLineStrokeOverprint { xsd:boolean }?,
        attribute DiagonalLineStrokeGapColor { xsd:string }?,
        attribute DiagonalLineStrokeGapTint { xsd:double }?,
        attribute DiagonalLineStrokeGapOverprint { xsd:boolean }?,
        attribute ClipContentToCell { xsd:boolean }?,
        attribute FirstBaselineOffset { FirstBaseline_EnumValue }?,
        attribute VerticalJustification { VerticalJustification_EnumValue }?,
        attribute ParagraphSpacingLimit { xsd:double }?,
        attribute MinimumFirstBaselineOffset { xsd:double {minInclusive="0"
maxInclusive="8640"} }?,
        attribute RotationAngle { xsd:double }?,
        attribute WritingDirection { xsd:boolean }?,
        attribute SingleColumnWidth { xsd:double }?
    }
}

```

**Table 56. Column Properties Represented as Attributes**

Name	Type	Req	Description
BottomInset	double	no	The bottom inset of the cell.
ClipContentToCell	boolean	no	If true, clips the cell's content to width and height of the cell.
DiagonalLineIn-Front	boolean	no	If true, draws the diagonal line in front of cell contents.
DiagonalLine-StrokeColor	string	no	The diagonal line color, specified as a swatch.
DiagonalLine-StrokeGapColor	string	no	The swatch (color, gradient, tint, or mixed ink) applied to the gap of the diagonal line stroke. Note: Not valid when diagonal line stroke type is solid.
DiagonalLine-StrokeGap-Overprint	boolean	no	If true, the stroke gap of the diagonal line will overprint. Note: Not valid when diagonal line stroke type is solid.
DiagonalLine-StrokeGapTint	double	no	The tint (as a percentage) of the diagonal line stroke gap color. Note: Not valid when diagonal line stroke type is solid.
DiagonalLine-StrokeOverprint	boolean	no	If true, the diagonal line stroke will overprint.

Name	Type	Req	Description
DiagonalLine-StrokeTint	double	no	The diagonal line tint (as a percentage). (Range: 0 to 100)
DiagonalLine-StrokeType	string	no	The stroke type of the diagonal line(s).
DiagonalLine-StrokeWeight	double	no	The diagonal line stroke weight.
FillColor	string	no	The swatch (color, gradient, tint, or mixed ink) applied to the fill of the Column.
FillTint	double	no	The tint (as a percentage) of the fill of the Column.
FirstBaseline-Offset	FirstBaseline_EnumValue	no	The distance between the baseline of the text and the top inset of the cell. Can be Ascent-Offset (The tallest character in the font falls below the top inset of the object), CapHeight (The tops of upper case letters touch the top inset of the object), LeadingOffset (The text leading value defines the distance between the baseline of the text and the top inset of the object), EmboxHeight (The text em box height is the distance between the baseline of the text and the top inset of the object), xHeight (The tops of lower case letters touch the top inset of the object), or FixedHeight (Uses the value specified for minimum first baseline offset as the distance between the baseline of the text and the top inset of the object).
LeftInset	double	no	The left inset of the cell.
MinimumFirst-BaselineOffset	double	no	The space between the baseline of the text and the top inset of the frame or cell.
Name	string	yes	The name of the column.
OverprintFill	boolean	no	If true, the fill of the Column will overprint.
ParagraphSpacing-Limit	double	no	The maximum space that can be added between paragraphs in a cell. Note: Valid only when vertical justification is justified.
RightInset	double	no	The right inset of the cell.
RotationAngle	double	no	The rotation angle (in degrees) of the cell, specified as one of the following values: 0, 90, 180, or 270.
SingleColumnWidth	double	no	The width of a single column.
TopInset	double	no	The top inset of the cell.
TopLeftDiagonal-Line	boolean	no	If true, draws a diagonal line starting from the top left.
TopRightDiagonal-Line	boolean	no	If true, draws a diagonal line starting from the top right.

Name	Type	Req	Description
Vertical-Justification	Vertical-Justification_EnumValue	no	The vertical alignment of cell. Can be Top-Align (Text is aligned at the top of the object), CenterAlign (Text is center aligned vertically in the object), BottomAlign (Text is aligned at the bottom of the object), or JustifyAlign (Lines of text are evenly distributed vertically between the top and bottom of the object).
WritingDirection	boolean	no	The direction of the text in the cell.

## Row

```
Row_Object = element Row {
    attribute Self { xsd:string },
    attribute Name { xsd:string },
    attribute TopInset { xsd:double }?,
    attribute LeftInset { xsd:double }?,
    attribute BottomInset { xsd:double }?,
    attribute RightInset { xsd:double }?,
    attribute FillColor { xsd:string }?,
    attribute FillTint { xsd:double }?,
    attribute OverprintFill { xsd:boolean }?,
    attribute TopLeftDiagonalLine { xsd:boolean }?,
    attribute TopRightDiagonalLine { xsd:boolean }?,
    attribute DiagonalLineInFront { xsd:boolean }?,
    attribute DiagonalLineStrokeWeight { xsd:double }?,
    attribute DiagonalLineStrokeType { xsd:string }?,
    attribute DiagonalLineStrokeColor { xsd:string }?,
    attribute DiagonalLineStrokeTint { xsd:double }?,
    attribute DiagonalLineStrokeOverprint { xsd:boolean }?,
    attribute DiagonalLineStrokeGapColor { xsd:string }?,
    attribute DiagonalLineStrokeGapTint { xsd:double }?,
    attribute DiagonalLineStrokeGapOverprint { xsd:boolean }?,
    attribute ClipContentToCell { xsd:boolean }?,
    attribute FirstBaselineOffset { FirstBaseline_EnumValue }?,
    attribute VerticalJustification { VerticalJustification_EnumValue }?,
    attribute ParagraphSpacingLimit { xsd:double }?,
    attribute MinimumFirstBaselineOffset { xsd:double {minInclusive="0"
maxInclusive="8640"} }?,
    attribute RotationAngle { xsd:double }?,
    attribute MinimumHeight { xsd:double }?,
    attribute MaximumHeight { xsd:double }?,
    attribute KeepWithNextRow { xsd:boolean }?,
    attribute StartRow { StartParagraph_EnumValue }?,
    attribute AutoGrow { xsd:boolean }?,
    attribute WritingDirection { xsd:boolean }?,
    attribute SingleRowHeight { xsd:double }?
}
```

**Table 57. Row Properties Represented as Attributes**

Name	Type	Req	Description
AutoGrow	boolean	no	If true, the height of the cell or the cells in the Row can increase or decrease automatically to fit cell content. Note: Allows cells to grow or shrink to the maximum or minimum height, if specified.
BottomInset	double	no	The bottom inset of the cell.
ClipContentToCell	boolean	no	If true, clips the cell's content to width and height of the cell.
DiagonalLineIn-Front	boolean	no	If true, draws the diagonal line in front of cell contents.
DiagonalLine-StrokeColor	string	no	The diagonal line color, specified as a swatch.
DiagonalLine-StrokeGapColor	string	no	The swatch (color, gradient, tint, or mixed ink) applied to the gap of the diagonal line stroke. Note: Not valid when diagonal line stroke type is solid.
DiagonalLine-StrokeGap-Overprint	boolean	no	If true, the stroke gap of the diagonal line will overprint. Note: Not valid when diagonal line stroke type is solid.
DiagonalLine-StrokeGapTint	double	no	The tint (as a percentage) of the diagonal line stroke gap color. Note: Not valid when diagonal line stroke type is solid.
DiagonalLine-StrokeOverprint	boolean	no	If true, the diagonal line stroke will overprint.
DiagonalLine-StrokeTint	double	no	The diagonal line tint (as a percentage). (Range: 0 to 100)
DiagonalLine-StrokeType	string	no	The stroke type of the diagonal line(s).
DiagonalLine-StrokeWeight	double	no	The diagonal line stroke weight.
FillColor	string	no	The swatch (color, gradient, tint, or mixed ink) applied to the fill of the Row.
FillTint	double	no	The tint (as a percentage) of the fill of the Row.

Name	Type	Req	Description
FirstBaseline-Offset	FirstBaseline_EnumValue	no	The distance between the baseline of the text and the top inset of the cell. Can be Ascent-Offset (The tallest character in the font falls below the top inset of the object), CapHeight (The tops of upper case letters touch the top inset of the object), LeadingOffset (The text leading value defines the distance between the baseline of the text and the top inset of the object), EmboxHeight (The text em box height is the distance between the baseline of the text and the top inset of the object), or FixedHeight (Uses the value specified for minimum first baseline offset as the distance between the baseline of the text and the top inset of the object).
KeepWithNextRow	boolean	no	If true, keeps the row with the next row when the table is split across text frames or pages.
LeftInset	double	no	The left inset of the cell.
MaximumHeight	double	no	The maximum height to which the row or the column's rows may grow. Note: The maximum height cannot be exceeded even when auto grow is set to true. Also, the maximum height can affect redistribution. For information, see redistribute.
MinimumFirst-BaselineOffset	double	no	The space between the baseline of the text and the top inset of the frame or cell.
MinimumHeight	double	no	The minimum height that the cell or the Row's cells are allowed to be. Note: When auto grow is true, cells can automatically grow larger than this amount when content is added. Also, the minimum height can affect redistribution. For information, see redistribute.
Name	string	yes	The name of the row.
OverprintFill	boolean	no	If true, the fill of the Row will overprint.
ParagraphSpacing-Limit	double	no	The maximum space that can be added between paragraphs in a cell. Note: Valid only when vertical justification is justified.
RightInset	double	no	The right inset of the cell.
RotationAngle	double	no	The rotation angle (in degrees) of the cell, specified as one of the following values: 0, 90, 180, or 270.
SingleRowHeight	double	no	The maximum height of a single row.

Name	Type	Req	Description
StartRow	StartParagraph_EnumValue	no	Indicates where to start the row. Can be Anywhere (Starts in the next available space), NextColumn (Starts at the top of the next column), NextFrame (Starts at the top of the next text frame in the thread), NextPage (Starts at the top of the next page), NextOddPage (Starts at the top of the next odd-numbered page), or NextEvenPage (Starts at the top of the next even-numbered page).
TopInset	double	no	The top inset of the cell.
TopLeftDiagonal-Line	boolean	no	If true, draws a diagonal line starting from the top left.
TopRightDiagonal-Line	boolean	no	If true, draws a diagonal line starting from the top right.
Vertical-Justification	Vertical-Justification_EnumValue	no	The vertical alignment of cell. Can be Top-Align (Text is aligned at the top of the object), CenterAlign (Text is center aligned vertically in the object), BottomAlign (Text is aligned at the bottom of the object), or JustifyAlign (Lines of text are evenly distributed vertically between the top and bottom of the object).
WritingDirection	boolean	no	The direction of the text in the cell.

### Footnotes

Footnotes are another example of an element that can appear in a <CharacterStyleRange> element.

#### Schema Example 47. Footnote

```
Footnote_Object = element Footnote {
    element Properties {
        element Label { element KeyValuePair { KeyValuePair_TypeDef }* }
    }?
}
?
'
(
    Table_Object*&
    TextVariableInstance_Object*&
    ParagraphStyleRange_Object*&
    CharacterStyleRange_Object*&
    TextFrame_Object*&
    Oval_Object*&
    Rectangle_Object*&
    GraphicLine_Object*&
    Polygon_Object*&
    Group_Object*&
    EPSText_Object*&
    FormField_Object*&
    Button_Object*&
    HiddenText_Object*
```

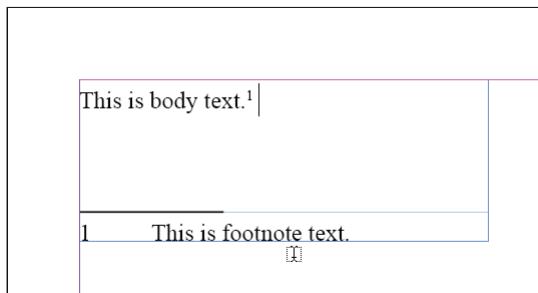
```
)  
}
```

The following example shows a very simple example footnote (again, we've omitted the `<Story>` element for clarity):

#### IDML Example 44. Footnote

```
<ParagraphStyleRange AppliedParagraphStyle="ParagraphStyle\kNormalParagraphStyle">
    <CharacterStyleRange AppliedCharacterStyle=
        "CharacterStyle\k[No character style]">
        <Content>This is body text.</Content>
    </CharacterStyleRange>
    <CharacterStyleRange AppliedCharacterStyle=
        "CharacterStyle\k[No character style]" Position="Superscript">
        <Footnote>
            <ParagraphStyleRange AppliedParagraphStyle=
                "ParagraphStyle\kNormalParagraphStyle">
                <CharacterStyleRange AppliedCharacterStyle=
                    "CharacterStyle\k[No character style]">
                    <Content><?ACE 4?> This is footnote text.</Content>
                </CharacterStyleRange>
            </ParagraphStyleRange>
        </Footnote>
    </CharacterStyleRange>
</ParagraphStyleRange>
```

*Figure 36. Footnote*



#### Notes

InDesign stories can contain non-printing notes. Notes in IDML are supported just like the other inline objects—they appear as child elements of a `<CharacterStyleRange>` object:

#### Schema Example 48.

```
Note_Object = element Note {
    attribute Collapsed { xsd:boolean }?,
    attribute CreationDate { xsd:dateTime }?,
    attribute ModificationDate { xsd:dateTime }?,
    attribute UserName { xsd:string }?,
    attribute AppliedDocumentUser { xsd:string }?,
    element Properties {
        element Label { element KeyValuePair { KeyValuePair_TypeDef }* }?
    }?
```

```

}
?

'
(
    Footnote_Object*&
    TextVariableInstance_Object*&
    TextFrame_Object*&
    Oval_Object*&
    Rectangle_Object*&
    GraphicLine_Object*&
    Polygon_Object*&
    Group_Object*&
    EPSText_Object*&
    FormField_Object*&
    Button_Object*&
    Table_Object*&
    ParagraphStyleRange_Object*&
    CharacterStyleRange_Object*&
    HiddenText_Object*
)
}
}

```

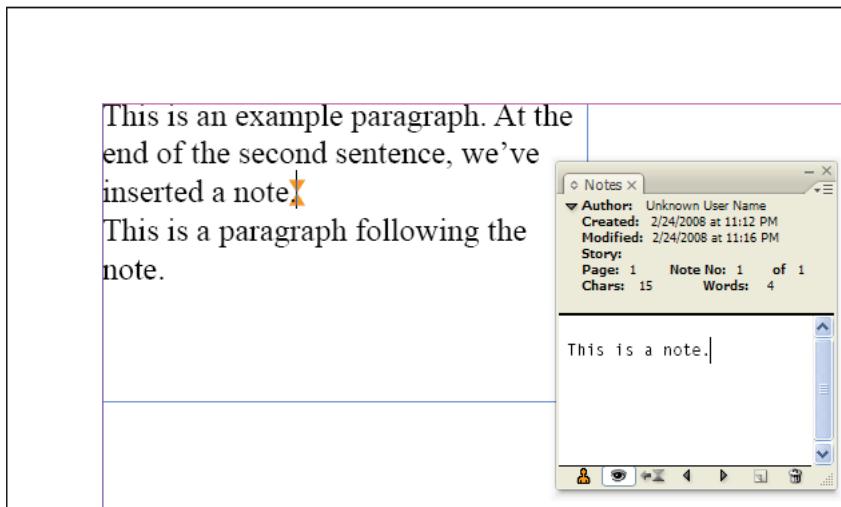
**IDML Example 45. Note**

```

<ParagraphStyleRange AppliedParagraphStyle="ParagraphStyle\kNormalParagraphStyle">
    <CharacterStyleRange AppliedCharacterStyle=
        "CharacterStyle\k[No character style]">
        <Content>This is an example paragraph. At the end of the second sentence, we've
        inserted a note.</Content>
        <Note Collapsed="false" CreationDate="2008-02-24T23:12:55"
            ModificationDate="2008-02-24T23:16:44" UserName="Unknown User Name"
            AppliedDocumentUser="dDocumentUser1">
            <ParagraphStyleRange AppliedParagraphStyle=
                "ParagraphStyle\k[No paragraph style]">
                <CharacterStyleRange AppliedCharacterStyle=
                    "CharacterStyle\k[No character style]">
                    <Content>This is a note.</Content>
                </CharacterStyleRange>
            </ParagraphStyleRange>
        </Note>
        <br/>
        <Content>This is a paragraph following the note.</Content>
    </CharacterStyleRange>
</ParagraphStyleRange>

```

Figure 37. Note



### Anchored Frames

InDesign documents often feature page items that have been embedded in text. These frames move with the text as the composition and layout of the text changes. These embedded frames are referred to as *anchored frames*. Anchored frames are also sometimes called *inline frames*—in IDML, an inline frame is a special case of an anchored frame.

The content of the inline text frame itself does not appear in the `<Story>` containing the frame, as shown in the following example. Instead, the `ParentStory` attribute inside the anchored `<TextFrame>` element includes a reference to another `<Story>` element in the same IDML package (in this example inside the, `Story_u109.xml` file). The `<Story>` element in that file contains the text elements that appear in the inline text frame.

The position of the anchored frame on the page is determined by the composition of the text surrounding the frame and preceding it in its parent `<Story>` element, and by the contents of the `<AnchoredObjectSetting>` element associated with the inline frame. The frame may “float” on the page, or may be positioned at points on the page determined by the contents of its `<AnchoredObjectSetting>`. You can change the shape and size of an anchored frame (in its inner coordinates) using its `<PathGeometry>` element, but you cannot transform the frame to a specific location in spread coordinates using the `ItemTransform` attribute of the frame (as you can when the frame is an independent page item).

For more on positioning anchored objects in text, refer to the InDesign online help.

#### Schema Example 49. AnchoredObjectSettings

```
AnchoredObjectSetting_Object = element AnchoredObjectSetting {
    attribute Self { xsd:string },
    attribute AnchoredPosition { AnchorPosition_EnumValue }?,
    attribute SpineRelative { xsd:boolean }?,
    attribute LockPosition { xsd:boolean }?,
    attribute PinPosition { xsd:boolean }?,
    attribute AnchorPoint { AnchorPoint_EnumValue }?,
    attribute HorizontalAlignment { HorizontalAlignment_EnumValue }?,
    attribute HorizontalReferencePoint { AnchoredRelativeTo_EnumValue }?,
}
```

```

        attribute VerticalAlignment { VerticalAlignment_EnumValue }?,
        attribute VerticalReferencePoint { VerticallyRelativeTo_EnumValue }?,
        attribute AnchorXoffset { xsd:double }?,
        attribute AnchorYoffset { xsd:double }?,
        attribute AnchorSpaceAbove { xsd:double }?
    }
}

```

**Table 58. AnchoredObjectSetting Properties Represented as Attributes**

Name	Type	Req	Description
AnchorPoint	AnchorPoint_EnumValue	no	The point in the anchored object to position. Can be TopLeftAnchor, TopCenterAnchor, TopRightAnchor, LeftCenterAnchor, CenterAnchor, RightCenterAnchor, BottomLeftAnchor, BottomCenterAnchor, or BottomRightAnchor.
AnchorSpaceAbove	double	no	The space above an anchored object. Valid only when AnchoredPosition is AboveLine.
AnchorXoffset	double	no	The horizontal (x) offset of the anchored object.
AnchorYoffset	double	no	The vertical (y) offset of the anchored object.
AnchoredPosition	AnchorPosition_EnumValue	no	The position of the anchored object relative to the anchor. Can be InlinePosition, AboveLine, or Anchored.
Horizontal-Alignment	Horizontal-Alignment_EnumValue	no	When AnchoredPosition is AboveLine, the position of the anchored object is relative to the text area. Can be RightAlign, LeftAlign, CenterAlign, or TextAlign. Not valid when anchored position is InlinePosition. Can be
Horizontal-ReferencePoint	AnchoredRelative-To_EnumValue	no	The horizontal reference point on the page. Valid only when AnchoredPosition is AnchoredPosition.
LockPosition	boolean	no	If true, prevents manual positioning of the anchored object.
PinPosition	boolean	no	If true, pins the position of the anchored object within the text frame top and bottom.
SpineRelative	boolean	no	If true, the position of the anchored object is relative to the binding spine of the page or spread.
VerticalAlignment	Vertical-Alignment_EnumValue	no	The vertical alignment of the anchored object reference point with the vertical reference point on the page. Can be TopAlign, BottomAlign, or CenterAlign. Valid only when AnchoredPosition is AnchoredPosition.
Vertical-ReferencePoint	Vertically-RelativeTo_EnumValue	no	The vertical reference point on the page. Valid only when AnchoredPosition is AnchoredPosition.

**IDML Example 46. Anchored Text Frame**

```

<ParagraphStyleRange SpaceAfter="12">
    <CharacterStyleRange>
        <Content>This is an example paragraph. We've inserted an inline text frame in

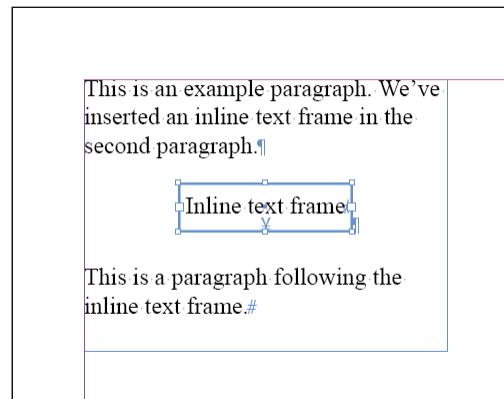
```

```

the second paragraph.</Content>
<br/>
</CharacterStyleRange>
</ParagraphStyleRange>
<ParagraphStyleRange SpaceAfter="12" Justification="CenterAlign">
<CharacterStyleRange AppliedCharacterStyle=
"CharacterStyle\k[No character style]">
<TextFrame Self="u106" ParentStory="u109" ItemTransform= "1 0 0 1 0 0">
<Properties>
<PathGeometry>
<GeometryPathType PathOpen="false">
<PathPointArray>
<PathPointType Anchor="97.2685546875 -225.5"
LeftDirection="97.2685546875 -225.5"
RightDirection="97.2685546875 -225.5"/>
<PathPointType Anchor="97.2685546875 -202"
LeftDirection="97.2685546875 -202"
RightDirection="97.2685546875 -202"/>
<PathPointType Anchor="183.5 -202" LeftDirection="183.5 -202"
RightDirection="183.5 -202"/>
<PathPointType Anchor="183.5 -225.5" LeftDirection="183.5 -225.5"
RightDirection="183.5 -225.5"/>
</PathPointArray>
</GeometryPathType>
</PathGeometry>
</Properties>
<AnchoredObjectSetting Self="u106AnchoredObjectSetting1"
AnchoredPosition="InlinePosition" SpineRelative="false"
LockPosition="false" PinPosition="true" AnchorPoint="BottomRightAnchor"
HorizontalAlignment="LeftAlign" HorizontalReferencePoint=
"TextFrame" VerticalAlignment="TopAlign"
VerticalReferencePoint="LineBaseline" AnchorXoffset="0"
AnchorYoffset="0" AnchorSpaceAbove="0"/>
</TextFrame>
<br/>
</CharacterStyleRange>
</ParagraphStyleRange>
<ParagraphStyleRangr SpaceAfter="12">
<CharacterStyleRange AppliedCharacterStyle=
"CharacterStyle\k[No character style]">
<Content>This is a paragraph following the inline text frame.</Content>
</CharacterStyleRange>
</ParagraphStyleRange>

```

Figure 38. Anchored Text Frame



### Anchored Graphics

Another type of anchored frame is a frame containing a graphic. These frames follow the same rules as anchored text frames, as discussed in the previous section.

#### IDML Example 47. Anchored Graphic

```
<Story Self="uce">
  <ParagraphStyleRange SpaceAfter="12">
    <CharacterStyleRange>
      <Content>This is an example paragraph. We've ve inserted an anchored frame
      containing a graphic in the second paragraph.</Content>
      <br/>
    </CharacterStyleRange>
  </ParagraphStyleRange>
  <ParagraphStyleRange Justification="CenterAlign">
    <CharacterStyleRange>
      <Rectangle Self="uec" ItemTransform="1 0 0 1 0 0">
        <Properties>
          <PathGeometry>
            <GeometryPathType PathOpen="false">
              <PathPointArray>
                <PathPointType Anchor="-72 -78.72" LeftDirection="-72 -78.72"
                  RightDirection="-72 -78.72"/>
                <PathPointType Anchor="-72 78.72" LeftDirection="-72 78.72"
                  RightDirection="-72 78.72"/>
                <PathPointType Anchor="72 78.72" LeftDirection="72 78.72"
                  RightDirection="72 78.72"/>
                <PathPointType Anchor="72 -78.72" LeftDirection="72 -78.72"
                  RightDirection="72 -78.72"/>
              </PathPointArray>
            </GeometryPathType>
          </PathGeometry>
        </Properties>
        <Image Self="ue6" Space="$ID/#Links_RGB" ActualPpi="300 300"
          EffectivePpi="300 300" ImageRenderingIntent="UseColorSettings"
          LocalDisplaySetting="Default" ImageTypeName="$ID/JPEG"
          ItemTransform="1 0 0 1 0 0"/>
      </Rectangle>
    </CharacterStyleRange>
  </ParagraphStyleRange>
```

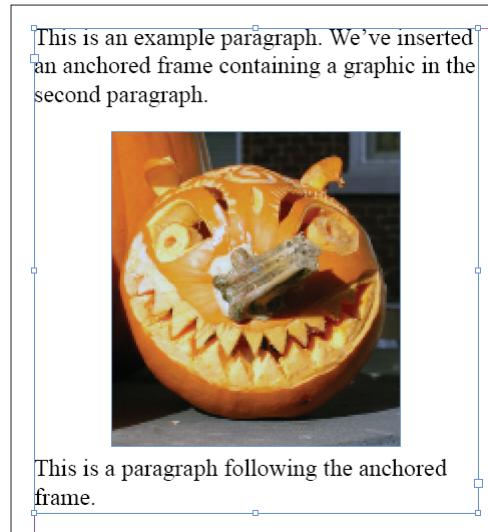
```

<Properties>
    <Profile type="string">$ID/Embedded</Profile>
    <GraphicBounds Left="0" Top="0" Right="144" Bottom="157.44"/>
</Properties>
<Link Self="ueb" AssetURL="$ID/" AssetID="$ID/"
LinkResourceURI="file:///ruri/documents/IDML/assets/pumpkin.jpg"
LinkResourceFormat="$ID/JPEG" StoredState="Normal" LinkClassID="35906"
LinkClientID="257" LinkResourceModified="false"
LinkObjectModified="false" ShowInUI="true" CanEmbed="true"
CanUnembed="true" CanPackage="true" ImportPolicy="NoAutoImport"
ExportPolicy="NoAutoExport"
LinkImportStamp="file 128385019586602016 396019"
LinkImportModificationTime="2007-11-02T11:32:38"
LinkImportTime="2008-06-14T15:12:33"/>
</Image>
</Rectangle>
<br/>
</CharacterStyleRange>
</ParagraphStyleRange>
<ParagraphStyleRange>
<CharacterStyleRange>
<Content>This is a paragraph following the anchored frame.</Content>
</CharacterStyleRange>
</ParagraphStyleRange>
</Story>

```

The graphic itself is not embedded in the `<Story>` element. Instead, the `<Link>` element stored in the `<Image>` element refers to a graphic file on disk.

*Figure 39. Inline Graphic*



### **Hyperlink Text Sources**

Hyperlink text sources differ from other inline elements as they appear as child elements of a `<CharacterStyleRange>`, and contain a `<Content>` element. Other inline elements appear as siblings of the `<Content>` element of the `<CharacterStyleRange>`.

#### **Schema Example 50. HyperlinkTextSource**

```
HyperlinkTextSource_Object = element HyperlinkTextSource {
    attribute Self { xsd:string },
    attribute Name { xsd:string }?,
    attribute Hidden { xsd:boolean }?,
    attribute AppliedCharacterStyle { xsd:string }?,
    element Properties {
        element Label { element KeyValuePair { KeyValuePair_TypeDef }*
            }?
    }
    ?
    ,
    (
        Note_Object*&
        Table_Object*&HyperlinkTextSource_Object
        TextVariableInstance_Object*&
        Footnote_Object*&
        TextFrame_Object*&
        Oval_Object*&
        Rectangle_Object*&
        GraphicLine_Object*&
        Polygon_Object*&
        Group_Object*&
        EPSText_Object*&
        FormField_Object*&
        Button_Object*&
        HyperlinkTextDestination_Object*&
        ParagraphDestination_Object*&
        Change_Object*&
        XMLElement_Object*&
        XMLComment_Object*&
        XMLInstruction_Object1*&
        DTD_Object*&
        HiddenText_Object*&
        ParagraphStyleRange_Object*&
        CharacterStyleRange_Object*&
        element Content {text}*&
        element Br {empty}*
    )
}
```

---

**Table 59. HyperlinkTextSource Properties Represented as Attributes**

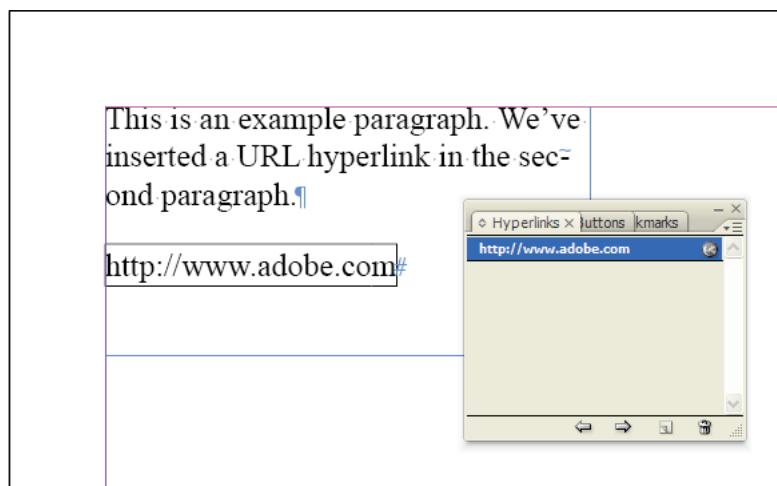
Name	Type	Req	Description
AppliedCharacter- Style	string	no	Character style of the hyperlink text source.

Name	Type	Req	Description
Hidden	boolean	no	If true, the HyperlinkTextSource is hidden.
Name	string	no	The name of the HyperlinkTextSource.

**IDML Example 48. Hyperlink Text Source**

```
<ParagraphStyleRange AppliedParagraphStyle="ParagraphStyle\kNormalParagraphStyle"
SpaceAfter="12">
<CharacterStyleRange>
<Content>This is an example paragraph. We've inserted a URL hyperlink in the
second paragraph.</Content>
<br/>
</CharacterStyleRange>
<CharacterStyleRange>
<HyperlinkTextSource Self="u12b" Name="Hyperlink_1" Hidden="false">
<Content>http://www.adobe.com</Content>
</HyperlinkTextSource>
</CharacterStyleRange>
</ParagraphStyleRange>
```

Figure 40. Hyperlink

**HyperlinkTextDestination**

A hyperlink text destination is one of the possible “targets” of a hyperlink.

**Schema Example 51. HyperlinkTextDestination**

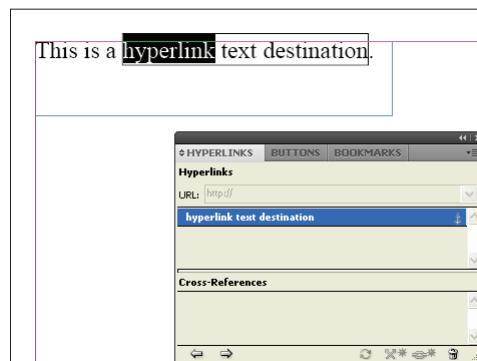
```
HyperlinkTextDestination_Object = element HyperlinkTextDestination {
    attribute Self { xsd:string },
    attribute Name { xsd:string }?,
    attribute Hidden { xsd:boolean }?,
    attribute DestinationUniqueKey { xsd:int }?,
    element Properties {
        element Label { element KeyValuePair { KeyValuePair_TypeDef }* }?
    }?
}
```

**Table 60. HyperlinkTextDestination Properties Represented as Attributes**

Name	Type	Req	Description
Destination-UniqueKey	int	no	A unique identifier for the hyperlink text destination.
Hidden	boolean	no	If true, the hyperlink text destination is hidden.
Name	string	no	The name of the hyperlink text destination.

**IDML Example 49. HyperlinkTextDestination**

```
<Story Self="uc8">
  <ParagraphStyleRange AppliedParagraphStyle=
    "ParagraphStyle/$ID/NormalParagraphStyle">
    <CharacterStyleRange AppliedCharacterStyle=
      "CharacterStyle/$ID/[No character style]">
      <Content>This is a </Content>
      <HyperlinkTextDestination Self="ue5"
        Name="hyperlink text destination" Hidden="false" DestinationUniqueKey="1"/>
      <HyperlinkTextSource Self="u104" Name="Hyperlink 1"
        Hidden="false" AppliedCharacterStyle="n">
        <Content>hyperlink text destination</Content>
      </HyperlinkTextSource>
      <Content>.</Content>
      <br/>
    </CharacterStyleRange>
  </ParagraphStyleRange>
</Story>
```

*Figure 41. HyperlinkTextDestination*

## 6.1 Fonts

The fonts and composite fonts used in a document are defined in the `Fonts.xml` file inside the Resources folder in an IDML package. Even an empty InDesign document contains references to default fonts and composite fonts.

**Note:** InDesign documents do not support font embedding, and fonts are not embedded in IDML. A font used within a document is represented by a simple reference to one of the fonts defined in the `<Fonts>` element. If fonts referred to in the IDML package are not installed on the system when you try to import an IDML package in InDesign, missing font warnings will appear.

There are two main elements stored in `Fonts.xml`: `<FontFamily>` and `<CompositeFont>`.

### 6.1.1 Font and Font Family

A font family is a group of similar fonts. For example, the fonts “Arial Regular”, “Arial Bold”, and “Arial Italic” all belong to the same font family. A `<FontFamily>` element groups together `<Font>` elements, and each `<Font>` element defines a specific font.

`<FontFamily>` and `<Font>` elements both contain an ID attribute. This attribute can be used as a reference throughout the IDML package.

#### Schema Example 52. `FontFamily`

```
FontFamily_Object = element FontFamily {
    attribute Self { xsd:string },
    attribute Name { xsd:string }?,
    (
        Font_Object*
    )
}
```

#### Schema Example 53. `Font`

```
Font_Object = element Font {
    attribute Self { xsd:string },
    attribute FontFamily { xsd:string }?,
    attribute Name { xsd:string }?,
    attribute PostScriptName { xsd:string }?,
    attribute Status { FontStatus_EnumValue }?,
    attribute FontStyleName { xsd:string }?,
    attribute FontType { FontTypes_EnumValue }?,
    attribute WritingScript { xsd:int }?,
    attribute FullName { xsd:string }?,
    attribute FullNameNative { xsd:string }?,
    attribute FontStyleNameNative { xsd:string }?,
    attribute PlatformName { xsd:string }?,
    attribute Version { xsd:string }?
}
```

The following example shows a `<Font>` definition for the “Myriad Pro Bold” font. Note: other fonts belonging to the “Myriad Pro” font family have been omitted from this example.

**IDML Example 50. Font**

```
<FontFamily Self="di77" Name="Myriad Pro">
  <Font Self="di77FontnMyriad Pro Bold" FontFamily="Myriad Pro"
    Name="Myriad Pro Bold" PostScriptName="$ID/MyriadPro-Bold" Status="Installed"
    FontStyleName="Bold" FontType="OpenTypeCFF" WritingScript="0"
    FullName="Myriad Pro Bold" FullNameNative="Myriad Pro Bold"
    FontStyleNameNative="Bold" PlatformName="$ID/"
    Version="Version 2.007;PS 002.000;Core 1.0.38;makeotf.lib1.7.9032"/>
</FontFamily>
```

**Table 61. Font Properties Represented as Attributes**

Name	Type	Req	Description
FontFamily	string	yes	A reference to the font family that contains this font, using the unique ID ( <code>Self</code> attribute) of the <code>&lt;FontFamily&gt;</code> element.
FontStyleName	string	yes	The name of the font style.
FontStyleName-Native	string	yes	The native name of the font style.
FontType	FontTypes_Enum-Value	yes	The type of font. Can be <code>Type1</code> , <code>TrueType</code> , <code>CID</code> , <code>ATC</code> , <code>Bitmap</code> , <code>OCF</code> , <code>OpenTypeCFF</code> , <code>OpenTypeCID</code> , <code>OpenTypeTT</code> , or <code>Unknown</code> .
FullName	string	yes	The full font name.
FullNameNative	string	yes	The full native name of the font.
Name	string	yes	The name of the Font.
PlatformName	string	yes	The platform font name.
PostScriptName	string	yes	The PostScript name of the font.
Status	FontStatus_Enum-Value	yes	The status of the font. Can be <code>Installed</code> , <code>NotAvailable</code> , <code>Fauxed</code> , <code>Substituted</code> , or <code>Unknown</code> .
Version	string	yes	The font version.
WritingScript	int	yes	The writing script: "0" for Roman, "1" for Japanese, "2" for Traditional Chinese.

**Referring to a Font**

Elements in an IDML package can refer to fonts using either the `Self` attribute or `Name` attribute of the `<Font>` element. There are many elements in the package that refer to fonts, including: `<CharacterStyle>`, `<CharacterStyleRange>`, `<ParagraphStyle>`, and `<ParagraphStyleRange>`.

Many elements use a combination of an `<AppliedFont>` property to specify the font family, and a `<FontStyle>` attribute to specify the font style.

The example below shows a `<CharacterStyleRange>` with a reference to the font "Arial Italic". The references here are by name:

**IDML Example 51. Referring to a Font**

```
<CharacterStyleRange AppliedCharacterStyle="CharacterStyle\k[No character style]">
```

```

    FontStyle="Italic">
    <Properties>
        <AppliedFont type="string">Arial</AppliedFont>
    </Properties>
    <Content>This is some text formatted using Arial Italic.</Content>
</CharacterStyleRange>

```

### 6.1.2 Composite Font

Composite fonts are part of the feature set of the Japanese version of InDesign, but can exist in any InDesign document. Even if you are not using composite fonts, you still need to be aware of them, and you should expect to encounter them in IDML documents.

A composite font is made up of multiple, separate fonts. For example, the default composite font, “[No composite font]”, consists of the following `<CompositeFontEntry>` elements: Kanji, Kana, Punctuation, Symbols, Alphabetic, and Numbers. Each of the `<CompositeFontEntry>` elements contains an `<AppliedFont>` property that defines the font it uses.

Schemas for `<CompositeFont>` and `<CompositeFontEntry>` are shown below. Note that in addition to `Name` and `Self` attributes, a `<CompositeFont>` contains multiple `<CompositeFontEntry>` elements, and each `<CompositeFontEntry>` refers to a `<Font>` in its `AppliedFont` property.

**Note:** Composite fonts in InDesign and IDML are not the same thing as composite fonts in PDF and PostScript. Composite fonts in PDF and PostScript are embedded font objects that are made up of glyphs from more than one font. InDesign’s composite fonts are based on Japanese fonts, are not embedded, and are made up of glyphs from multiple fonts. InDesign composite fonts are not files on disk or stored within the document; they’re temporary sets of characters created in the InDesign user interface.

#### Schema Example 54. CompositeFont

```

CompositeFont_Object = element CompositeFont {
    attribute Self { xsd:string },
    attribute Name { xsd:string },
    element Properties {
        element Label { element KeyValuePair { KeyValuePair_TypeDef }* }
    }?
}
?
(
    CompositeFontEntry_Object*
)
}

```

#### Schema Example 55. CompositeFontEntry

```

CompositeFontEntry_Object = element CompositeFontEntry {
    attribute Self { xsd:string },
    attribute Name { xsd:string }?,
    attribute FontStyle { xsd:string }?,
    attribute RelativeSize { xsd:double }?,
    attribute HorizontalScale { xsd:double }?,
    attribute VerticalScale { xsd:double }?,
    attribute CustomCharacters { xsd:string }?,
    attribute Locked { xsd:boolean }?,
}

```

```
attribute ScaleOption { xsd:boolean }?,
attribute BaselineShift { xsd:double }?,
element Properties {
    element AppliedFont {
        (object_type, xsd:string) |
        (string_type, xsd:string)
    }?&
    element Label { element KeyValuePair { KeyValuePair_TypeDef }* }
}?
}
```

## 6.2 Graphics

In an IDML package, the `Graphic.xml` file stores elements that define graphic attributes in an InDesign document. Page items and text elements refer to these attributes to apply formatting to page items and text. Inside `Graphic.xml`, you'll find the following elements.

- Colors
- Swatches
- Gradients
- Tints
- Inks
- Mixed Inks
- Mixed Ink Groups
- PastedSmoothShades
- Stroke Styles

### 6.2.1 Colors and Swatches

In the InDesign user interface, colors, tints, gradients, and mixed inks are all swatches, and can be applied to the fill or stroke of a page item or text. Swatches share similar attributes and are used in similar ways. Swatches are not the same as inks (see “Ink”), but each swatch generally corresponds to one ink or to a specific set of inks. A `<Color>` element, for example, might correspond to a single spot ink, or might be made up of percentages of process inks. (If a spot ink exists in the document, a `<Color>` element with the same name will also exist.) A `<MixedInk>` might be based on two or more spot inks, or on at least one spot ink and one or more process inks. For more on the relationship between swatches and inks, refer to the InDesign online help.

Colors can be specified as spot colors or process colors, and can be defined using the LAB, RGB, or CMYK color model.

To support consistent color across multiple devices, you can apply a color profile to a document using the color management attributes of the `<Document>` element in the `designmap.xml` file. The color management profile assigned to the document changes the appearance of the color for display, printing, and export, but does not change the base values of the definition of the color in IDML or in an InDesign document. For more detailed information on color management, please refer to the InDesign documentation.

**Note:** InDesign will create the “None” swatch, and the required “Paper,” “Black,” and “Registration” colors, even if those elements are not included in the IDML document. In addition, all InDesign documents contain three hidden, reserved process colors: Cyan, Magenta, and Yellow. You cannot create colors with these names in IDML or in the InDesign user interface.

## **Swatch**

In the InDesign user interface, all named colors, tints, gradients, and mixed inks are swatches, but there is only one object whose type is swatch—“None”, or no color. Every InDesign document contains this special swatch.

### **Schema Example 56. Swatch**

```
Swatch_Object = element Swatch {
    attribute Self { xsd:string },
    attribute Name { xsd:string },
    attribute ColorEditable { xsd:boolean }?,
    attribute ColorRemovable { xsd:boolean }?,
    attribute Visible { xsd:boolean }?,
    attribute SwatchCreatorID { xsd:int }?,
    element Properties {
        element Label { element KeyValuePair { KeyValuePair_TypeDef }* }
    }?
}
?
```

**Table 62. Swatch Properties Represented as Attributes**

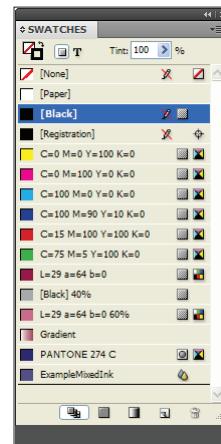
Name	Type	Req	Description
ColorEditable	boolean	no	If true, the swatch is editable.
ColorRemovable	boolean	no	If true, the swatch can be deleted. If false, the swatch is used in an imported graphic file in the document.
Name	string	yes	The name of the swatch.
SwatchCreatorID	int	no	The unique ID of the creator (or vendor) of the swatch. SwatchCreatorID is useful when the swatch added is from a color library that InDesign shipped with. The SwatchCreator-ID allows InDesign to load the proper color/swatch library quickly. If it is an user defined swatch, it simply defaults to the InDesign Swatch-CreatorID.
Visible	boolean	no	If true, the swatch is visible in the user interface. Unnamed process color and gradient swatches can have this attribute set to false. All named swatches should have this flag set to true.

All <Color>, <Tint>, <Gradient>, <MixedInk>, and <MixedInkGroup> elements inherit the above properties.

### **IDML Example 52. “None” Swatch**

```
<Swatch Self="Swatch\cNone" Name="None" ColorEditable="false" Color-
Removable="false" Visible="true" SwatchCreatorID="7937"/>
```

Figure 42. InDesign Swatches



## Color

The `<Color>` element corresponds to a color in a document, including both named and unnamed colors. The `Model` attribute specifies the color model, the `Space` attribute specifies the color space, and the `ColorValue` attribute contains the corresponding array of values that define the color in the appropriate color model. For information on color models and color spaces, refer to the InDesign online help.

### Schema Example 57. Color

```
Color_Object = element Color {
    attribute Self { xsd:string },
    attribute Model { ColorModel_EnumValue }?,
    attribute Space { ColorSpace_EnumValue }?,
    attribute ColorValue { list { xsd:double * } }?,
    attribute ColorOverride { ColorOverride_EnumValue }?,
    attribute BaseColor { xsd:string }?,
    attribute SpotInkAliasSpotColorReference { xsd:string }?,
    attribute AlternateSpace { ColorSpace_EnumValue }?,
    attribute AlternateColorValue { list { xsd:double * } }?,
    attribute Name { xsd:string },
    attribute ColorEditable { xsd:boolean }?,
    attribute ColorRemovable { xsd:boolean }?,
    attribute Visible { xsd:boolean }?,
    attribute SwatchCreatorID { xsd:int }?,
    element Properties {
        element Label { element KeyValuePair { KeyValuePair_TypeDef }* }?
    }?
}
```

In addition to the attributes shared with the `<Swatch>` element, the `<Color>` element also defines the following attributes.

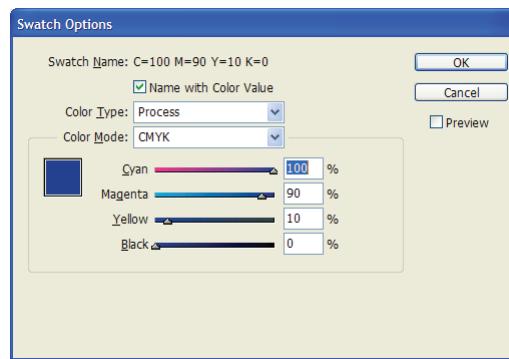
**Table 63. Color Properties Represented as Attributes**

Name	Type	Req	Description
AlternateColor-Value	list of doubles as a space-separated string	no	An alternate color value for the color. The <ColorValue> of the <Color> is defined as a sequence of values in the order in which they appear in the color space specified by the AlternateSpace attribute. Leave empty if AlternateSpace is NoAlternateColor.
AlternateSpace	ColorSpace_Enum-Value	no	The color space for the alternate color. Can be RGB, CMYK, LAB, MixedInk, NoAlternateColor.
BaseColor	string	no	Specifies the color that this color is based on, if any. Use n if the color is not based on another color.
ColorOverride	ColorOverride_EnumValue	no	Can be Normal, Specialpaper, Specialblack, Specialregistration, Hiddenreserved, or Mixedinkparent. Use Normal when defining color swatches in IDML—the other enumeration values are for special, default colors.
ColorValue	list of doubles as a space-separated string	no	The number of values required and the range depends on the color space. For RGB, specify three values, with each value in the range 0 to 255; for CMYK, specify four values representing C, M, Y, and K, with each value in the range 0 to 100; for LAB, specify three values representing L (Range: 0 to 100), A (Range: -128 to 127), and B (Range: -128 to 127).
Model	ColorModel_Enum-Value	no	The color model. Can be Spot, or Process.
Space	ColorSpace_Enum-Value	no	The color space. Can be RGB, CMYK, or LAB.
SpotInkAliasSpot-ColorReference	string	no	A spot color can be aliased to another spot ink or process ink. From the user interface, this is done through the Ink Manager dialog. This attribute contains a reference to the original spot ink.

**IDML Example 53. Process Color**

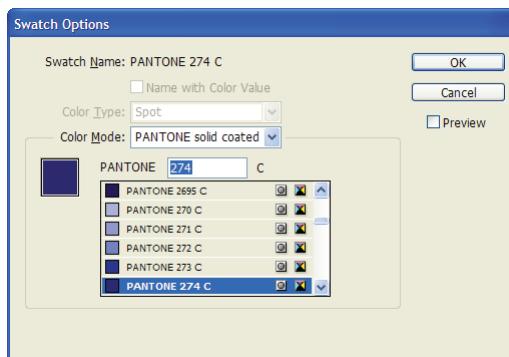
```
<Color Self="Color\C=100 M=90 Y=10 K=0" Model="Process" Space="CMYK"
ColorValue="100 90 10 0" ColorOverride="Normal" BaseColor="n"
AlternateSpace="NoAlternateColor" AlternateColorValue="" Name="C=100 M=90 Y=10
K=0" ColorEditable="true" ColorRemovable="true" Visible="true" SwatchCreator-
ID="7937"/>
```

Figure 43. Process Color

**IDML Example 54. Spot Color**

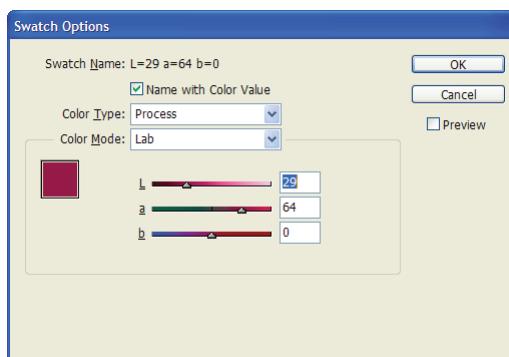
```
<Color Self="Color\PANTONE 274 C" Model="Spot" Space="CMYK" ColorValue="100 100 0 28" ColorOverride="Normal" BaseColor="n" SpotInkAliasSpotColorReference="n" AlternateSpace="LAB" AlternateColorValues="12.549019607843137 23 -42" Name="PANTONE 274 C" ColorEditable="true" ColorRemovable="true" Visible="true" SwatchCreatorID="31500"/>
```

Figure 44. SpotColor

**IDML Example 55. LAB Color**

```
<Color Self="Color\L=29 a=64 b=0" Model="Process" Space="LAB" ColorValue="29 64 0" ColorOverride="Normal" BaseColor="n" AlternateSpace="NoAlternateColor" AlternateColorValue="" Name="L=29 a=64 b=0" ColorEditable="true" ColorRemovable="true" Visible="true" SwatchCreatorID="7937"/>
```

Figure 45. LAB Color



## **Gradient**

A gradient is a blend between two or more colors or between two tints of the same color. Gradients can include the “Paper” color, process colors, spot colors, or mixed inks using any color model or color space. Gradients are defined by a series of gradient stops. A gradient stop is the point at which a gradient changes from one color to another. Gradients can be linear or radial.

In IDML, each `<Gradient>` element has a `Type` attribute that defines the type of gradient (Linear or Radial). `<Gradient>` elements also contain two or more `<GradientStop>` elements. Each `<GradientStop>` element contains a reference to a color in the `<StopColor>` element, a `Location` attribute that specifies the location of the gradient stop in the gradient (as a percentage of the total width of the gradient), and a `Midpoint` attribute that defines the midpoint of the change in color from this gradient stop to the next (as a percentage of the distance between the two gradient stops).

When mixing gradients that contain gradient stops with different color spaces, InDesign looks through all of the gradient stops for the widest gamut color space and uses it as a unified color space. However, if CMYK color space is one of the gradient stops, then CMYK color space is always the preferred unified space. All process color gradient stops that are not in the unified color space are converted to the unified space before blending.

For more on gradients, refer to the InDesign online help.

### **Schema Example 58. Gradient**

```
Gradient_Object = element Gradient {
    attribute Self { xsd:string },
    attribute Type { GradientType_EnumValue }?,
    attribute Name { xsd:string },
    attribute ColorEditable { xsd:boolean }?,
    attribute ColorRemovable { xsd:boolean }?,
    attribute Visible { xsd:boolean }?,
    attribute SwatchCreatorID { xsd:int }?,
    element Properties {
        element Label { element KeyValuePair { KeyValuePair_TypeDef }* }?
    }?
}
?
(
    GradientStop_Object*
)
}
```

In addition to the attributes shared with the `<Swatch>` element, the `<Gradient>` element also defines the following attributes.

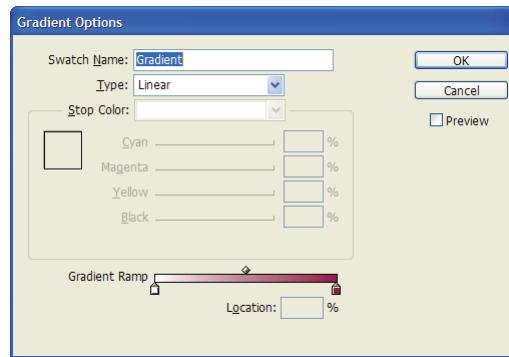
**Table 64. Gradient Properties Represented as Attributes**

Name	Type	Req	Description
Type	GradientType_EnumValue	no	The gradient type. Can be Linear or Radial.

**IDML Example 56. Gradient**

```
<Gradient Self="Gradient\ExampleGradient" Type="Linear" Name="ExampleGradient">
  ColorEditable="true" ColorRemovable="true" Visible="true" SwatchCreatorID="7937">
    <GradientStop Self="ucdGradientStop0" StopColor="Color\u7e" Location="0"/>
    <GradientStop Self="ucdGradientStop1" StopColor="Color\cL=29 a=64 b=0"
      Location="100" Midpoint="50"/>
  </Gradient>
```

Figure 46. Gradient



The example gradient contains two `<GradientStop>` elements, and each `<GradientStop>` refers to the `Self` value of a `<Color>` element in the `Graphic.xml` file.

**GradientStop****Schema Example 59. GradientStop**

```
GradientStop_Object = element GradientStop {
  attribute Self { xsd:string },
  attribute StopColor { xsd:string }?,
  attribute Location { xsd:double {minInclusive="0" maxInclusive="100"} }?,
  attribute Midpoint { xsd:double {minInclusive="13" maxInclusive="87"} }?
}
```

**Table 65. GradientStop Properties Represented as Attributes**

Name	Type	Req	Description
Location	double	no	The starting location (as a percentage of the gradient length) of the gradient stop on the gradient. (Range: 0 to 100).
Midpoint	double	no	The mid-point (as a percentage of the gradient length) of the gradient stop. (Range: 13 to 87)
StopColor	string	no	The color, tint, or mixed ink applied to the gradient stop.

**Tint**

A tint is a color that is based on a percentage of another a color. The appearance of a tint is determined by attribute `BaseColor` (a reference to the color that the tint is based on) and the

attribute `TintValue` (the percentage of the base color). For more on tints and their relationship with their base colors, refer to the InDesign documentation.

**Schema Example 60. Tint**

```

Tint_Object = element Tint {
    attribute Self { xsd:string },
    attribute TintValue { xsd:double {minInclusive="0" maxInclusive="100"} }?,
    attribute BaseColor { xsd:string },
    attribute Name { xsd:string },
    attribute ColorOverride { ColorOverride_EnumValue }?,
    attribute SpotInkAliasSpotColorReference { xsd:string }?,
    attribute AlternateSpace { ColorSpace_EnumValue }?,
    attribute AlternateColorValue { list { xsd:double * } }?,
    attribute ColorEditable { xsd:boolean }?,
    attribute ColorRemovable { xsd:boolean }?,
    attribute Visible { xsd:boolean }?,
    attribute SwatchCreatorID { xsd:int }?,
    element Properties {
        element Label { element KeyValuePair { KeyValuePair_TypeDef }* }
    }?
}
?
```

In addition to the attributes shared with the `<Swatch>`, the `<Tint>` element also defines the following attributes.

**Table 66. Tint Properties Represented as Attributes**

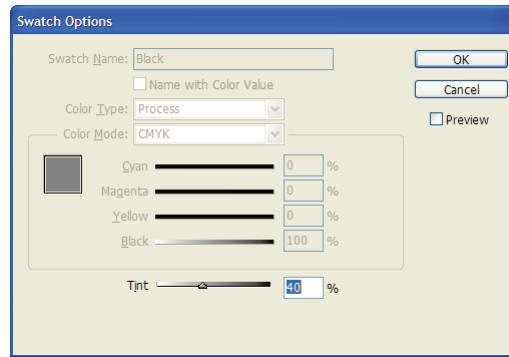
Name	Type	Req	Description
AlternateColor-Value	list of doubles as a space-separated string	no	An alternate color value for the tint. The <code>&lt;ColorValue&gt;</code> of the <code>&lt;Tint&gt;</code> is defined as a sequence of values in the order in which they appear in the color space specified by the <code>AlternateSpace</code> attribute. Leave empty if <code>AlternateSpace</code> is <code>NoAlternateColor</code> .
AlternateSpace	ColorSpace_Enum-Value	no	The color space for the alternate color. Can be RGB, CMYK, LAB, MixedInk, NoAlternateColor.
BaseColor	string	yes	A reference to the color that the tint is based on.
ColorOverride	ColorOverride_EnumValue	no	Can be Normal, Specialpaper, Specialblack, Specialregistration, Hiddenreserved, or Mixedinkparent. Use Normal when defining tint swatches.
SpotInkAliasSpot-ColorReference	string	no	A spot color can be aliased to another spot ink or process ink. From the user interface, this is done through the Ink Manager dialog. This attribute contains a reference to the original spot ink.

Name	Type	Req	Description
TintValue	double	no	The percent of the base color. For process colors, the TintValue is the percentage of each process ink that makes up the base color. For RGB, each component is multiplied by $0.8 + 0.2$ , or $r,g,b = \text{tint}*(r,g,b) + (1-\text{tint})$ . Tints of Lab colors use the same formula as RGB, but add the L component $(1-\text{tint})*100.0$ .

**IDML Example 57. Tint**

```
<Tint Self="Tint\[Black] 40%" TintValue="40" BaseColor="Color\Black" Name="[Black] 40%" ColorOverride="Normal" AlternateSpace="NoAlternateColor" AlternateColorValue="" ColorEditable="true" ColorRemovable="true" Visible="true" SwatchCreatorID="7937"/>
```

Figure 47. Tint

**Mixed Ink**

A mixed ink is a swatch created by mixing inks—you can combine up to sixteen spot inks, or mix one spot ink with one or more process inks. For more on mixed inks, refer to the InDesign documentation.

**Schema Example 61. MixedInk**

```
MixedInk_Object = element MixedInk {
    attribute Self { xsd:string },
    attribute Model { ColorModel_EnumValue }?,
    attribute Space { ColorSpace_EnumValue }?,
    attribute InkList { list { xsd:string * } }?,
    attribute InkPercentages { list { xsd:double * } }?,
    attribute BaseColor { xsd:string }?,
    attribute InkNameList { list { xsd:string * } }?,
    attribute MixedInkSpotColorNameList { list { xsd:string * } }?,
    attribute MixedInkSpotColorList { list { xsd:string * } }?,
    attribute Name { xsd:string },
    attribute ColorEditable { xsd:boolean }?,
    attribute ColorRemovable { xsd:boolean }?,
    attribute Visible { xsd:boolean }?,
    attribute SwatchCreatorID { xsd:int }?,
    element Properties {
        element Label { element KeyValuePair { KeyValuePair_TypeDef } }*
    }
}
```

```

    }?
}
?
}

```

In addition to the attributes inherited from the `<Swatch>` element, the `<MixedInk>` element also defines the following attributes.

**Table 67. MixedInk Properties Represented as Attributes**

Name	Type	Req	Description
BaseColor	string	yes	The mixed ink group that a mixed ink swatch is based on. Use <code>n</code> for a standalone mixed ink.
InkList	list of ink references as a space-separated string	yes	The component inks. Each ink is represented by its unique ID (the value of its <code>Self</code> attribute).
InkNameList	list of ink names as a space-separated string	no	The names of the component inks.
InkPercentages	list of doubles as a space-separated string	no	The array of tint percentages for inks in the ink list (in the same order as the inks appear in the <code>InkList</code> attribute). Note: Specify a value for each ink.
MixedInkSpot-ColorList	list of spot color references as a space-separated string	no	The spot colors used in the mixed ink. Each spot color is represented by its unique ID (the value of its <code>Self</code> attribute).
MixedInkSpot-ColorNameList	list of color names as a space-separated string	no	The names of the spot colors used in the mixed ink.
Model	ColorModel_Enum-Value	no	The color model. Use <code>Mixedinkmodel</code> .
Space	ColorSpace_Enum-Value	no	The color space. Use <code>MixedInk</code> .

The following example shows a mixed ink with one spot ink (Pantone 274 C) and one process ink (Black).

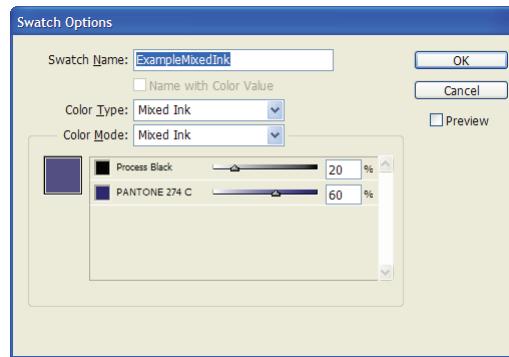
#### IDML Example 58. Mixed ink

```

<MixedInk Self="MixedInk\cExampleMixedInk" Model="Mixedinkmodel" Space="Mixed-
Ink" InkList="Ink\kProcess%20Black Ink\cPANTONE%20274%20C" InkPercentages="20 60"
BaseColor="n" InkNameList="$ID/Process%20Black PANTONE%20274%20C" MixedInkSpot-
ColorNameList="PANTONE%20274%20C" MixedInkSpotColorList="Color\cPANTONE%20274%20C"
Name="ExampleMixedInk" ColorEditable="true" ColorRemovable="true" Visible="true"
SwatchCreatorID="7937"/>

```

Figure 48. MixedInk



### **MixedInkGroup**

A mixed ink group contains a series of swatches created from incremental percentages of different process and spot color inks. For example, mixing a spot ink with five tints of a the default color Black (10%, 20%, 30%, 40%, and 50%) results in a mixed ink group that contains five different mixed ink swatches.

The mixed inks that make up a mixed ink group are the same as standalone mixed inks, except that the value of the BaseColor attribute for a mixed ink in a mixed ink group will contain a reference to the unique ID of the mixed ink group (rather than `n` for “none”).

#### **Schema Example 62. MixedInkGroup**

```
MixedInkGroup_Object = element MixedInkGroup {
    attribute Self { xsd:string },
    attribute Model { ColorModel_EnumValue }?,
    attribute InkList { list { xsd:string * } }?,
    attribute InkNameList { list { xsd:string * } }?,
    attribute MixedInkSpotColorNameList { list { xsd:string * } }?,
    attribute MixedInkSpotColorList { list { xsd:string * } }?,
    attribute Name { xsd:string },
    attribute ColorEditable { xsd:boolean }?,
    attribute ColorRemovable { xsd:boolean }?,
    attribute Visible { xsd:boolean }?,
    attribute SwatchCreatorID { xsd:int }?,
    element Properties {
        element Label { element KeyValuePair { KeyValuePair_TypeDef }* }?
    }?
}
```

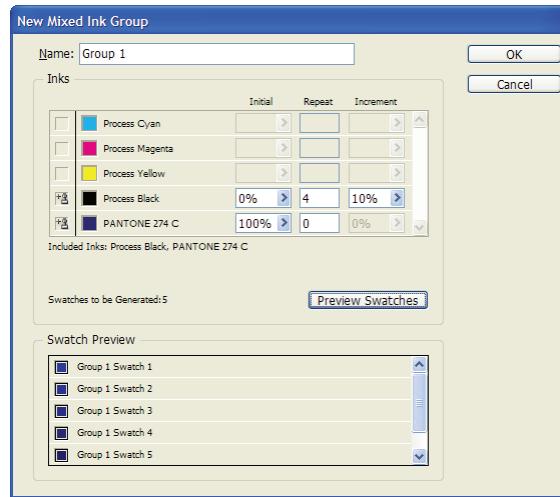
In addition to the attributes inherited from the `<Swatch>` element, the `<MixedInkGroup>` element also defines the following attributes.

**Table 68. MixedInkGroup Properties Represented as Attributes**

Name	Type	Req	Description
InkList	list of ink references as a space-separated string	yes	The inks used in the mixed ink group. Each ink is represented by its unique ID (the value of its Self attribute).
InkNameList	list of ink names as a space-separated string	no	The names of the inks used in the mixed ink.
MixedInkSpot-ColorList	list of spot color references as a space-separated string	no	The spot colors used in the mixed ink group. Each spot color is represented by its unique ID (the value of its Self attribute).
MixedInkSpot-ColorNameList	list of color names as a space-separated string	no	The names of the spot colors used in the mixed ink.
Model	ColorModel_Enum-Value	no	The color model. Use Mixedinkmodel.

**IDML Example 59. MixedInkGroup**

```
<MixedInkGroup Self="MixedInkGroup\cExampleMixedInkGroup" Model="Mixedinkmodel"
InkList="Ink\kProcess%20Black Ink\cPANTONE%20274%20C" InkNameList="$ID/Process%20
Black PANTONE%20274%20C" MixedInkSpotColorNameList="PANTONE%20274%20C" Mixed-
InkSpotColorList="Color\cPANTONE%20274%20C" Name="ExampleMixedInkGroup" Color-
Editable="true" ColorRemovable="true" Visible="true" SwatchCreatorID="7937"/>
```

*Figure 49. MixedInkGroup***Ink**

Inks are used to produce specific colors in printing. For process colors, Cyan, Magenta, Yellow and Black inks are mixed together to produce a range of colors. Every spot color, by contrast, represents an individual ink or printing plate. For more on the relationship between inks and colors, refer to the InDesign documentation.

**Schema Example 63. Ink**

```
Ink_Object = element Ink {
    attribute Self { xsd:string },
    attribute Name { xsd:string },
    attribute AliasInkName { xsd:string }?,
    attribute Angle { xsd:double {minInclusive="0" maxInclusive="360"} }?,
    attribute ConvertToProcess { xsd:boolean }?,
    attribute Frequency { xsd:double {minInclusive="1" maxInclusive="500"} }?,
    attribute NeutralDensity { xsd:double{minInclusive="0.001" maxInclusive="10"} }?,
    attribute PrintInk { xsd:boolean }?,
    attribute TrapOrder { xsd:int }?,
    attribute InkType { InkTypes_EnumValue }?,
    element Properties {
        element Label { element KeyValuePair { KeyValuePair_TypeDef }* }
    }?
}
?
```

**Table 69. Ink Properties Represented as Attributes**

Name	Type	Req	Description
AliasInkName	string	no	A reference to the unique ID ( <code>Self</code> attribute) of the alias ink. For more on ink aliases, refer to the InDesign documentation.
Angle	double	no	The screen angle of the ink. (Range: 0 to 360)
ConvertToProcess	boolean	no	Converts spot colors to process inks (for printing or export). Setting this value to true changes the printing or output values of the colors; it does not affect the definition of the spot colors in the document.
Frequency	double	no	The screen frequency of the ink. (Range: 1 to 500)
InkType	InkTypes_EnumValue	no	The trapping type of the ink. Can be <code>Normal</code> , <code>Opaque</code> , <code>Transparent</code> , <code>OpaqueIgnore</code> .
Name	string	yes	The name of the ink.
NeutralDensity	double	no	The neutral density of the ink used for trapping. (Range: 0.001 to 10.0)
PrintInk	boolean	no	If true (the default), prints the ink.
TrapOrder	int	no	The place of the ink in the trapping sequence.

**IDML Example 60. Process Magenta ink**

```
<Ink Self="Ink\Process Magenta" Name="$ID/Process Magenta" AliasInkName="[No-Alias]" Angle="15" ConvertToProcess="false" Frequency="70" NeutralDensity="0.76" PrintInk="true" TrapOrder="2" InkType="Normal"/>
```

Note that the “\$ID” preceding the name of the default ink, “Magenta” in the above example indicates that the name of this ink can be localized.

**IDML Example 61. Pantone 274 C Spot ink**

```
<Ink Self="Ink\cPANTONE 274 C" Name="PANTONE 274 C" AliasInkName="[No-
Alias]" Angle="45" ConvertToProcess="false" Frequency="70" Neutral-
Density="1.6130121989933028" PrintInk="true" TrapOrder="5" InkType="Normal"/>
```

**6.2.2 Stroke Styles**

The stroke styles used in an InDesign document are represented in an IDML package by the `<StrokeStyle>`, `<DashedStrokeStyle>`, `<DottedStrokeStyle>`, and `<StripedStrokeStyle>` elements in the `Graphics.xml` file.

**Schema Example 64. StrokeStyle**

```
StrokeStyle_Object = element StrokeStyle {
    attribute Self { xsd:string },
    attribute Name { xsd:string }
}
```

**Table 70. StrokeStyle Properties Represented as Attributes**

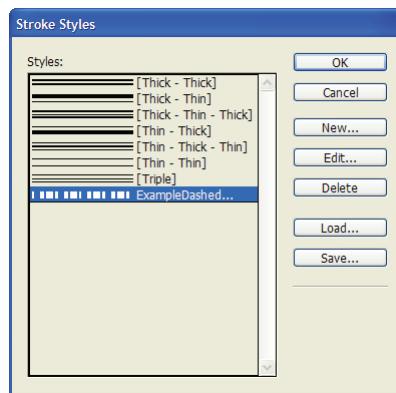
Name	Type	Req	Description
Name	string	yes	The name of the stroke style.

**IDML Example 62. Default StrokeStyles**

```
<StrokeStyle Self="StrokeStyle\Triple_Stroke" Name="$ID/Triple_Stroke"/>
<StrokeStyle Self="StrokeStyle\ThickThinThick" Name="$ID/ThickThinThick"/>
<StrokeStyle Self="StrokeStyle\ThinThickThin" Name="$ID/ThinThickThin"/>
<StrokeStyle Self="StrokeStyle\ThickThick" Name="$ID/ThickThick"/>
<StrokeStyle Self="StrokeStyle\ThickThin" Name="$ID/ThickThin"/>
<StrokeStyle Self="StrokeStyle\ThinThick" Name="$ID/ThinThick"/>
<StrokeStyle Self="StrokeStyle\ThinThin" Name="$ID/ThinThin"/>
<StrokeStyle Self="StrokeStyle\Japanese Dots" Name="$ID/Japanese Dots"/>
<StrokeStyle Self="StrokeStyle\White Diamond" Name="$ID/White Diamond"/>
<StrokeStyle Self="StrokeStyle\Left Slant Hash" Name="$ID/Left Slant Hash"/>
<StrokeStyle Self="StrokeStyle\Right Slant Hash" Name="$ID/Right Slant Hash"/>
<StrokeStyle Self="StrokeStyle\Straight Hash" Name="$ID/Straight Hash"/>
<StrokeStyle Self="StrokeStyle\Wavy" Name="$ID/Wavy"/>
<StrokeStyle Self="StrokeStyle\Canned Dotted" Name="$ID/Canned Dotted"/>
<StrokeStyle Self="StrokeStyle\Canned Dashed 3x2" Name="$ID/Canned Dashed 3x2"/>
<StrokeStyle Self="StrokeStyle\Canned Dashed 4x4" Name="$ID/Canned Dashed 4x4"/>
<StrokeStyle Self="StrokeStyle\Dashed" Name="$ID/Dashed"/>
<StrokeStyle Self="StrokeStyle\Solid" Name="$ID/Solid"/>
```

Only the default stroke styles will appear as `<StrokeStyle>` elements. All custom stroke styles will appear as `<DashedStrokeStyle>`, `<DottedStrokeStyle>`, or `<StripedStrokeStyle>` elements. The names of the default stroke styles above have the “\$ID” prefix, which means that the string will change based on the installed locale of the application. For more on the default stroke styles, refer to the InDesign online help.

Figure 50. InDesign Stroke Styles

**Schema Example 65. DashedStrokeStyle**

```
DashedStrokeStyle_Object = element DashedStrokeStyle {
    attribute Self { xsd:string },
    attribute DashArray { list { xsd:double * } }?,
    attribute StrokeCornerAdjustment { StrokeCornerAdjustment_EnumValue }?,
    attribute EndCap { EndCap_EnumValue }?,
    attribute Name { xsd:string }
}
```

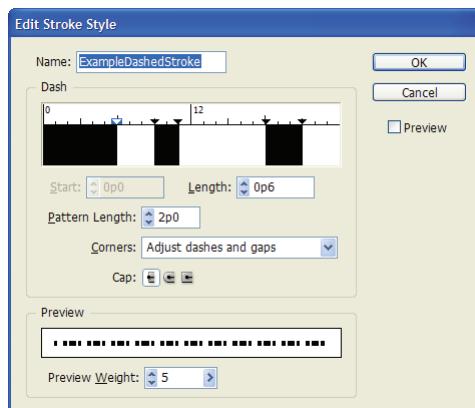
**Table 71. DashedStrokeStyle Properties Represented as Attributes**

Name	Type	Req	Description
DashArray	list of doubles as a space-separated string	no	The pattern of dashes and gaps that make up the dashed stroke, in the format [dash length1, gap length1, dash length2, gap length2, ...], in points. Define up to ten values. This pattern repeats along the path. The exact length of the dashes and gaps can be affected by the value of the StrokeCornerAdjustment attribute.
EndCap	EndCap_EnumValue	no	The end shape of the dashes. Can be ButtEndCap, RoundEndCap, or ProjectingEndCap. For a description of these end shapes, refer to the InDesign online help.
Name	string	no	The name of the dashed stroke style.
StrokeCorner-Adjustment	StrokeCorner-Adjustment_EnumValue	no	The corner adjustment applied to the dashed stroke style. Can be None, Dashes, Gaps, or DashesAndGaps. For a description of these adjustments, refer to the InDesign online help.

**IDML Example 63. DashedStrokeStyle**

```
<DashedStrokeStyle Self="DashedStrokeStyle\cExampleDashedStroke" DashArray="6 3 2
7 3 3" StrokeCornerAdjustment="DashesAndGaps" EndCap="ButtEndCap" Name="Example-
DashedStroke"/>
```

Figure 51. DashedStrokeStyle

**Schema Example 66. DottedStrokeStyle**

```
DottedStrokeStyle_Object = element DottedStrokeStyle {
    attribute Self { xsd:string },
    attribute DotArray { list { xsd:double * } }?,
    attribute StrokeCornerAdjustment { StrokeCornerAdjustment_EnumValue }?,
    attribute Name { xsd:string }
}
```

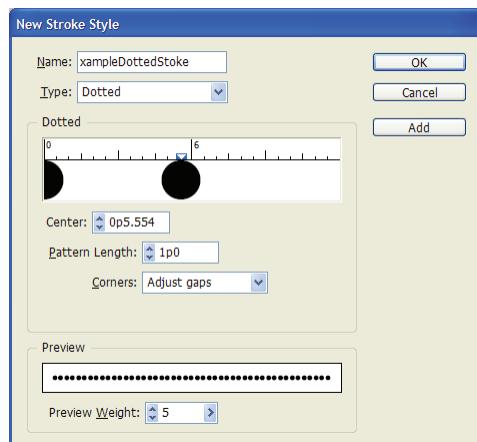
**Table 72. DottedStrokeStyle Properties Represented as Attributes**

Name	Type	Req	Description
DotArray	list of doubles as a space-separated string	no	The length of gaps between dots, in points (the diameter of the dots is determined by the stroke width). This pattern repeats along the path. The exact distance between the dots can be affected by the value of the StrokeCornerAdjustment attribute.
Name	string	no	The name of the dotted stroke style.
StrokeCorner-Adjustment	StrokeCorner-Adjustment_EnumValue	no	The corner adjustment applied to the dotted stroke style. Can be None, Dashes, Gaps, or DashesAndGaps. For a description of these adjustments, refer to the InDesign online help.

**IDML Example 64. DottedStrokeStyle**

```
<DottedStrokeStyle Self="DottedStrokeStyle\cExampleDottedStoke" Dot-Array="5.554054054054054 6.445945945945946" StrokeCornerAdjustment="Gaps" Name="ExampleDottedStoke"/>
```

Figure 52. DottedStrokeStyle

**Schema Example 67. StripedStrokeStyle**

```
StripedStrokeStyle_Object = element StripedStrokeStyle {
    attribute Self { xsd:string },
    attribute StripeArray { list { xsd:double * } }?,
    attribute Name { xsd:string }
}
```

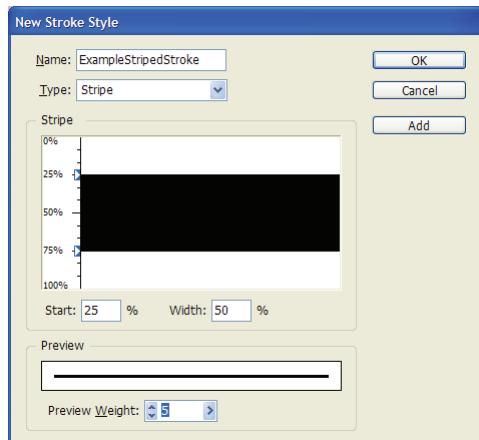
**Table 73. StripedStrokeStyle Properties Represented as Attributes**

Name	Type	Req	Description
Name	string	no	The name of the striped stroke style.
StripeArray	list of doubles as a space-separated string	no	The width and position of stripes in a striped stroke pattern. Each stripe is specified by a start-end pair in the format start1, end1, start2, end2; each value indicates a percentage of the stroke weight. Each value must be greater than the previous value. Range: 0 to 100.

**IDML Example 65. StripedStrokeStyle**

```
<StripedStrokeStyle Self="StripedStrokeStyle\cExampleStripedStroke" Stripe-  
Array="25 75" Name="ExampleStripedStroke"/>
```

Figure 53. *StripedStrokeStyle*



## 6.3 Preferences

IDML files can contain a number of preferences elements that control both document preferences and various default values. In an IDML package, preferences elements are found in the `Preferences.xml` file in the Resources folder.

Many of these elements no effect on the interpretation of the IDML file, but do have an effect on the user interface settings in the InDesign document that results when a user opens the IDML file. `<DataMerge>`, `<LayoutAdjustmentPreference>`, `<XMLImportPreference>`, and `<ExportForWebPreference>`, are examples of preferences that do not affect the reconstruction of elements (page items or text, for example) in an IDML file. By contrast, some of the preferences, such as `<TextDefault>` and `<AnchoredObjectDefault>` can have a very large effect on the formatting of objects specified in the IDML package.

In general, when you construct a page item (in a `<Spread>` element) or text (in a `<Story>` element), you can choose to omit many of the formatting options that can be applied to the element. When you do this, InDesign will apply default values to the object as it opens the IDML package. Most of these default values are stored in preferences elements.

This means that you can choose an approach that works best for you. You can achieve complete control by fully-specifying every attribute and element of every element you add to an IDML document, or you can rely on defaults and preferences for common values and specify only a minimum—this can make your IDML file smaller and easier to read. Or you can mix and match these approaches in an IDML file.

When you specify a preference in the IDML file, it will override the corresponding application preference for this document. If you do not specify a preference, InDesign will apply the default value from the IDML defaults file.

### **Schema Example 68. Preferences\_File**

```
Preferences_File = element idPkg:Preferences {
    DataMerge_Object?&
    DataMergeOption_Object?&
    LayoutAdjustmentPreference_Object?&
    XMLImportPreference_Object?&
    XMLExportPreference_Object?&
    XMLPreference_Object?&
    ExportForWebPreference_Object?&
    TransparencyPreference_Object?&
    TransparencyDefaultContainerObject_Object?&
    StoryPreference_Object?&
    TextFramePreference_Object?&
    TextPreference_Object?&
    TextDefault_Object?&
    DictionaryPreference_Object?&
    AnchoredObjectDefault_Object?&
    AnchoredObjectSetting_Object?&
    BaselineFrameGridOption_Object?&
    FootnoteOption_Object?&
    TextWrapPreference_Object?&
    DocumentPreference_Object?&
    GridPreference_Object?&
    GuidePreference_Object?&
```

```

MarginPreference_Object?&
PasteboardPreference_Object?&
ViewPreference_Object?&
PrintPreference_Object?&
PrintBookletOption_Object?&
PrintBookletPrintPreference_Object?&
MetadataPacketPreference_Object?&
IndexOptions_Object?&
IndexHeaderSetting_Object?&
PageItemDefault_Object?&
FrameFittingOption_Object?&
ButtonPreference_Object?&
TinDocumentDataObject_Object?&
LayoutGridDataInformation_Object?&
StoryGridDataInformation_Object?&
CjkGridPreference_Object?&
MojikumiUiPreference_Object?&
ChapterNumberPreference_Object?
}

```

### 6.3.1 DataMerge

The <DataMerge> element stores the settings that are be used by the DataMerge feature of InDesign. This preference object does not have any effect on the interpretation of layout elements (page items or stories, for example) in the IDML file.

#### Schema Example 69. DataMerge

```

DataMerge_Object = element DataMerge {
    attribute Self { xsd:string },
    attribute DataSourceFileType { DataSourceType_EnumValue }?,
    attribute DataSourceFile { xsd:string }?,
    (
        DataMergeField_Object*
    )
}

```

---

**Table 74. DataMerge Properties Represented as Attributes**

Name	Type	Req	Description
DataSourceFile	string	no	The file path to the data file.
DataSourceFileType	DataSourceType_EnumValue	no	The separator type used in the data file. Use CommaSeparated or TabDelimited.

#### Schema Example 70. DataMergeField

```

DataMergeField_Object = element DataMergeField {
    attribute Self { xsd:string },
    attribute FieldName { xsd:string }?
}

```

**Table 75. DataMergeField Properties Represented as Attributes**

Name	Type	Req	Description
FieldName	string	yes	The name of the field.

### 6.3.2 DataMergeOption

The <DataMergeOption> element stores the preferences for the DataMerge feature of InDesign.

**Schema Example 71. DataMergeOption**

```
DataMergeOption_Object = element DataMergeOption {
    attribute Self { xsd:string },
    attribute FittingOption { Fitting_EnumValue }?,
    attribute CenterImage { xsd:boolean }?,
    attribute LinkImages { xsd:boolean }?,
    attribute RemoveBlankLines { xsd:boolean }?,
    attribute CreateNewDocument { xsd:boolean }?,
    attribute DocumentSize { xsd:int }?
}
```

**Table 76. DataMergeOption Properties Represented as Attributes**

Name	Type	Req	Description
CenterImage	boolean	no	If true, centers the image in the frame; preserves the frame size as well as content size and proportions. Note: If the content is larger than the frame, content around the edges is obscured by the bounding box of the frame.
CreateNewDocument	boolean	no	If true, creates a new document when records are merged.
DocumentSize	int	no	The maximum number of pages per document.
FittingOption	Fitting_EnumValue	no	Instructions for fitting content in a frame. Use None, ContentToFrame, Proportionally, or FillProportionally.
LinkImages	boolean	no	If true, links images to the target document. If false, embeds images in the target document.
RemoveBlankLines	boolean	no	If true, removes blank lines caused by empty fields.

### 6.3.3 LayoutAdjustmentPreference

The <LayoutAdjustmentPreference> element stores the settings that are be used by the Layout Adjustment feature of InDesign. This preference object does not have any effect on the interpretation of layout elements (page items or stories, for example) in the IDML file.

**Schema Example 72. LayoutAdjustmentPreference**

```
LayoutAdjustmentPreference_Object = element LayoutAdjustmentPreference {
    attribute Self { xsd:string },
    attribute EnableLayoutAdjustment { xsd:boolean }?,
```

```

        attribute SnapZone { xsd:double {minInclusive="0" maxInclusive="12"} }?,
        attribute AllowGraphicsToResize { xsd:boolean }?,
        attribute AllowRulerGuidesToMove { xsd:boolean }?,
        attribute IgnoreRulerGuideAlignments { xsd:boolean }?,
        attribute IgnoreObjectOrLayerLocks { xsd:boolean }?
    }
}

```

**Table 77. LayoutAdjustmentPreference Properties Represented as Attributes**

Name	Type	Req	Description
AllowGraphicsTo- Resize	boolean	no	If true, allows graphics to be resized.
AllowRulerGuides- ToMove	boolean	no	If true, allows ruler guides to move.
EnableLayout- Adjustment	boolean	no	If true, layout adjustment is enabled.
IgnoreObjectOr- LayerLocks	boolean	no	If true, ignores object or layer locks.
IgnoreRulerGuide- Alignments	boolean	no	If true, ignores ruler guide alignments.
SnapZone	double	no	The range within which an object snaps to guides. Range: 0 to 12.

### 6.3.4 XMLImportPreference

The <XMLImportPreference> element stores the settings that are be used by when importing XML into an InDesign document. This preference object does not have any effect on the interpretation of XML elements in the IDML file.

#### Schema Example 73. XMLImportPreference

```

XMLImportPreference_Object = element XMLImportPreference {
    attribute Self { xsd:string },
    attribute ImportToSelected { xsd:boolean }?,
    attribute ImportStyle { XMLImportStyles_EnumValue }?,
    attribute CreateLinkToXML { xsd:boolean }?,
    attribute RepeatTextElements { xsd:boolean }?,
    attribute IgnoreUnmatchedIncoming { xsd:boolean }?,
    attribute ImportTextIntoTables { xsd:boolean }?,
    attribute IgnoreWhitespace { xsd:boolean }?,
    attribute RemoveUnmatchedExisting { xsd:boolean }?,
    attribute AllowTransform { xsd:boolean }?,
    attribute ImportCALSTables { xsd:boolean }?,
    element Properties {
        element TransformFilename {
            (file_type, xsd:string ) |
            (enum_type, XMLTransformFile_EnumValue )
        }?&
        element TransformParameters { element ListItem {
            attribute Name { xsd:string },
            attribute Value { xsd:string }
        }?&
    }
}

```

```

    } *
}
?
}
?
}
}

```

**Table 78. XMLImportPreference Properties Represented as Attributes**

Name	Type	Req	Description
AllowTransform	boolean	no	If true, transforms the XML using an XSLT file.
CreateLinkToXML	boolean	no	If true, creates a link to the imported XML file. If false, embeds the file.
IgnoreUnmatched-Incoming	boolean	no	If true, ignores elements that do not match the existing structure. Note: Valid only when import style is merge.
IgnoreWhitespace	boolean	no	If true, leaves existing content in place if the matching XML content contains only whitespace characters such as a carriage return or a tab character. Note: Valid only when import style is merge.
ImportCALSTables	boolean	no	If true, imports CALS tables as InDesign tables.
ImportStyle	XMLImportStyles_EnumValue	no	The style of incorporating imported XML content into the document. Can be AppendImport or MergeImport .
ImportTextInto-Tables	boolean	no	If true, imports text into tables if tags match placeholder tables and their cells. Note: Valid only when import style is MergeImport .
ImportToSelected	boolean	no	If true, imports into the selected XML element. If false, imports at the root element.
RemoveUnmatched-Existing	boolean	no	If true, deletes existing elements or placeholders in the document that do not have matches in the XML file. Note: Valid only when import style is MergeImport .
RepeatText-Elements	boolean	no	If true, repeating text elements inherit the formatting applied to placeholder text. Note: Valid only when import style is MergeImport .

**Table 79. XMLImportPreference Properties Represented as Elements**

Name	Type	Req	Description
TransformFilename	string	no	If true, allow The name of the XSLT file or StylesheetInXML (when the XSLT file is referred to in the XML document). Note: Valid when allow transform is true.
Transform-Parameters	ListItem	no	Stylesheet parameters as a series of ListItem elements containing name/value pairs.

### 6.3.5 XMLExportPreference

The <XMLExportPreference> element stores the settings that are used by when exporting XML into an InDesign document. This preference object does not have any effect on the interpretation of XML elements in the IDML file.

#### Schema Example 74. XMLExportPreference

```
XMLExportPreference_Object = element XMLExportPreference {
    attribute Self { xsd:string },
    attribute ViewAfterExport { xsd:boolean }?,
    attribute ExportFromSelected { xsd:boolean }?,
    attribute FileEncoding { XMLFileEncoding_EnumValue }?,
    attribute Ruby { xsd:boolean }?,
    attribute ExcludeDtd { xsd:boolean }?,
    attribute CopyOriginalImages { xsd:boolean }?,
    attribute CopyOptimizedImages { xsd:boolean }?,
    attribute CopyFormattedImages { xsd:boolean }?,
    attribute ImageConversion { ImageConversion_EnumValue }?,
    attribute GIFOptionsPalette { GIFOptionsPalette_EnumValue }?,
    attribute GIFOptionsInterlaced { xsd:boolean }?,
    attribute JPEGOptionsQuality { JPEGOptionsQuality_EnumValue }?,
    attribute JPEGOptionsFormat { JPEGOptionsFormat_EnumValue }?,
    attribute AllowTransform { xsd:boolean }?,
    attribute CharacterReferences { xsd:boolean }?,
    attribute ExportUntaggedTablesFormat { XMLExportUntaggedTablesFormat_EnumValue }?,
    element Properties {
        element PreferredBrowser {
            (file_type, xsd:string ) |
            (enum_type, NothingEnum_EnumValue )
        }?&
        element TransformFilename {
            (file_type, xsd:string ) |
            (enum_type, XMLTransformFile_EnumValue )
        }?
    }
}?
```

**Table 80. XMLExportPreference Properties Represented as Attributes**

Name	Type	Req	Description
AllowTransform	boolean	no	If true, transforms the XML using an XSLT file.
Character-References	boolean	no	If true, replaces special characters with character references.
CopyFormatted-Images	boolean	no	If true, copies formatted images to the images subfolder.
CopyOptimized-Images	boolean	no	If true, copies optimized images to the images subfolder.
CopyOriginal-Images	boolean	no	If true, copies original images to the images subfolder.

Name	Type	Req	Description
ExcludeDtd	boolean	no	If true, excludes the DTD from the exported XML content.
ExportFrom-Selected	boolean	no	If true, exports XML content from the selected XML element. If false, exports the entire document.
ExportUntagged-TablesFormat	XMLExport-UntaggedTables-Format_EnumValue	no	The export format for untagged tables in tagged stories. Can be None or CALS.
FileEncoding	XMLFileEncoding_EnumValue	no	The file encoding type for exporting XML content. Can be UTF8, UTF16, or ShiftJIS.
GIFOptions-Interlaced	boolean	no	If true, generates interlaced GIFs. Note: Not valid when image conversion is JPEG.
GIFOptionsPalette	GIFOptions-Palette_EnumValue	no	The color palette for GIF conversion. Note: Not valid when image conversion is JPEG. Can be AdaptivePalette, MacintoshPalette, Web-Palette, or WindowsPalette.
ImageConversion	ImageConversion_EnumValue	no	The file format to use for converted images. Note: Valid only when copy optimized images and/or copy formatted images is true. Can be Automatic, JPEG, or GIF.
JPEGOptionsFormat	JPEGOptions-Format_EnumValue	no	The formatting method for converted JPEG images. Note: Not valid when image conversion is GIF. Can be BaselineEncoding or ProgressiveEncoding.
JPEGOptions-Quality	JPEGOptions-Quality_EnumValue	no	The quality of converted JPEG images. Note: Not valid when image conversion is GIF. Can be Low, Medium, High, or Maximum.
Ruby	boolean	no	If true, includes Ruby text in the exported XML content.
ViewAfterExport	boolean	no	If true, displays exported XML content in a specified viewer.

**Table 81. XMLExportPreference Properties Represented as Elements**

Name	Description
PreferredBrowser	The preferred browser for viewing XML.
TransformFilename	The name of the XSLT file. Note: Valid when allow transform is true.

### 6.3.6 XMLPreference

The <XMLPreference> element stores general XML preferences for an InDesign document. This preference object does not have any effect on the interpretation of XML elements in the IDML file.

#### Schema Example 75. XMLPreference

```
XMLPreference_Object = element XMLPreference {
    attribute Self { xsd:string },
    attribute DefaultStoryTagName { xsd:string }?,
```

```

        attribute DefaultTableName { xsd:string }?,
        attribute DefaultCellTagName { xsd:string }?,
        attribute DefaultImageTagName { xsd:string }?,
        element Properties {
            element DefaultStoryTagColor { InDesignUIColorType_TypeDef }?&
            element DefaultTableTagColor { InDesignUIColorType_TypeDef }?&
            element DefaultCellTagColor { InDesignUIColorType_TypeDef }?&
            element DefaultImageTagColor { InDesignUIColorType_TypeDef }?
        }
    ?
}
}

```

**Table 82. XMLPreference Properties Represented as Attributes**

Name	Type	Req	Description
DefaultCellTagName	string	no	The name of the default tag to use for new table cell elements. Note: Either specifies an existing tag or creates a new tag.
DefaultImageTagName	string	no	The default name for new image elements created automatically.
DefaultStoryTagName	string	no	The name of the default tag to use for new story elements. Note: Either specifies an existing tag or creates a new tag.
DefaultTableTagName	string	no	The name of the default tag to use for new table elements. Note: Either specifies an existing tag or creates a new tag.

**Table 83. XMLPreference Properties Represented as Elements**

Name	Type	Req	Description
DefaultCellTagColor	InDesignUIColorType	no	The color of the default cell tag, specified either as an array of <code>ListItem</code> elements, each in the range 0 to 255 and representing R, G, and B values, or as a <code>UIColor</code> enumeration. Note: Valid only when default cell tag name value creates a new tag. Does not update the color of an existing tag. .
DefaultImageTagColor	InDesignUIColorType	no	The color to give a new image tag, specified either as an array of <code>ListItem</code> elements, each in the range 0 to 255 and representing R, G, and B values, or as a <code>UIColor</code> enumeration. Note: Used only when the tag needs to be created.
DefaultStoryTagColor	InDesignUIColorType	no	The color of the default story tag, specified either as an array of <code>ListItem</code> elements, each in the range 0 to 255 and representing R, G, and B values, or as a <code>UIColor</code> enumeration. Note: Valid only when default story tag name value creates a new tag. Does not update the color of an existing tag.

Name	Type	Req	Description
DefaultTableTag-Color	InDesignUIColor-Type	no	The color of the default table tag, specified either as an array of <code>ListItem</code> elements, each in the range 0 to 255 and representing R, G, and B values, or as a <code>UIColor</code> enumeration. Note: Valid only when default table tag name value creates a new tag. Does not update the color of an existing tag.

### 6.3.7 ExportForWebPreference

The `<ExportForWebPreference>` element stores the settings that are be used by the Export for Web feature of InDesign. This preference object does not have any effect on the interpretation of layout elements (page items or stories, for example) in the IDML file.

#### Schema Example 76. ExportForWebPreference

```
ExportForWebPreference_Object = element ExportForWebPreference {
    attribute Self { xsd:string },
    attribute CopyFormattedImages { xsd:boolean }?,
    attribute CopyOptimizedImages { xsd:boolean }?,
    attribute CopyOriginalImages { xsd:boolean }?,
    attribute ImageConversion { ImageConversion_EnumValue }?,
    attribute GIFOptionsPalette { GIFOptionsPalette_EnumValue }?,
    attribute GIFOptionsInterlaced { xsd:boolean }?,
    attribute JPEGOptionsQuality { JPEGOptionsQuality_EnumValue }?,
    attribute JPEGOptionsFormat { JPEGOptionsFormat_EnumValue }?
}
```

**Table 84. ExportForWebPreference Properties Represented as Attributes**

Name	Type	Req	Description
CopyFormatted-Images	boolean	no	If true, copies formatted images to the images subfolder.
CopyOptimized-Images	boolean	no	If true, copies optimized images to the images subfolder.
CopyOriginal-Images	boolean	no	If true, copies original images to the images subfolder.
GIFOptions-Interlaced	boolean	no	If true, generates interlaced GIFs. Note: Not valid when image conversion is JPEG.
GIFOptionsPalette	GIFOptions-Palette_EnumValue	no	The color palette for GIF conversion. Note: Not valid when image conversion is JPEG. Can be AdaptivePalette, MacintoshPalette, WebPalette, or WindowsPalette.
ImageConversion	ImageConversion_EnumValue	no	The file format to use for converted images. Note: Valid only when copy optimized images and/or copy formatted images is true. Can be Automatic, JPEG, or GIF.

Name	Type	Req	Description
JPEGOptionsFormat	JPEGOptions-Format_EnumValue	no	The formatting method for converted JPEG images. Note: Not valid when image conversion is GIF. Can be BaselineEncoding or ProgressiveEncoding.
JPEGOptions-Quality	JPEGOptions-Quality_EnumValue	no	The quality of converted JPEG images. Note: Not valid when image conversion is GIF. Can be Low, Medium, High, or Maximum.

### 6.3.8 TransparencyPreference

The <TransparencyPreference> element stores the default transparency settings for the document. Values that you specify here will apply to all transparent objects in the document that do not explicitly define these attributes.

#### Schema Example 77. TransparencyPreference

```
TransparencyPreference_Object = element TransparencyPreference {
    attribute Self { xsd:string },
    attribute BlendingSpace { BlendingSpace_EnumValue }?,
    attribute GlobalLightAngle { xsd:double {minInclusive="-180"
maxInclusive="180"} }?,
    attribute GlobalLightAltitude { xsd:double {minInclusive="0"
maxInclusive="100"} }?
}
```

**Table 85. TransparencyPreference Properties Represented as Attributes**

Name	Type	Req	Description
BlendingSpace	BlendingSpace_EnumValue	no	The transparency blending space. Can be Low, Default, CMYK, or RGB.
GlobalLight-Altitude	double	no	The global light altitude. Range: 0 to 100.
GlobalLightAngle	double	no	The global light angle. Range: -180 to 180.

#### IDML Example 66. TransparencyPreference

```
<TransparencyPreference Self="dTransparencyPreference1" BlendingSpace="CMYK"
GlobalLightAngle="120" GlobalLightAltitude="30"/>
```

### 6.3.9 TransparencyDefaultContainerObject

#### Schema Example 78. TransparencyDefaultContainerObject

```
TransparencyDefaultContainerObject_Object = element TransparencyDefaultContainer-
Object {
(
    TransparencySetting_Object?&
    StrokeTransparencySetting_Object?&
    FillTransparencySetting_Object?&
```

```

        ContentTransparencySetting_Object?
)
}

```

### 6.3.10 StoryPreference

The <StoryPreference> element controls the default story preferences for stories in an InDesign document. Values that you specify here will apply to the story preferences of all stories that do not explicitly define these attributes.

#### Schema Example 79. StoryPreference

```

StoryPreference_Object = element StoryPreference {
    attribute Self { xsd:string },
    attribute OpticalMarginAlignment { xsd:boolean }?,
    attribute OpticalMarginSize { xsd:double {minInclusive="0.1"
maxInclusive="1296"} }?,
    attribute FrameType { FrameTypes_EnumValue }?,
    attribute StoryOrientation { StoryHorizontalOrVertical_EnumValue }?,
    attribute StoryDirection { StoryDirectionOptions_EnumValue }?
}

```

#### IDML Example 67. StoryPreference

```

<StoryPreference Self="dStoryPreference1" OpticalMarginAlignment="false" Optical-
MarginSize="12" FrameType="TextFrameType" StoryOrientation="Horizontal"

StoryDirection="LeftToRightDirection"/>

```

---

**Table 86. StoryPreference Properties Represented as Attributes**

Name	Type	Req	Description
FrameType	FrameTypes_EnumValue	no	The type of text frame. Can be TextFrameType or FrameGridType.
OpticalMargin-Alignment	boolean	no	If true, adjust the position of characters at the edges of the frame to provide a better appearance.
OpticalMarginSize	double	no	The point size used as a base for calculating optical margin alignment. (Range: 0.1 to 1296)
StoryDirection	StoryDirection-Options_EnumValue	no	The direction of the story. Can be Left-ToRightDirection or RightToLeftDirection.
StoryOrientation	StoryHorizontal-OrVertical_EnumValue	no	The direction of the text in the story. Can be Horizontal or Vertical.

### 6.3.11 TextFramePreference

The <TextFramePreferences> element controls the default text frame preferences for text frames in an InDesign document. Values that you specify here will apply to the text frame preferences of all text frames that do not explicitly define these attributes. If you do not define these preferences,

and do not explicitly define them for an element, the corresponding values from the IDML defaults file will be used for that element.

#### **Schema Example 80. TextFramePreference**

```
TextFramePreference_Object = element TextFramePreference {
    attribute Self { xsd:string },
    attribute TextColumnCount { xsd:int {minInclusive="1" maxInclusive="40"} }?,
    attribute TextColumnGutter { xsd:double {minInclusive="0"
maxInclusive="8640"} }?,
    attribute TextColumnFixedWidth { xsd:double {minInclusive="0"
maxInclusive="8640"} }?,
    attribute UseFixedColumnWidth { xsd:boolean }?,
    attribute FirstBaselineOffset { FirstBaseline_EnumValue }?,
    attribute MinimumFirstBaselineOffset { xsd:double {minInclusive="0"
maxInclusive="8640"} }?,
    attribute VerticalJustification { VerticalJustification_EnumValue }?,
    attribute VerticalThreshold { xsd:double {minInclusive="0"
maxInclusive="8640"} }?,
    attribute IgnoreWrap { xsd:boolean }?,
    element Properties {
        element InsetSpacing {
            (unit_type, xsd:double {minInclusive="0" maxInclusive="8640"} ) |
            (list_type,
                element ListItem { unit_type, xsd:double {minInclusive="0"
maxInclusive="8640"} },
                element ListItem { unit_type, xsd:double {minInclusive="0"
maxInclusive="8640"} })
        }?
    }
}?
}
```

#### **IDML Example 68. TextFramePreferences**

```
<TextFramePreference Self="dTextFramePreference1" TextColumnCount="1" TextColumn-
Gutter="12" TextColumnFixedWidth="144" UseFixedColumnWidth="false" FirstBaseline-
Offset="BaselineOffset" MinimumFirstBaselineOffset="0" VerticalJustification="Top-
Align" VerticalThreshold="0" IgnoreWrap="false">
    <Properties>
        <InsetSpacing type="list">
            <ListItem type="unit">0</ListItem>
            <ListItem type="unit">0</ListItem>
            <ListItem type="unit">0</ListItem>
            <ListItem type="unit">0</ListItem>
        </InsetSpacing>
    </Properties>
</TextFramePreference>
```

**Table 87. TextFramePreference Properties Represented as Attributes**

Name	Type	Req	Description
FirstBaseline-Offset	FirstBaseline_EnumValue	no	The distance between the baseline of the text and the top inset of the text frame. Can be AscentOffset, CapHeight, LeadingOffset, EmboxHeight, XHeight, or FixedHeight.
IgnoreWrap	boolean	no	If true, ignores text wrap settings for objects whose text wrap intersects the area of the text frame.
MinimumFirst-BaselineOffset	double	no	The minimum distance between the baseline of the text and the top inset of the text frame. Range 0 to 8640.
TextColumnCount	int	no	The number of columns in the text frame. Note: Depending on the value of use fixed column width, the number of columns can change automatically when the text frame size changes. Range 1 to 40.
TextColumnFixed-Width	double	no	The column width of the columns in the text frame. Range: 0 to 8640.
TextColumnGutter	double	no	The space between columns in the text frame. Range: 0 to 8640.
UseFixedColumn-Width	boolean	no	If true, maintains column width when the text frame is resized. If false, causes columns to resize when the text frame is resized. Note: When true, resizing the frame can change the number of columns in the frame.
Vertical-Justification	Vertical-Justification_EnumValue	no	The vertical alignment of the text content.
VerticalThreshold	double	no	The maximum amount of vertical space between two paragraphs. Note: Valid only when vertical justification is justified; the specified amount is applied in addition to the space before or space after values defined for the paragraph. Range: 0 to 8640.

**Table 88. TextFramePreference Properties Represented as Elements**

Name	Type	Req	Description
InsetSpacing	ListItem or double	no	The text insets applied to the text frame. Rectangular text frames can use four inset values (for the left, top, right, and bottom of the frame); non-rectangular text frames can use a single value (all sides). Can contain up to four ListItem elements. Optional.

### 6.3.12 TextPreference

The <TextPreference> element stores general text preferences for a document.

#### Schema Example 81. TextPreference

```
TextPreference_Object = element TextPreference {
    attribute Self { xsd:string },
    attribute TypographersQuotes { xsd:boolean }?,
    attribute HighlightHjViolations { xsd:boolean }?,
    attribute HighlightKeeps { xsd:boolean }?,
    attribute HighlightSubstitutedGlyphs { xsd:boolean }?,
    attribute HighlightCustomSpacing { xsd:boolean }?,
    attribute HighlightSubstitutedFonts { xsd:boolean }?,
    attribute UseOpticalSize { xsd:boolean }?,
    attribute UseParagraphLeading { xsd:boolean }?,
    attribute SuperscriptSize { xsd:double {minInclusive="1" maxInclusive="200"} }?,
    attribute SuperscriptPosition { xsd:double {minInclusive="-500"
maxInclusive="500"} }?,
    attribute SubscriptSize { xsd:double {minInclusive="1" maxInclusive="200"} }?,
    attribute SubscriptPosition { xsd:double {minInclusive="-500"
maxInclusive="500"} }?,
    attribute SmallCap { xsd:double {minInclusive="1" maxInclusive="200"} }?,
    attribute LeadingKeyIncrement { xsd:double {minInclusive="0.001"
maxInclusive="200"} }?,
    attribute BaselineShiftKeyIncrement { xsd:double {minInclusive="0.001"
maxInclusive="200"} }?,
    attribute KerningKeyIncrement { xsd:double {minInclusive="1"
maxInclusive="100"} }?,
    attribute ShowInvisibles { xsd:boolean }?,
    attribute JustifyTextWraps { xsd:boolean }?,
    attribute AbutTextToTextWrap { xsd:boolean }?,
    attribute ZOrderTextWrap { xsd:boolean }?,
    attribute LinkTextFilesWhenImporting { xsd:boolean }?,
    attribute HighlightKinsoku { xsd:boolean }?,
    attribute UseNewVerticalScaling { xsd:boolean }?,
    attribute UseCidMojikumi { xsd:boolean }?,
    attribute EnableStylePreviewMode { xsd:boolean }?,
    attribute EnableDynamicAutoflow { xsd:boolean }?,
    attribute AutoPageInsertion { AutoPageInsertion_EnumValue }?,
    attribute RestrictToMasterTextFrames { xsd:boolean }?,
    attribute PreserveRectoVerso { xsd:boolean }?,
    attribute AutoPageDeletion { xsd:boolean }?
}
```

**Table 89. TextPreference Properties Represented as Attributes**

Name	Type	Req	Description
AbutTextToText-Wrap	boolean	no	If true, moves wrapped text to the next available leading increment below the text wrap objects.
AutoPageDeletion	boolean	no	If true, deletes pages when changing content results in empty pages.

Name	Type	Req	Description
AutoPageInsertion	AutoPage- Insertion_Enum- Value	no	Controls the addition of pages as content changes. Can be AtEndOfStory, AtEndOfSection, or AtEndOfDocument.
BaselineShiftKey- Increment	double	no	The amount that the baseline shift increases each time the user presses the option/alt-shift-up arrow keys or decreases each time the user presses the option/alt-shift-down arrow keys. (Range: .001 to 100)
EnableDynamic- Autoflow	boolean	no	If true, enable dynamic autoflow.
EnableStyle- PreviewMode	boolean	no	If true, highlights character and paragraph styles with colored backgrounds.
HighlightCustom- Spacing	boolean	no	If true, highlights custom kerned or tracked characters.
HighlightHj- Violations	boolean	no	If true, highlights hyphenation and justification rule violations in the text.
HighlightKeeps	boolean	no	If true, highlights paragraphs that violate keep options.
HighlightKinsoku	boolean	no	If true, uses on-screen highlighting to identify kinsoku.
Highlight- SubstitutedFonts	boolean	no	If true, highlights missing fonts.
Highlight- SubstitutedGlyphs	boolean	no	If true, highlights substituted glyphs.
JustifyTextWraps	boolean	no	If true, justifies text around text wrap objects.
KerningKey- Increment	double	no	The amount the kerning value per 1000 ems increases each time the user presses of the option/alt-right arrow keys or decreases each time the user presses the option/alt-left arrow keys. (Range: 1 to 100)
LeadingKey- Increment	double	no	The amount that leading increases each time the user presses the option/alt-up arrow keys or decreases each time the user presses the option/alt-down arrow keys. (Range: .001 to 200)
LinkTextFiles- WhenImporting	boolean	no	If true, links placed text files and spreadsheet files. If false, embeds the files.
PreserveRecto- Verso	boolean	no	If true, preserve right/left page orientation.
RestrictToMaster- TextFrames	boolean	no	
ShowInvisibles	boolean	no	If true, shows hidden characters.
SmallCap	double	no	The size of text formatted as small caps, specified as a percentage of the font size. (Range: 1 to 200)
SubscriptPosition	double	no	The position of subscript characters, specified as a percentage of the regular leading. (Range: -500 to 500)

Name	Type	Req	Description
SubscriptSize	double	no	The size of subscript characters, specified as a percentage of the font size. (Range: 0 to 200)
Superscript-Position	double	no	The position of superscript characters, specified as a percentageage of the regular leading. (Range: -500 to 500)
SuperscriptSize	double	no	The size of superscript characters, specified as a percentageage of the font size. (Range: 0 to 200)
Typographers-Quotes	boolean	no	If true, converts straight quotes to typographic quotes.
UseCidMojikumi	boolean	no	If true, uses the glyph CID to get the mojikumi class of the character.
UseNewVertical-Scaling	boolean	no	If true, reverses X and Y scaling on Roman characters in vertical text.
UseOpticalSize	boolean	no	If true, automatically selects the correct optical size.
UseParagraph-Leading	boolean	no	If true, applies the leading changes made to a text range to the entire paragraph. If false, applies leading changes only to the text range.
ZOrderTextWrap	boolean	no	If true, text wrap does not affect text on layers above the layer that contains the text wrap object. If false, text wrap affects text on all visible layers.

### 6.3.13 TextDefault

The `<TextDefault>` element controls the default text formatting for an InDesign document. For all text that has been styled with the default “[Basic Paragraph]” style, this formatting is applied as local overrides. Values that you specify here will apply to all text formatted using the “[Basic Paragraph]” style that does not explicitly define these attributes and elements. If you do not define these values, the corresponding values from the IDML defaults file will be used.

#### Schema Example 82. TextDefault

```
TextDefault_Object = element TextDefault {
    attribute Self { xsd:string },
    attribute FirstLineIndent { xsd:double }?,
    attribute LeftIndent { xsd:double }?,
    attribute RightIndent { xsd:double }?,
    attribute SpaceBefore { xsd:double }?,
    attribute SpaceAfter { xsd:double }?,
    attribute Justification { Justification_EnumValue }?,
    attribute SingleWordJustification { SingleWordJustification_EnumValue }?,
    attribute AutoLeading { xsd:double }?,
    attribute DropCapLines { xsd:short {minInclusive="0" maxInclusive="25"} }?,
    attribute DropCapCharacters { xsd:short {minInclusive="0"
maxInclusive="150"} }?,
    attribute KeepLinesTogether { xsd:boolean }?,
    attribute KeepAllLinesTogether { xsd:boolean }?,
    attribute KeepWithNext { xsd:short {minInclusive="0" maxInclusive="5"} }?,
    attribute KeepFirstLines { xsd:short {minInclusive="1" maxInclusive="50"} }?,
```

```

attribute KeepLastLines { xsd:short {minInclusive="1" maxInclusive="50"} }?,
attribute StartParagraph { StartParagraph_EnumValue }?,
attribute Composer { xsd:string }?,
attribute MinimumWordSpacing { xsd:double }?,
attribute MaximumWordSpacing { xsd:double }?,
attribute DesiredWordSpacing { xsd:double }?,
attribute MinimumLetterSpacing { xsd:double }?,
attribute MaximumLetterSpacing { xsd:double }?,
attribute DesiredLetterSpacing { xsd:double }?,
attribute MinimumGlyphScaling { xsd:double }?,
attribute MaximumGlyphScaling { xsd:double }?,
attribute DesiredGlyphScaling { xsd:double }?,
attribute RuleAbove { xsd:boolean }?,
attribute RuleAboveOverprint { xsd:boolean }?,
attribute RuleAboveLineWeight { xsd:double }?,
attribute RuleAboveTint { xsd:double }?,
attribute RuleAboveOffset { xsd:double }?,
attribute RuleAboveLeftIndent { xsd:double }?,
attribute RuleAboveRightIndent { xsd:double }?,
attribute RuleAboveWidth { RuleWidth_EnumValue }?,
attribute RuleAboveGapTint { xsd:double }?,
attribute RuleAboveGapOverprint { xsd:boolean }?,
attribute RuleBelow { xsd:boolean }?,
attribute RuleBelowLineWeight { xsd:double }?,
attribute RuleBelowTint { xsd:double }?,
attribute RuleBelowOffset { xsd:double }?,
attribute RuleBelowLeftIndent { xsd:double }?,
attribute RuleBelowRightIndent { xsd:double }?,
attribute RuleBelowWidth { RuleWidth_EnumValue }?,
attribute RuleBelowGapTint { xsd:double }?,
attribute HyphenateCapitalizedWords { xsd:boolean }?,
attribute Hyphenation { xsd:boolean }?,
attribute HyphenateBeforeLast { xsd:short {minInclusive="1" maxInclusive="15"} }?,
attribute HyphenateAfterFirst { xsd:short {minInclusive="1" maxInclusive="15"} }?,
attribute HyphenateWordsLongerThan { xsd:short {minInclusive="3" maxInclusive="25"} }?,
attribute HyphenateLadderLimit { xsd:short {minInclusive="0" maxInclusive="25"} }?,
attribute HyphenationZone { xsd:double }?,
attribute HyphenWeight { xsd:short {minInclusive="0" maxInclusive="10"} }?,
attribute AppliedParagraphStyle { xsd:string }?,
attribute AppliedCharacterStyle { xsd:string }?,
attribute FontStyle { xsd:string }?,
attribute PointSize { xsd:double }?,
attribute KerningMethod { xsd:string }?,
attribute Tracking { xsd:double }?,
attribute Capitalization { Capitalization_EnumValue }?,
attribute Position { Position_EnumValue }?,
attribute Underline { xsd:boolean }?,
attribute StrikeThru { xsd:boolean }?,
attribute Ligatures { xsd:boolean }?,
attribute NoBreak { xsd:boolean }?,

```

```

attribute HorizontalScale { xsd:double }?,
attribute VerticalScale { xsd:double }?,
attribute BaselineShift { xsd:double }?,
attribute Skew { xsd:double }?,
attribute FillTint { xsd:double }?,
attribute StrokeTint { xsd:double }?,
attribute StrokeWeight { xsd:double }?,
attribute OverprintStroke { xsd:boolean }?,
attribute OverprintFill { xsd:boolean }?,
attribute OTFFigureStyle { OTFFigureStyle_EnumValue }?,
attribute OTFOrdinal { xsd:boolean }?,
attribute OTFFraction { xsd:boolean }?,
attribute OTFDiscretionaryLigature { xsd:boolean }?,
attribute OTFTitling { xsd:boolean }?,
attribute OTFContextualAlternate { xsd:boolean }?,
attribute OTFSwash { xsd:boolean }?,
attribute UnderlineTint { xsd:double }?,
attribute UnderlineGapTint { xsd:double }?,
attribute UnderlineOverprint { xsd:boolean }?,
attribute UnderlineGapOverprint { xsd:boolean }?,
attribute UnderlineOffset { xsd:double }?,
attribute UnderlineWeight { xsd:double }?,
attribute StrikeThroughTint { xsd:double }?,
attribute StrikeThroughGapTint { xsd:double }?,
attribute StrikeThroughOverprint { xsd:boolean }?,
attribute StrikeThroughGapOverprint { xsd:boolean }?,
attribute StrikeThroughOffset { xsd:double }?,
attribute StrikeThroughWeight { xsd:double }?,
attribute FillColor { xsd:string }?,
attribute StrokeColor { xsd:string }?,
attribute AppliedLanguage { xsd:string }?,
attribute LastLineIndent { xsd:double }?,
attribute HyphenateLastWord { xsd:boolean }?,
attribute OTFSlashedZero { xsd:boolean }?,
attribute OTFHistorical { xsd:boolean }?,
attribute OTFStylisticSets { xsd:int }?,
attribute GradientFillLength { xsd:double }?,
attribute GradientFillAngle { xsd:double }?,
attribute GradientStrokeLength { xsd:double }?,
attribute GradientStrokeAngle { xsd:double }?,
attribute GradientFillStart { UnitPointType_TypeDef }?,
attribute GradientStrokeStart { UnitPointType_TypeDef }?,
attribute RuleBelowOverprint { xsd:boolean }?,
attribute RuleBelowGapOverprint { xsd:boolean }?,
attribute DropcapDetail { xsd:int }?,
attribute HyphenateAcrossColumns { xsd:boolean }?,
attribute KeepRuleAboveInFrame { xsd:boolean }?,
attribute IgnoreEdgeAlignment { xsd:boolean }?,
attribute OTFMark { xsd:boolean }?,
attribute OTFLocale { xsd:boolean }?,
attribute PositionalForm { PositionalForms_EnumValue }?,
attribute ParagraphDirection { ParagraphDirection_EnumValue }?,
attribute ParagraphJustification { ParagraphJustification_EnumValue }?,
attribute MiterLimit { xsd:double {minInclusive="1" maxInclusive="500"} }?,

```

```

attribute StrokeAlignment { TextStrokeAlign_EnumValue }?,
attribute EndJoin { OutlineJoin_EnumValue }?,
attribute OTFOverlapSwash { xsd:boolean }?,
attribute OTFStylisticAlternate { xsd:boolean }?,
attribute OTFJustificationAlternate { xsd:boolean }?,
attribute OTFSretchedAlternate { xsd:boolean }?,
attribute CharacterDirection { CharacterDirection_EnumValue }?,
attribute KeyboardDirection { CharacterDirection_EnumValue }?,
attribute DigitsType { DigitsType_EnumValue }?,
attribute Kashidas { Kashidas_EnumValue }?,
attribute DiacriticPosition { DiacriticPosition_EnumValue }?,
attribute XOffsetDiacritic { xsd:double }?,
attribute YOffsetDiacritic { xsd:double }?,
attribute GotoNextX { GotoNextX_EnumValue }?,
attribute PageNumberType { PageNumberType_EnumValue }?,
attribute AppliedNamedGrid { xsd:string }?,
attribute CharacterAlignment { CharacterAlignment_EnumValue }?,
attribute Tsume { xsd:double }?,
attribute LeadingAki { xsd:double }?,
attribute TrailingAki { xsd:double }?,
attribute CharacterRotation { xsd:double }?,
attribute Jidori { xsd:short }?,
attribute ShataiMagnification { xsd:double }?,
attribute ShataiDegreeAngle { xsd:double }?,
attribute ShataiAdjustRotation { xsd:boolean }?,
attribute ShataiAdjustTsume { xsd:boolean }?,
attribute Tatechuyoko { xsd:boolean }?,
attribute TatechuyokoXOffset { xsd:double }?,
attribute TatechuyokoYOffset { xsd:double }?,
attribute KentenTint { xsd:double }?,
attribute KentenStrokeTint { xsd:double }?,
attribute KentenWeight { xsd:double }?,
attribute KentenOverprintFill { AdornmentOverprint_EnumValue }?,
attribute KentenOverprintStroke { AdornmentOverprint_EnumValue }?,
attribute KentenKind { KentenCharacter_EnumValue }?,
attribute KentenPlacement { xsd:double }?,
attribute KentenAlignment { KentenAlignment_EnumValue }?,
attribute KentenPosition { RubyKentenPosition_EnumValue }?,
attribute KentenFontSize { xsd:double }?,
attribute KentenXScale { xsd:double }?,
attribute KentenYScale { xsd:double }?,
attribute KentenCustomCharacter { xsd:string }?,
attribute KentenCharacterSet { KentenCharacterSet_EnumValue }?,
attribute RubyTint { xsd:double }?,
attribute RubyWeight { xsd:double }?,
attribute RubyOverprintFill { AdornmentOverprint_EnumValue }?,
attribute RubyOverprintStroke { AdornmentOverprint_EnumValue }?,
attribute RubyStrokeTint { xsd:double }?,
attribute RubyFontSize { xsd:double }?,
attribute RubyOpenTypePro { xsd:boolean }?,
attribute RubyXScale { xsd:double }?,
attribute RubyYScale { xsd:double }?,
attribute RubyType { RubyTypes_EnumValue }?,
attribute RubyAlignment { RubyAlignments_EnumValue }?,

```

```

        attribute RubyPosition { RubyKetenPosition_EnumValue }?,
        attribute RubyXOffset { xsd:double }?,
        attribute RubyYOffset { xsd:double }?,
        attribute RubyParentSpacing { RubyParentSpacing_EnumValue }?,
        attribute RubyAutoAlign { xsd:boolean }?,
        attribute RubyOverhang { xsd:boolean }?,
        attribute RubyAutoScaling { xsd:boolean }?,
        attribute RubyParentScalingPercent { xsd:double }?,
        attribute RubyParentOverhangAmount { RubyOverhang_EnumValue }?,
        attribute Warichu { xsd:boolean }?,
        attribute WarichuSize { xsd:double }?,
        attribute WarichuLines { xsd:short }?,
        attribute WarichuLineSpacing { xsd:double }?,
        attribute WarichuAlignment { WarichuAlignment_EnumValue }?,
        attribute WarichuCharsAfterBreak { xsd:short }?,
        attribute WarichuCharsBeforeBreak { xsd:short }?,
        attribute OTFProportionalMetrics { xsd:boolean }?,
        attribute OTFHVKana { xsd:boolean }?,
        attribute OTFRomanItalics { xsd:boolean }?,
        attribute ScaleAffectsLineHeight { xsd:boolean }?,
        attribute CjkGridTracking { xsd:boolean }?,
        attribute GlyphForm { AlternateGlyphForms_EnumValue }?,
        attribute GridAlignFirstLineOnly { xsd:boolean }?,
        attribute GridAlignment { GridAlignment_EnumValue }?,
        attribute GridGyoudori { xsd:short }?,
        attribute AutoTcy { xsd:short }?,
        attribute AutoTcyIncludeRoman { xsd:boolean }?,
        attribute KinsokuType { KinsokuType_EnumValue }?,
        attribute KinsokuHangType { KinsokuHangTypes_EnumValue }?,
        attribute BunriKinshi { xsd:boolean }?,
        attribute Rensuuji { xsd:boolean }?,
        attribute RotateSingleByteCharacters { xsd:boolean }?,
        attribute LeadingModel { LeadingModel_EnumValue }?,
        attribute RubyAutoTcyDigits { xsd:short }?,
        attribute RubyAutoTcyIncludeRoman { xsd:boolean }?,
        attribute RubyAutoTcyAutoScale { xsd:boolean }?,
        attribute TreatIdeographicSpaceAsSpace { xsd:boolean }?,
        attribute AllowArbitraryHyphenation { xsd:boolean }?,
        attribute ParagraphGyoudori { xsd:boolean }?,
        attribute NumberingExpression { xsd:string }?,
        attribute BulletsTextAfter { xsd:string }?,
        attribute NumberingLevel { xsd:int }?,
        attribute NumberingContinue { xsd:boolean }?,
        attribute NumberingStartAt { xsd:int }?,
        attribute NumberingApplyRestartPolicy { xsd:boolean }?,
        attribute BulletsAlignment { ListAlignment_EnumValue }?,
        attribute NumberingAlignment { ListAlignment_EnumValue }?,
        attribute BulletsAndNumberingListType { ListType_EnumValue }?,
        element Properties {
            element BalanceRaggedLines {
                (bool_type, xsd:boolean ) |
                (enum_type, BalanceLinesStyle_EnumValue )
            }?&
            element RuleAboveColor {

```

```

(object_type, xsd:string) |
(string_type, xsd:string)
} ?&
element RuleAboveGapColor {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element RuleAboveType {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element RuleBelowColor {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element RuleBelowGapColor {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element RuleBelowType {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element AllNestedStyles { list_type, element ListItem {
  record_type,
  (
    element AppliedCharacterStyle { object_type, xsd:string } &
    element Delimiter {
      (string_type, xsd:string) |
      (enum_type, NestedStyleDelimiters_EnumValue)
    } &
    element Repetition { long_type, xsd:int } &
    element Inclusive { bool_type, xsd:boolean })
  } *
} ?&
element TabList { list_type, element ListItem {
  record_type,
  (
    element Alignment { enum_type, TabStopAlignment_EnumValue } &
    element AlignmentCharacter { string_type, xsd:string } &
    element Leader { string_type, xsd:string } &
    element Position { unit_type, xsd:double })
  } *
} ?&
element AppliedFont {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element Leading {
  (unit_type, xsd:double) |
  (enum_type, Leading_EnumValue)
} ?&
element UnderlineColor {
  (object_type, xsd:string) |

```

```

        (string_type, xsd:string )
    }?&
    element UnderlineGapColor {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element UnderlineType {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element StrikeThroughColor {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element StrikeThroughGapColor {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element StrikeThroughType {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element KentenFillColor {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element KentenStrokeColor {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element KentenFont {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element KentenFontStyle {
        (string_type, xsd:string ) |
        (enum_type, NothingEnum_EnumValue )
    }?&
    element RubyFill {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element RubyStroke {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element RubyFont {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element RubyFontStyle {
        (string_type, xsd:string ) |
        (enum_type, NothingEnum_EnumValue )
    }?&

```

```

element KinsokuSet {
    (object_type, xsd:string) |
    (enum_type, KinsokuSet_EnumValue) |
    (string_type, xsd:string)
} ?&
element Mojikumi {
    (object_type, xsd:string) |
    (string_type, xsd:string) |
    (enum_type, MojikumiTableDefaults_EnumValue)
} ?&
element BulletsFont {
    (object_type, xsd:string) |
    (string_type, xsd:string) |
    (enum_type, AutoEnum_EnumValue)
} ?&
element BulletsFontStyle {
    (string_type, xsd:string) |
    (enum_type, NothingEnum_EnumValue) |
    (enum_type, AutoEnum_EnumValue)
} ?&
element BulletsCharacterStyle {
    (object_type, xsd:string) |
    (string_type, xsd:string)
} ?&
element NumberingCharacterStyle {
    (object_type, xsd:string) |
    (string_type, xsd:string)
} ?&
element AppliedNumberingList {
    (object_type, xsd:string) |
    (string_type, xsd:string)
} ?&
element NumberingFormat {
    (enum_type, NumberingStyle_EnumValue) |
    (string_type, xsd:string)
} ?&
element NumberingRestartPolicies {
    attribute RestartPolicy { RestartPolicy_EnumValue },
    attribute LowerLevel { xsd:int },
    attribute UpperLevel { xsd:int }
} ?&
element BulletChar {
    attribute BulletCharacterType { BulletCharacterType_EnumValue },
    attribute BulletCharacterValue { xsd:int }
} ?
}
?
'
(
    NestedLineStyle_Object*&
    NestedGrepStyle_Object*
)
}

```

The <TextDefaults> element shares all of its attributes and elements with other text objects. Refer to “Common Text Elements.”

### 6.3.14 DictionaryPreference

The <DictionaryPreference> element defines the language dictionaries used for spelling and hyphenation in a document.

#### Schema Example 83. DictionaryPreference

```
DictionaryPreference_Object = element DictionaryPreference {
    attribute Self { xsd:string },
    attribute Composition { ComposeUsing_EnumValue }?,
    attribute MergeUserDictionary { xsd:boolean }?,
    attribute RecomposeWhenChanged { xsd:boolean }?
}
```

### 6.3.15 AnchoredObjectDefault

The <AnchoredObjectDefault> element controls the default formatting applied to anchored objects in an InDesign document. Values that you specify here will apply to all anchored objects that do not explicitly define these attributes.

#### Schema Example 84. AnchoredObjectDefault

```
AnchoredObjectDefault_Object = element AnchoredObjectDefault {
    attribute Self { xsd:string },
    attribute AnchorContent { ContentType_EnumValue }?,
    attribute InitialAnchorHeight { xsd:double }?,
    attribute InitialAnchorWidth { xsd:double }?,
    attribute AnchoredParagraphStyle { xsd:string }?,
    attribute AnchoredObjectStyle { xsd:string }?
}
```

#### IDML Example 69. AnchoredObjectDefault

```
<AnchoredObjectDefault Self="dAnchoredObjectDefault1" AnchorContent="Unassigned"
InitialAnchorHeight="72" InitialAnchorWidth="72" AnchoredParagraph-
Style="ParagraphStyle\k[No paragraph style]" AnchoredObjectStyle="ObjectStyle\
k[None]"/>
```

**Table 90. AnchoredObjectDefault Properties Represented as Attributes**

Name	Type	Req	Description
AnchorContent	ContentType_EnumValue	no	The initial frame type of a new anchored object. Can be Unassigned, GraphicType, or TextType.
AnchoredObject-Style	string	no	The initial object style of a new anchored object.
Anchored-ParagraphStyle	string	no	The initial paragraph style of a new anchored object. Note: Valid when anchor content is TextType.

Name	Type	Req	Description
InitialAnchorHeight	double	no	The initial height of a new anchored object.
InitialAnchorWidth	double	no	The initial width of a new anchored object.

### 6.3.16 AnchoredObjectSetting

The `<AnchoredObjectSetting>` element controls the default positioning for anchored objects in an InDesign document. Values that you specify here will apply to all anchored objects that do not explicitly define these attributes.

#### Schema Example 85. AnchoredObjectSetting

```
AnchoredObjectSetting_Object = element AnchoredObjectSetting {
    attribute Self { xsd:string },
    attribute AnchoredPosition { AnchorPosition_EnumValue }?,
    attribute SpineRelative { xsd:boolean }?,
    attribute LockPosition { xsd:boolean }?,
    attribute PinPosition { xsd:boolean }?,
    attribute AnchorPoint { AnchorPoint_EnumValue }?,
    attribute HorizontalAlignment { HorizontalAlignment_EnumValue }?,
    attribute HorizontalReferencePoint { AnchoredRelativeTo_EnumValue }?,
    attribute VerticalAlignment { VerticalAlignment_EnumValue }?,
    attribute VerticalReferencePoint { VerticallyRelativeTo_EnumValue }?,
    attribute AnchorXoffset { xsd:double }?,
    attribute AnchorYoffset { xsd:double }?,
    attribute AnchorSpaceAbove { xsd:double }?
}
```

#### IDML Example 70. AnchoredObjectSetting

```
<AnchoredObjectSetting Self="dAnchoredObjectSetting1" AnchoredPosition="Inline-Position" SpineRelative="false" LockPosition="false" PinPosition="true" AnchorPoint="BottomRightAnchor" HorizontalAlignment="LeftAlign" HorizontalReferencePoint="TextFrame" VerticalAlignment="TopAlign" VerticalReferencePoint="Line-Baseline" AnchorXoffset="0" AnchorYoffset="0" AnchorSpaceAbove="0"/>
```

**Table 91. AnchoredObjectSetting Properties Represented as Attributes**

Name	Type	Req	Description
AnchorPoint	AnchorPoint_EnumValue	no	The point in the anchored object to position. Can be TopLeftAnchor, TopCenterAnchor, TopRightAnchor, LeftCenterAnchor, CenterAnchor, RightCenterAnchor, BottomLeftAnchor, BottomCenterAnchor, or BottomRightAnchor.
AnchorSpaceAbove	double	no	The space above an above-line anchored object.
AnchorXoffset	double	no	The horizontal (x) offset of the anchored object.
AnchorYoffset	double	no	The vertical (y) offset of the anchored object.

Name	Type	Req	Description
AnchoredPosition	AnchorPosition_EnumValue	no	The position of the anchored object relative to the anchor. Can be InlinePosition, AboveLine, Anchored.
Horizontal-Alignment	Horizontal-Alignment_EnumValue	no	When anchored position is above line, the position of the anchored object is relative to the text area. When anchored position is custom, the horizontal alignment of the anchored object is set by the horizontal reference point. Note: Not valid when anchored position is InlinePosition. Can be RightAlign, LeftAlign, CenterAlign, TextAlign
Horizontal-ReferencePoint	AnchoredRelativeTo_EnumValue	no	The horizontal reference point on the page. Can be ColumnEdge, TextFrame, PageMargins, PageEdge, AnchorLocation
LockPosition	boolean	no	If true, prevents manual positioning of the anchored object.
PinPosition	boolean	no	If true, pins the position of the anchored object within the text frame top and bottom.
SpineRelative	boolean	no	If true, the position of the anchored object is relative to the binding spine of the page or spread.
VerticalAlignment	Vertical-Alignment_EnumValue	no	The vertical alignment of the anchored object reference point with the vertical reference point on the page. Can be TopAlign, BottomAlign, or CenterAlign.
Vertical-ReferencePoint	Vertically-RelativeTo_EnumValue	no	The vertical reference point on the page. Can be ColumnEdge, TextFrame, PageMargins, PageEdge, LineBaseline, LineXheight, LineAscent, Capheight, TopOfLeading.

### 6.3.17 BaselineFrameGridOption

The <BaselineFrameGridOption> element controls the default formatting of baseline frame grids in an InDesign document. Values that you specify here will apply to all baseline frame grids that do not explicitly define these attributes.

#### Schema Example 86. BaselineFrameGridOption

```
BaselineFrameGridOption_Object = element BaselineFrameGridOption {
    attribute Self { xsd:string },
    attribute UseCustomBaselineFrameGrid { xsd:boolean }?,
    attribute StartingOffsetForBaselineFrameGrid { xsd:double {minInclusive="0"
maxInclusive="8640"} }?,
    attribute BaselineFrameGridRelativeOption {
        BaselineFrameGridRelativeOption_EnumValue }?,
    attribute BaselineFrameGridIncrement { xsd:double {minInclusive="1"
maxInclusive="8640"} }?,
    element Properties {
        element BaselineFrameGridColor { InDesignUIColorType_TypeDef }?
    }
??
}
```

**IDML Example 71. BaselineFrameGridOption**

```
<BaselineFrameGridOption Self="dBaselineFrameGridOption1"
UseCustomBaselineFrameGrid="false" StartingOffsetForBaselineFrameGrid="0"
BaselineFrameGridRelativeOption="TopOfInset" BaselineFrameGridIncrement="12">
<Properties>
<BaselineFrameGridColor type="enumeration">LightBlue
</BaselineFrameGridColor>
</Properties>
</BaselineFrameGridOption>
```

**Table 92. BaselineFrameGridOption Properties Represented as Attributes**

Name	Type	Req	Description
BaselineFrame-GridIncrement	double	no	The distance between grid lines. Range 1 to 8640.
BaselineFrame-GridRelative-Option	BaselineFrame-GridRelative-Option_EnumValue	no	The object from which to offset the custom baseline grid. Can be TopOfPage, TopOfMargin, TopOfFrame, or TopOfInset.
StartingOffset-ForBaselineFrame-Grid	double	no	The amount to offset the baseline grid. Range 1 to 8640.
UseCustom-BaselineFrameGrid	boolean	no	If true, uses a custom baseline frame grid.

**Table 93. BaselineFrameGridOption Properties Represented as Elements**

Name	Type	Req	Description
BaselineFrame-GridColor	InDesignUIColor-Type	no	The color of the baseline frame grid, as a UIColors enumeration or an RGB color as a list of three <ListItem> elements.

**6.3.18 FootnoteOption**

The <FootnoteOption> element controls the default formatting of footnotes in an InDesign document. Values that you specify here will apply to all footnotes that do not explicitly define these attributes and properties.

**Schema Example 87. FootnoteOption**

```
FootnoteOption_Object = element FootnoteOption {
    attribute Self { xsd:string },
    attribute StartAt { xsd:int {minInclusive="1" maxInclusive="100000"} }?,
    attribute Prefix { xsd:string }?,
    attribute Suffix { xsd:string }?,
    attribute FootnoteTextStyle { xsd:string }?,
    attribute FootnoteMarkerStyle { xsd:string }?,
    attribute SeparatorText { xsd:string }?,
    attribute SpaceBetween { xsd:double {minInclusive="0" maxInclusive="864"} }?,
    attribute Spacer { xsd:double {minInclusive="0" maxInclusive="864"} }?,
    attribute FootnoteFirstBaselineOffset { FootnoteFirstBaseline_EnumValue }?,
    attribute FootnoteMinimumFirstBaselineOffset { xsd:double {minInclusive="0" }
```

```

maxInclusive="103680"} }?,
attribute EosPlacement { xsd:boolean }?,
attribute NoSplitting { xsd:boolean }?,
attribute RuleOn { xsd:boolean }?,
attribute RuleLineWeight { xsd:double {minInclusive="0" maxInclusive="1000"} }?,
attribute RuleTint { xsd:double {minInclusive="0" maxInclusive="100"} }?,
attribute RuleGapTint { xsd:double {minInclusive="0" maxInclusive="100"} }?,
attribute RuleGapOverprint { xsd:boolean }?,
attribute RuleOverprint { xsd:boolean }?,
attribute RuleLeftIndent { xsd:double {minInclusive="-103680"
maxInclusive="103680"} }?,
attribute RuleWidth { xsd:double {minInclusive="0" maxInclusive="103680"} }?,
attribute RuleOffset { xsd:double {minInclusive="-15552"
maxInclusive="15552"} }?,
attribute ContinuingRuleOn { xsd:boolean }?,
attribute ContinuingRuleLineWeight { xsd:double {minInclusive="0"
maxInclusive="1000"} }?,
attribute ContinuingRuleTint { xsd:double {minInclusive="0"
maxInclusive="100"} }?,
attribute ContinuingRuleGapTint { xsd:double {minInclusive="0"
maxInclusive="100"} }?,
attribute ContinuingRuleOverprint { xsd:boolean }?,
attribute ContinuingRuleGapOverprint { xsd:boolean }?,
attribute ContinuingRuleLeftIndent { xsd:double {minInclusive="-103680"
maxInclusive="103680"} }?,
attribute ContinuingRuleWidth { xsd:double {minInclusive="0"
maxInclusive="103680"} }?,
attribute ContinuingRuleOffset { xsd:double {minInclusive="-15552"
maxInclusive="15552"} }?,
element Properties {
    element FootnoteNumberingStyle {
        (enum_type, FootnoteNumberingStyle_EnumValue ) |
        (string_type, xsd:string )
    }?&
    element RestartNumbering {
        (enum_type, FootnoteRestarting_EnumValue ) |
        (string_type, xsd:string )
    }?&
    element ShowPrefixSuffix {
        (enum_type, FootnotePrefixSuffix_EnumValue ) |
        (string_type, xsd:string )
    }?&
    element MarkerPositioning {
        (enum_type, FootnoteMarkerPositioning_EnumValue ) |
        (string_type, xsd:string )
    }?&
    element RuleType {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element RuleColor {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
}

```

```

element RuleGapColor {
    (object_type, xsd:string) |
    (string_type, xsd:string)
} ?&
element ContinuingRuleType {
    (object_type, xsd:string) |
    (string_type, xsd:string)
} ?&
element ContinuingRuleColor {
    (object_type, xsd:string) |
    (string_type, xsd:string)
} ?&
element ContinuingRuleGapColor {
    (object_type, xsd:string) |
    (string_type, xsd:string)
} ?
}
?
}

```

**Table 94. FootnoteOption Properties Represented as Attributes**

Name	Type	Req	Description
ContinuingRule-GapOverprint	boolean	no	If true, overprints the gap color of the rule above continued footnote text. Note: Valid when continuing rule type is not solid.
ContinuingRule-GapTint	double	no	The tint (as a percentage) of the gap color of the rule above continued footnote text. (Range: 0 to 100) Note: Valid when continuing rule type is not solid.
ContinuingRule-LeftIndent	double	no	The amount to left indent the rule above continued footnote text. Note: Valid when continuing rule on is true. Range: -103680 to 103680.
ContinuingRule-LineWeight	double	no	The stroke weight of the rule above continued footnote text. (Range: 0 to 1000) Note: Valid when continuing rule on is true.
ContinuingRule-Offset	double	no	The vertical offset of the rule above continued footnote text. Note: Valid when continuing rule on is true. Range: -15552 to 15552.
ContinuingRuleOn	boolean	no	If true, draws a rule above footnote text that continues from a previous column. Note: Valid when no splitting is false or undefined.
ContinuingRule-Overprint	boolean	no	If true, overprints the rule above continued footnote text. Note: Valid when continuing rule on is true.
ContinuingRule-Tint	double	no	The tint (as a percentage) of the rule above continued footnote text. (Range: 0 to 100) Note: Valid when continuing rule type is not solid.
ContinuingRule-Width	double	no	The length of the rule above continued footnote text. Note: Valid when continuing rule on is true. Range: 0 to 103680.

Name	Type	Req	Description
EosPlacement	boolean	no	If true, footnotes at the end of the story are placed just below the text. If false, footnotes at the end of the story are placed at the bottom of the column.
FootnoteFirst-BaselineOffset	FootnoteFirst-Baseline_Enum-Value	no	The distance between the top of the footnote container and the footnote text. Can be AscentOffset, CapHeight, LeadingOffset, EmboxHeight, XHeight, or FixedHeight.
FootnoteMarker-Style	string	no	The character style to apply to footnote reference numbers in the main text.
FootnoteMinimum-FirstBaseline-Offset	double	no	The minimum distance between the baseline of the text and the top of the footnote container. Range: 0 to 103680.
FootnoteTextStyle	string	no	The paragraph style to apply to footnotes. Note: The space before and after the paragraph defined in the paragraph style is ignored for footnotes. To define space above and between footnotes, see spacer and space between.
NoSplitting	boolean	no	If true, footnotes cannot split across columns. If false, footnotes flow into succeeding columns when the footnote text causes the footnote area to expand upward to reach the footnote reference number in the main text.
Prefix	string	no	The prefix text of the footnote. (Limit: 0 to 100 characters)
RuleGapOverprint	boolean	no	If true, overprints the gap color of the rule above the first footnote in the column. Note: Valid when rule type is not solid.
RuleGapTint	double	no	The tint (as a percentage) of the gap color of the rule above the first footnote in the column. (Range: 0 to 100) Note: Valid when rule type is not solid.
RuleLeftIndent	double	no	The amount to left indent the rule above the first footnote in the column. Note: Valid when rule on is true. Range: -103860 to 103860.
RuleLineWidth	double	no	The stroke weight of the rule above the first footnote in the column. (Range: 0 to 1000) Note: Valid when rule on is true.
RuleOffset	double	no	The vertical offset of the rule above the first footnote in the column. Note: Valid when rule on is true. Range -15552 to 15552.
RuleOn	boolean	no	If true, draws a rule between the text and the first footnote in the column.
RuleOverprint	boolean	no	If true, overprints the rule above the first footnote in the column. Note: Valid when rule on is true.
RuleTint	double	no	The tint (as a percentage) of the rule above the first footnote in the column. (Range: 0 to 100) Note: Valid when rule on is true.

Name	Type	Req	Description
RuleWidth	double	no	The length of the rule above the first footnote in the column. Note: Valid when rule on is true. Range: 0 to 103680.
Self	string	yes	The unique ID of the object.
SeparatorText	string	no	The text to insert between the footnote marker number and the footnote text. (Range: 0 to 100 characters)
SpaceBetween	double	no	The amount of vertical space between footnotes. Note: The space before and space after defined for the paragraph style applied to the footnote is ignored. For information on the applied paragraph style, see footnote text style. Range: 0 to 864.
Spacer	double	no	The minimum amount of vertical space between the bottom of the text column and the first footnote. Note: The space before amount defined in the paragraph style applied to the footnote is ignored for the first footnote. For information on the applied paragraph style, see footnote text style. Range: 0 to 864.
StartAt	int	no	The number at which to start footnote numbering. Range: 1 to 100000.
Suffix	string	no	The suffix text of the footnote. (Limit: 0 to 100 characters)

**Table 95. FootnoteOption Properties Represented as Elements**

Name	Type	Req	Description
ContinuingRule-Color	string	no	A reference to the swatch (color, gradient, tint, or mixed ink) applied to the stroke of the footnote continuing rule above. The string can contain either an element reference (the value of the Self attribute of the swatch), or the name of the color (for example: Color/Black). Note: Valid when continuing rule on is true.
ContinuingRule-GapColor	string	no	The swatch (color, gradient, tint, or mixed ink) applied to the stroke gap of the footnote continuing rule above. The string can contain either an element reference (the value of the Self attribute of the swatch), or the name of the color (for example: Color/Black). Note: Valid when continuing rule on is true.
ContinuingRule-Type	string	no	The stroke type of the rule above continued footnote text. Note: Valid when continuing rule on is true.
Footnote-NumberingStyle	FootnoteNumberingStyle_Enum-Value or string	no	The footnote numbering style.

Name	Type	Req	Description
MarkerPositioning	FootnoteMarker-Positioning_Enum-Value or string	no	The position of footnote reference numbers in the main text.
RestartNumbering	Footnote-Restarting_Enum-Value or string	no	The point at which to restart footnote numbering.
RuleColor	string	no	The swatch (color, gradient, tint, or mixed ink) applied to the stroke of the rule above the first footnote in the column. The string can contain either an element reference (the value of the Self attribute of the swatch), or the name of the color (for example: Color/Black). Note: Valid when rule on is true.
RuleGapColor	string	no	The swatch (color, gradient, tint, or mixed ink) applied to the stroke gap of the rule above the first footnote in the column. The string can contain either an element reference (the value of the Self attribute of the swatch), or the name of the color (for example: Color/Black). Note: Valid when rule type is not solid.
RuleType	string	no	The stroke type of the rule above the first footnote in a column. Note: Valid when rule on is true.
ShowPrefixSuffix	FootnotePrefix-Suffix_EnumValue or string	no	The position of the footnote prefix and/or suffix.

**IDML Example 72. FootnoteOptions**

```

<FootnoteOption Self="dFootnoteOption1" StartAt="1" Prefix="" Suffix="" Footnote-TextStyle="ParagraphStyle\kNormalParagraphStyle" FootnoteMarkerStyle="Character-Style\k[No character style]" SeparatorText="#" SpaceBetween="0" Spacer="0" FootnoteFirstBaselineOffset="LeadingOffset" FootnoteMinimumFirstBaselineOffset="0" EosPlacement="false" NoSplitting="false" RuleOn="true" RuleLineWeight="1" Rule-Tint="100" RuleGapTint="100" RuleGapOverprint="false" RuleOverprint="false" Rule-LeftIndent="0" RuleWidth="72" RuleOffset="0" ContinuingRuleOn="true" Continuing-RuleLineWeight="1" ContinuingRuleTint="100" ContinuingRuleGapTint="100" ContinuingRuleOverprint="false" ContinuingRuleGapOverprint="false" ContinuingRule-LeftIndent="0" ContinuingRuleWidth="288" ContinuingRuleOffset="0">
  <Properties>
    <FootnoteNumberingStyle type="enumeration">Arabic</FootnoteNumberingStyle>
    <RestartNumbering type="enumeration">DontRestart</RestartNumbering>
    <ShowPrefixSuffix type="enumeration">NoPrefixSuffix</ShowPrefixSuffix>
    <MarkerPositioning type="enumeration">SuperscriptMarker</MarkerPositioning>
    <RuleType type="object">StrokeStyle\kSolid</RuleType>
    <RuleColor type="object">Color\cBlack</RuleColor>
    <RuleGapColor type="object">Swatch\cNone</RuleGapColor>
    <ContinuingRuleType type="object">StrokeStyle\kSolid</ContinuingRuleType>
    <ContinuingRuleColor type="object">Color\cBlack</ContinuingRuleColor>
    <ContinuingRuleGapColor type="object">Swatch\cNone</ContinuingRuleGapColor>
  </Properties>
</FootnoteOption>

```

### 6.3.19 TextWrapPreference

The <TextWrapPreference> element controls the default text wrap applied to page items in an InDesign document. Values that you specify here will apply to all text wraps that do not explicitly define these attributes and elements.

#### Schema Example 88. TextWrapPreference

```
TextWrapPreference_Object = element TextWrapPreference {
    attribute TextWrapType { TextWrapTypes_EnumValue }?,
    attribute Inverse { xsd:boolean }?,
    attribute ApplyToMasterPageOnly { xsd:boolean }?,
    attribute TextWrapSide { TextWrapSideOptions_EnumValue }?,
    attribute ItemTransform { TransformationMatrixType_TypeDef }?,
    element Properties {
        element TextWrapOffset { UnitRectangleBoundsType_TypeDef }?&
        element PathGeometry { element GeometryPathType { GeometryPathType_TypeDef }* }?
    }
    ?
    '
    (
        ContourOption_Object?
    )
}
```

#### Schema Example 89. ContourOption

```
ContourOption_Object = element ContourOption {
    attribute Self { xsd:string },
    attribute ContourType { ContourOptionsTypes_EnumValue }?,
    attribute IncludeInsideEdges { xsd:boolean }?,
    attribute ContourPathName { xsd:string }?
}
```

#### IDML Example 73. TextWrapPreference

```
<TextWrapPreference Self="dTextWrapPreference1" TextWrapType="None"
Inverse="false" ApplyToMasterPageOnly="false" TextWrapSide="BothSides">
<Properties>
    <TextWrapOffset Top="0" Left="0" Bottom="0" Right="0"/>
</Properties>
<ContourOption Self="dTextWrapPreference1ContourOption1" ContourType="SameAs-
Clipping" IncludeInsideEdges="false" ContourPathName="$ID/" />
</TextWrapPreference>
```

**Table 96. ContourOption Properties Represented as Attributes**

Name	Type	Req	Description
ContourPathName	string	no	The name of the alpha channel or Photoshop path to use for the contour option. Valid only when the contour options is photoshop path or alpha channel.

Name	Type	Req	Description
ContourType	ContourOptions-Types_EnumValue	no	The contour type. Can be BoundingBox, PhotoshopPath, DetectEdges, Alpha-Channel, GraphicFrame, or SameAs-Clipping.
IncludeInside-Edges	boolean	no	If true, creates interior clipping paths within the surrounding clipping path. Note: Valid only when clipping type is AlphaChannel or DetectEdges.

### 6.3.20 DocumentPreference

The <DocumentPreference> element contains various preferences for the document. Document preferences define the basic layout of the document. If you omit some or all of the properties set in the <DocumentPreferences> element, InDesign will apply its application default value to the missing property as it opens the IDML file.

#### Schema Example 90. DocumentPreference

```
DocumentPreference_Object = element DocumentPreference {
    attribute Self { xsd:string },
    attribute PageHeight { xsd:double }?,
    attribute PageWidth { xsd:double }?,
    attribute PagesPerDocument { xsd:int }?,
    attribute FacingPages { xsd:boolean }?,
    attribute DocumentBleedTopOffset { xsd:double }?,
    attribute DocumentBleedBottomOffset { xsd:double }?,
    attribute DocumentBleedInsideOrLeftOffset { xsd:double }?,
    attribute DocumentBleedOutsideOrRightOffset { xsd:double }?,
    attribute DocumentBleedUniformSize { xsd:boolean }?,
    attribute SlugTopOffset { xsd:double }?,
    attribute SlugBottomOffset { xsd:double }?,
    attribute SlugInsideOrLeftOffset { xsd:double }?,
    attribute SlugRightOrOutsideOffset { xsd:double }?,
    attribute DocumentSlugUniformSize { xsd:boolean }?,
    attribute PreserveLayoutWhenShuffling { xsd:boolean }?,
    attribute AllowPageShuffle { xsd:boolean }?,
    attribute OverprintBlack { xsd:boolean }?,
    attribute PageBinding { PageBindingOptions_EnumValue }?,
    attribute ColumnDirection { HorizontalOrVertical_EnumValue }?,
    attribute ColumnGuideLocked { xsd:boolean }?,
    attribute MasterTextFrame { xsd:boolean }?,
    attribute SnippetImportUsesOriginalLocation { xsd:boolean }?,
    element Properties {
        element ColumnGuideColor { InDesignUIColorType_TypeDef }?&
        element MarginGuideColor { InDesignUIColorType_TypeDef }?
    }
}
```

#### IDML Example 74. DocumentPreferences

```
<DocumentPreference Self="dDocumentPreference1" PageHeight="792" PageWidth="612"
PagesPerDocument="1" FacingPages="true" DocumentBleedTopOffset="0" DocumentBleed-
```

```

BottomOffset="0" DocumentBleedInsideOrLeftOffset="0" DocumentBleedOutsideOrRight-
Offset="0" DocumentBleedUniformSize="true" SlugTopOffset="0" SlugBottomOffset="0"
SlugInsideOrLeftOffset="0" SlugRightOrOutsideOffset="0" DocumentSlugUniform-
Size="false" PreserveLayoutWhenShuffling="true" AllowPageShuffle="true" Overprint-
Black="true" PageBinding="LeftToRight" ColumnDirection="Horizontal" ColumnGuide-
Locked="true" MasterTextFrame="false" SnippetImportUsesOriginalLocation="false">
<Properties>
    <ColumnGuideColor type="enumeration">Violet</ColumnGuideColor>
    <MarginGuideColor type="enumeration">Magenta</MarginGuideColor>
</Properties>
</DocumentPreference>

```

**Table 97. DocumentPreference Properties Represented as Attributes**

Name	Type	Req	Description
AllowPageShuffle	boolean	no	If true, guarantees that all new spreads added to the document contain a maximum of two pages. If false, allows pages to be added or moved into existing spreads. For override information, see preserve layout when shuffling.
ColumnDirection	HorizontalOr- Vertical_Enum- Value	yes	The direction of text in the column
ColumnGuideLocked	boolean	no	If true, locks column guides.
DocumentBleed- BottomOffset	double	no	The amount to offset the bottom document bleed. Note: To set the bleed bottom offset, document bleed uniform size must be false.
DocumentBleed- InsideOrLeft- Offset	double	no	The amount to offset the inside or left document bleed. Note: To set the bleed inside or left offset, document bleed uniform size must be false.
DocumentBleed- OutsideOrRight- Offset	double	no	The amount to offset the outside or right document bleed. Note: To set the bleed outside or right offset, document bleed uniform size must be false.
DocumentBleedTop- Offset	double	no	The amount to offset the top document bleed.
DocumentBleed- UniformSize	boolean	no	If true, uses the document bleed top offset value for bleed offset measurements on all sides of the document. The default setting is true.
DocumentSlug- UniformSize	boolean	no	If true, uses the slug top offset value for slug measurements on all sides of the document. The default value is false.
FacingPages	boolean	no	If true, the document has facing pages.
MasterTextFrame	boolean	no	If true, the document A-master has auto text-frames.
OverprintBlack	boolean	no	If true, overprints black when saving the document.
PageBinding	PageBinding- Options_EnumValue	yes	The page binding. Use Default, RightToLeft, or LeftToRight.

Name	Type	Req	Description
PageHeight	double	no	The height of the page.
PageWidth	double	no	The width of the page.
PagesPerDocument	int	no	The number of pages in the document. (Range: 1 to 9999)
PreserveLayout-WhenShuffling	boolean	no	If true, preserves the layout of spreads that contained more than two pages when allow page shuffle was turned on. If false, changes multi-page spreads to two-page spreads if the spreads were created or changed since allow page shuffle was turned on.
SlugBottomOffset	double	no	The amount to offset the bottom slug. Note: To set the slug bottom offset, document slug uniform size must be false.
SlugInsideOrLeft-Offset	double	no	The amount to offset the inside or left slug. Note: To set the slug inside or left offset, document slug uniform size must be false.
SlugRightOr-OutsideOffset	double	no	The amount to offset the outside or right slug. Note: To set the slug right or outside offset, document slug uniform size must be false.
SlugTopOffset	double	no	The amount to offset the top slug.
SnippetImport-UsesOriginal-Location	boolean	no	If true, causes UI-based snippet import to use original location for page items.

**Table 98. DocumentPreference Properties Represented as Elements**

Name	Type	Req	Description
ColumnGuide-ColorGridColor	InDesignUIColor-Type	no	The color of the column guides, as a <code>UIColors</code> enumeration or an RGB color as a list of three <code>&lt;ListItem&gt;</code> elements.
MarginGuideColor	InDesignUIColor-Type	no	The color of the margin guides, as a <code>UIColors</code> enumeration or an RGB color as a list of three <code>&lt;ListItem&gt;</code> elements.

### 6.3.21 GridPreference

The `<GridPreference>` element stores the grid preferences for the document. Document grid preferences for the baseline grid can have an effect on all text frames that do not specifically define a baseline frame grid (depending on the state of the Align to Baseline setting of the paragraphs in the text frames). For more on this topic, refer to the InDesign online help.

#### Schema Example 91. GridPreference

```
GridPreference_Object = element GridPreference {
    attribute Self { xsd:string },
    attribute DocumentGridShown { xsd:boolean }?,
    attribute DocumentGridSnapt0 { xsd:boolean }?,
    attribute HorizontalGridlineDivision { xsd:double {minInclusive="0.01"
maxInclusive="1000"} }?,
```

```

        attribute VerticalGridlineDivision { xsd:double {minInclusive="0.01"
maxInclusive="1000"} }?,
        attribute HorizontalGridSubdivision { xsd:int {minInclusive="1"
maxInclusive="1000"} }?,
        attribute VerticalGridSubdivision { xsd:int {minInclusive="1"
maxInclusive="1000"} }?,
        attribute GridsInBack { xsd:boolean }?,
        attribute BaselineGridShown { xsd:boolean }?,
        attribute BaselineStart { xsd:double {minInclusive="0" maxInclusive="1000"} }?,
        attribute BaselineDivision { xsd:double {minInclusive="1"
maxInclusive="8640"} }?,
        attribute BaselineViewThreshold { xsd:double {minInclusive="5"
maxInclusive="4000"} }?,
        attribute BaselineGridRelativeOption { BaselineGridRelativeOption_EnumValue }?,
        element Properties {
            element GridColor { InDesignUIColorType_TypeDef }?&
            element BaselineColor { InDesignUIColorType_TypeDef }?
        }
    ?
}
}

```

**IDML Example 75. GridPreference**

```

<GridPreference Self="dGridPreference1" DocumentGridShown="false" DocumentGrid-
SnapTo="false" HorizontalGridlineDivision="72" VerticalGridlineDivision="72"
HorizontalGridSubdivision="8" VerticalGridSubdivision="8" GridsInBack="true"
BaselineGridShown="false" BaselineStart="36" BaselineDivision="12" BaselineView-
Threshold="75" BaselineGridRelativeOption="TopOfPageOfBaselineGridRelativeOption">
<Properties>
    <GridColumn type="enumeration">LightGray</GridColumn>
    <BaselineColor type="enumeration">LightBlue</BaselineColor>
</Properties>
</GridPreference>

```

**Table 99. GridPreference Properties Represented as Attributes**

Name	Type	Req	Description
BaselineDivision	double	no	The amount of space between baseline grid lines. Range: 1 to 8640.
BaselineGrid-RelativeOption	BaselineGrid-RelativeOption_EnumValue	no	The zero point for the baseline grid offset.
BaselineGridShown	boolean	no	If true, displays the baseline grid.
BaselineStart	double	no	The amount to offset the baseline grid from the zero point. Range: 0 to 1000.
BaselineView-Threshold	double	no	The magnification (as a percentage) less than which ruler guides do not appear. (Range: 5 to 4000)
DocumentGridShown	boolean	no	If true, displays the document grid.
DocumentGrid-SnapTo	boolean	no	If true, an object snaps to the nearest grid line when the object is created, moved, or resized.

Name	Type	Req	Description
GridsInBack	boolean	no	If true, places grids behind all other objects on the spread.
HorizontalGrid-Subdivision	int	no	The number of rows into which to subdivide the space between horizontal document grid lines. Range: 1 to 1000.
Horizontal-GridlineDivision	double	no	The amount of space between major horizontal lines in the document grid. Range: .01 to 1000.
VerticalGrid-Subdivision	int	no	The number of columns into which to subdivide the space between vertical document grid lines. Range: .01 to 1000.
VerticalGridline-Division	double	no	The amount of space between major vertical lines in the document grid. Range: .01 to 1000.

**Table 100. GridPreference Properties Represented as Elements**

Name	Type	Req	Description
BaselineColor	InDesignUIColor-Type	no	The color of the baseline grid, as a <code>UIColors</code> enumeration or an RGB color as a list of three <code>&lt;List Item&gt;</code> elements (in the order R, G, B).
GridColor	InDesignUIColor-Type	no	The color of the document grid, as a <code>UIColors</code> enumeration or an RGB color as a list of three <code>&lt;List Item&gt;</code> elements (in the order R, G, B).

### 6.3.22 GuidePreference

The `<GuidePreference>` element stores the guide preferences for the document.

#### Schema Example 92. GuidePreference

```
GuidePreference_Object = element GuidePreference {
    attribute Self { xsd:string },
    attribute GuidesInBack { xsd:boolean }?,
    attribute GuidesShown { xsd:boolean }?,
    attribute GuidesLocked { xsd:boolean }?,
    attribute GuidesSnapt0 { xsd:boolean }?,
    attribute RulerGuidesViewThreshold { xsd:double }?,
    element Properties {
        element RulerGuidesColor { InDesignUIColorType_Def }?
    }
}?
```

#### IDML Example 76. GuidePreference

```
<GuidePreference Self="dGuidePreference1" GuidesInBack="false" GuidesShown="true"
GuidesLocked="false" GuidesSnapt0="true" RulerGuidesViewThreshold="5">
<Properties>
    <RulerGuidesColor type="enumeration">Cyan</RulerGuidesColor>
</Properties>
</GuidePreference>
```

**Table 101. GuidePreference Properties Represented as Attributes**

Name	Type	Req	Description
GuidesInBack	boolean	no	If true, places guides behind all other objects on the spread.
GuidesLocked	boolean	no	If true, guides cannot be moved, added, or deleted.
GuidesShown	boolean	no	If true, displays the guides.
GuidesSnapt0	boolean	no	If true, an object within the specified range snaps to the nearest guide when the object is created, moved, or resized. For range information, see guide snapt0 zone.
RulerGuidesView-Threshold	double	no	The magnification (as a percentage) less than which ruler guides do not appear. (Range: 5 to 4000)

**Table 102. GuidePreference Properties Represented as Elements**

Name	Type	Req	Description
RulerGuidesColor	InDesignUIColor-Type	no	The color of the ruler guides, as a <code>UIColors</code> enumeration or an RGB color as a list of three <code>&lt;ListItem&gt;</code> elements (in the order R, G, B).

### 6.3.23 MarginPreference

The `<MarginPreference>` element controls the default margin and column settings for the pages in a spread. Values that you specify here will apply to the margin preferences of all pages where you have not explicitly defined these attributes and elements.

#### Schema Example 93. MarginPreference

```
MarginPreference_Object = element MarginPreference {
    attribute Self { xsd:string },
    attribute ColumnCount { xsd:int {minInclusive="1" maxInclusive="216"} }?,
    attribute ColumnGutter { xsd:double {minInclusive="0" maxInclusive="1440"} }?,
    attribute Top { xsd:double }?,
    attribute Bottom { xsd:double }?,
    attribute Left { xsd:double }?,
    attribute Right { xsd:double }?,
    attribute ColumnDirection { HorizontalOrVertical_EnumValue }?,
    attribute ColumnsPositions { list { xsd:double * } }?
}
```

### 6.3.24 PasteboardPreference

The `<PasteboardPreference>` element controls the height of the pasteboard (through the `MinimumSpaceAboveAndBelow` attribute) and colors used to display the pasteboard and pasteboard guides.

#### Schema Example 94. PasteboardPreference

```
PasteboardPreference_Object = element PasteboardPreference {
```

```

        attribute Self { xsd:string },
        attribute MinimumSpaceAboveAndBelow { xsd:double }?&
        element Properties {
            element PreviewBackgroundColor { InDesignUIColorType_TypeDef }?&
            element BleedGuideColor { InDesignUIColorType_TypeDef }?&
            element SlugGuideColor { InDesignUIColorType_TypeDef }?
        }
    ?
}

```

**Table 103.** PasteboardPreference Properties Represented as Attributes

Name	Type	Req	Description
MinimumSpace-AboveAndBelow	double	no	The minimum space above and below a page.

**Table 104.** PasteboardPreference Properties Represented as Elements

Name	Type	Req	Description
BleedGuideColor	InDesignUIColor-Type	no	The color of the bleed guides, as a <code>UIColors</code> enumeration or an RGB color as a list of three <code>&lt;ListItem&gt;</code> elements (in the order R, G, B).
Preview-BackgroundColor	InDesignUIColor-Type	no	The color of the pasteboard background, as a <code>UIColors</code> enumeration or an RGB color as a list of three <code>&lt;ListItem&gt;</code> elements (in the order R, G, B).
SlugGuideColor	InDesignUIColor-Type	no	The color of the slug guides, as a <code>UIColors</code> enumeration or an RGB color as a list of three <code>&lt;ListItem&gt;</code> elements (in the order R, G, B).

### 6.3.25 ViewPreference

The `<ViewPreference>` element controls the measurement units, ruler origin, and other user interface properties of the document.

**Note:** Changing the measurement units in the `<ViewPreference>` element only changes the units displayed and used in the InDesign user interface after you have opened the IDML document. Measurement units inside the XML files in an IDML package are always points.

#### Schema Example 95. ViewPreference

```

ViewPreference_Object = element ViewPreference {
    attribute Self { xsd:string },
    attribute GuideSnaptоЖone { xsd:int {minInclusive="1" maxInclusive="36"} }?,
    attribute CursorKeyIncrement { xsd:double {minInclusive="0.001"
maxInclusive="100"} }?,
    attribute HorizontalMeasurementUnits { MeasurementUnits_EnumValue }?,
    attribute VerticalMeasurementUnits { MeasurementUnits_EnumValue }?,
    attribute RulerOrigin { RulerOrigin_EnumValue }?,
    attribute ShowRulers { xsd:boolean }?,
    attribute ShowFrameEdges { xsd:boolean }?,
    attribute TypographicMeasurementUnits { MeasurementUnits_EnumValue }?,

```

```

attribute TextSizeMeasurementUnits { MeasurementUnits_EnumValue }?,
attribute PrintDialogMeasurementUnits { MeasurementUnits_EnumValue }?,
attribute LineMeasurementUnits { MeasurementUnits_EnumValue }?,
attribute PointsPerInch { xsd:double {minInclusive="60" maxInclusive="80"} }?,
attribute HorizontalCustomPoints { xsd:double {minInclusive="4"
maxInclusive="256"} }?,
attribute VerticalCustomPoints { xsd:double {minInclusive="4"
maxInclusive="256"} }?,
attribute ShowNotes { xsd:boolean }?
}

```

**Table 105. ViewPreference Properties Represented as Attributes**

Name	Type	Req	Description
CursorKey-Increment	double	no	The distance to move a specified object when an arrow key is pressed. (Range: 0.001 to 100.)
GuideSnaptоЖone	int	no	The range (in pixels) within which an object snaps to guides. (Range: 1 to 36) Note: Snapping occurs only when guides are shown.
HorizontalCustom-Points	double	no	The distance (in points) between major tick marks on the horizontal ruler. (Range: 4 to 256) Valid only when horizontal measurement units is custom.
Horizontal-MeasurementUnits	MeasurementUnits_EnumValue	no	The measurement unit for the horizontal ruler and other horizontally-measured spaces such as grid columns, horizontal offsets, column gutters, or others. Can be Points, Picas, Inches, InchesDecimal, Millimeters, Centimeters, Ciceros, Q, Ha, AmericanPoints, Custom, or Agates.
LineMeasurement-Units	MeasurementUnits_EnumValue	no	Can be Points, Picas, Inches, Inches-Decimal, Millimeters, Centimeters, Ciceros, Q, Ha, AmericanPoints, Custom, or Agates.
PointsPerInch	double	no	The number of points per inch, typically 72. (Range: 60 to 80)
PrintDialog-MeasurementUnits	MeasurementUnits_EnumValue	no	The measurement units used in the Print dialog box. Can be Points, Picas, Inches, Inches-Decimal, Millimeters, Centimeters, Ciceros, Q, Ha, AmericanPoints, Custom, or Agates.
RulerOrigin	RulerOrigin_Enum-Value	no	The default zero point at the intersection of the vertical and horizontal rulers and the scope of the horizontal ruler. Can be SpreadOrigin, PageOrigin, SpineOrigin.
ShowFrameEdges	boolean	no	If true, displays borders of unselected frames and the diagonal lines in empty unselected frames.
ShowNotes	boolean	no	If true, notes are displayed.
ShowRulers	boolean	no	If true, displays the horizontal and vertical rulers.

Name	Type	Req	Description
TextSize-MeasurementUnits	MeasurementUnits_EnumValue	no	The measurement units used for text size. Can be Points, Picas, Inches, InchesDecimal, Millimeters, Centimeters, Ciceros, Q, Ha, AmericanPoints, Custom, or Agates.
Typographic-MeasurementUnits	MeasurementUnits_EnumValue	no	The measurement units used for type formatting properties (other than text size). Can be Points, Picas, Inches, InchesDecimal, Millimeters, Centimeters, Ciceros, Q, Ha, AmericanPoints, Custom, or Agates.
VerticalCustom-Points	double	no	The distance (in points) between major tick marks on the vertical ruler. (Range: 4 to 256) Valid only when VerticalMeasurementUnits is custom.
Vertical-MeasurementUnits	MeasurementUnits_EnumValue	no	The measurement unit for the vertical ruler and other vertically-measured spaces such as grid rows, vertical offsets, row heights, or others. Can be Points, Picas, Inches, InchesDecimal, Millimeters, Centimeters, Ciceros, Q, Ha, AmericanPoints, Custom, or Agates.

### 6.3.26 PrintPreference

The <PrintPreference> element stores the printing preferences for the document.

#### Schema Example 96. PrintPreference

```
PrintPreference_Object = element PrintPreference {
    attribute Self { xsd:string },
    attribute PrintFile { xsd:string }?,
    attribute Copies { xsd:int }?,
    attribute Collating { xsd:boolean }?,
    attribute ReverseOrder { xsd:boolean }?,
    attribute Sequence { Sequences_EnumValue }?,
    attribute PrintSpreads { xsd:boolean }?,
    attribute PrintMasterPages { xsd:boolean }?,
    attribute PrintNonprinting { xsd:boolean }?,
    attribute PrintBlankPages { xsd:boolean }?,
    attribute PrintGuidesGrids { xsd:boolean }?,
    attribute PaperOffset { xsd:double }?,
    attribute PaperGap { xsd:double }?,
    attribute PaperTransverse { xsd:boolean }?,
    attribute PrintPageOrientation { PrintPageOrientation_EnumValue }?,
    attribute PagePosition { PagePositions_EnumValue }?,
    attribute ScaleMode { ScaleModes_EnumValue }?,
    attribute ScaleWidth { xsd:double }?,
    attribute ScaleHeight { xsd:double }?,
    attribute ScaleProportional { xsd:boolean }?,
    attribute Thumbnails { xsd:boolean }?,
    attribute ThumbnailsPerPage { ThumbsPerPage_EnumValue }?,
    attribute Tile { xsd:boolean }?,
    attribute TilingType { TilingTypes_EnumValue }?,
    attribute TilingOverlap { xsd:double }?,
```

```

attribute AllPrinterMarks { xsd:boolean }?,
attribute CropMarks { xsd:boolean }?,
attribute BleedMarks { xsd:boolean }?,
attribute RegistrationMarks { xsd:boolean }?,
attribute ColorBars { xsd:boolean }?,
attribute PageInformationMarks { xsd:boolean }?,
attribute MarkLineWeight { MarkLineWeight_EnumValue }?,
attribute MarkOffset { xsd:double }?,
attribute UseDocumentBleedToPrint { xsd:boolean }?,
attribute BleedTop { xsd:double {minInclusive="0" maxInclusive="432"} }?,
attribute BleedBottom { xsd:double {minInclusive="0" maxInclusive="432"} }?,
attribute BleedInside { xsd:double {minInclusive="0" maxInclusive="432"} }?,
attribute BleedOutside { xsd:double {minInclusive="0" maxInclusive="432"} }?,
attribute IncludeSlugToPrint { xsd:boolean }?,
attribute ColorOutput { ColorOutputModes_EnumValue }?,
attribute TextAsBlack { xsd:boolean }?,
attribute Trapping { Trapping_EnumValue }?,
attribute Flip { Flip_EnumValue }?,
attribute Negative { xsd:boolean }?,
attribute CompositeAngle { xsd:double }?,
attribute CompositeFrequency { xsd:double }?,
attribute SimulateOverprint { xsd:boolean }?,
attribute PrintCyan { xsd:boolean }?,
attribute CyanAngle { xsd:double }?,
attribute CyanFrequency { xsd:double }?,
attribute PrintMagenta { xsd:boolean }?,
attribute MagentaAngle { xsd:double }?,
attribute MagentaFrequency { xsd:double }?,
attribute PrintYellow { xsd:boolean }?,
attribute YellowAngle { xsd:double }?,
attribute YellowFrequency { xsd:double }?,
attribute PrintBlack { xsd:boolean }?,
attribute BlackAngle { xsd:double }?,
attribute BlackFrequency { xsd:double }?,
attribute SendImageData { ImageDataTypes_EnumValue }?,
attribute FontDownloading { FontDownloading_EnumValue }?,
attribute DownloadPDFFonts { xsd:boolean }?,
attribute PostScriptLevel { PostScriptLevels_EnumValue }?,
attribute DataFormat { DataFormat_EnumValue }?,
attribute SourceSpace { SourceSpaces_EnumValue }?,
attribute Intent { RenderingIntent_EnumValue }?,
attribute OPIImageReplacement { xsd:boolean }?,
attribute OmitEPS { xsd:boolean }?,
attribute OmitPDF { xsd:boolean }?,
attribute OmitBitmaps { xsd:boolean }?,
attribute FlattenerPresetName { xsd:string }?,
attribute IgnoreSpreadOverrides { xsd:boolean }?,
attribute DeviceType { xsd:int }?,
attribute PrintTo { xsd:int }?,
attribute PPDFFile { xsd:string }?,
attribute PrintToDisk { xsd:boolean }?,
attribute PrintRecord { xsd:string }?,
attribute PrintResolution { xsd:double }?,
attribute PaperSizeSelector { xsd:string }?,

```

```

        attribute PaperHeightRange { list { xsd:double ,xsd:double } }?,
        attribute PaperWidthRange { list { xsd:double ,xsd:double } }?,
        attribute PaperOffsetRange { list { xsd:double ,xsd:double } }?,
        attribute SeparationScreening { xsd:string }?,
        attribute CompositeScreening { xsd:string }?,
        attribute SpotAngle { xsd:double }?,
        attribute SpotFrequency { xsd:double }?,
        attribute BleedChain { xsd:boolean }?,
        attribute PreserveColorNumbers { xsd:boolean }?,
        attribute BitmapPrinting { xsd:boolean }?,
        attribute BitmapResolution { xsd:int {minInclusive="72" maxInclusive="1200"} }?,
        attribute PrintLayers { PrintLayerOptions_EnumValue }?,
        element Properties {
            element ActivePrinterPreset {
                (enum_type, PrinterPresetTypes_EnumValue ) |
                (object_type, xsd:string )
            }?&
            element Printer {
                (enum_type, Printer_EnumValue ) |
                (string_type, xsd:string )
            }?&
            element PPD {
                (enum_type, PPDValues_EnumValue ) |
                (string_type, xsd:string )
            }?&
            element PaperSize {
                (enum_type, PaperSizes_EnumValue ) |
                (string_type, xsd:string )
            }?&
            element PaperHeight {
                (enum_type, PaperSize_EnumValue ) |
                (unit_type, xsd:double )
            }?&
            element PaperWidth {
                (enum_type, PaperSize_EnumValue ) |
                (unit_type, xsd:double )
            }?&
            element MarkType {
                (enum_type, MarkTypes_EnumValue ) |
                (string_type, xsd:string )
            }?&
            element Screening {
                (enum_type, Screeening_EnumValue ) |
                (string_type, xsd:string )
            }?&
            element Profile {
                (enum_type, Profile_EnumValue ) |
                (string_type, xsd:string )
            }?&
            element CRD {
                (enum_type, ColorRenderingDictionary_EnumValue ) |
                (string_type, xsd:string )
            }?&
            element PageRange {

```

```

(enum_type, PageRange_EnumValue ) |
(string_type, xsd:string )
} ?&
element PaperSizeRect { RectangleBoundsType_TypeDef } ?&
element ImageablePaperSizeRect { RectangleBoundsType_TypeDef } ?
}
?
}

```

**Table 106. PrintPreference Properties Represented as Attributes**

Name	Type	Req	Description
AllPrinterMarks	boolean	no	If true, prints all printer marks. If false, prints specified printer marks.
BitmapPrinting	boolean	no	If true, uses bitmap printing.
BitmapResolution	int	no	The resolution for bitmap printing. (Range: 72 to 1200) Note: Valid when bitmap printing is true.
BlackAngle	double	no	The angle override for black ink. (Range: 0 to 360)
BlackFrequency	double	no	The frequency override for black ink. (Range: 1 to 500)
BleedBottom	double	no	The height of the bleed area at the bottom of the page. Note: Valid only when use document bleed to print is true.
BleedChain	boolean	no	If true, forces all bleed area settings to be the same, using the most recent bleed measurement setting. If false, allows bleed top, bleed bottom, bleed inside, and bleed outside to have different measurements.
BleedInside	double	no	The width of the bleed area at the inside of the page. Note: Valid only when use document bleed to print is true. (Range: 0 to 432)
BleedMarks	boolean	no	If true, print bleed marks.
BleedOutside	double	no	The width of the bleed area at the outside of the page. Note: Valid only when use document bleed to print is true. (Range: 0 to 432)
BleedTop	double	no	The height of the bleed area at the top of the page. Note: Valid only when use document bleed to print is true. (Range: 0 to 432)
Collating	boolean	no	If true, collate printed copies.
ColorBars	boolean	no	If true, add small squares of color representing the CMYK inks and tints of gray in 10% increments.
ColorOutput	ColorOutputModes_EnumValue	no	The color output mode for composites. Note: Not valid when a device-independent PPD is specified.
CompositeAngle	double	no	The screen angle to use when printing composites. (Range: 0 to 360) Note: Valid only for PostScript or PDF files that use custom screening.

Name	Type	Req	Description
Composite-Frequency	double	no	The screen frequency to use when printing composites. (Range: 1 to 500) Note: Valid only for PostScript or PDF files that use custom screening.
Composite-Screening	string	no	The screening applied to composite printing.
Copies	int	no	The number of copies to print. Note: Not valid when printer is PostScript File.
CropMarks	boolean	no	Prints crop marks that define where the page should be trimmed.
CyanAngle	double	no	The angle override for cyan ink. (Range: 0 to 360)
CyanFrequency	double	no	The frequency override for cyan ink. (Range: 1 to 500)
DataFormat	DataFormat_EnumValue	no	The format in which to send image data to the printer.
DeviceType	int	no	The type of the selected device.
DownloadPPDFonts	boolean	no	If true, downloads all fonts listed in the selected PPD. Valid only when font downloading is complete or subset.
FlattenerPreset-Name	string	no	The name of the transparency flattener preset.
Flip	Flip_EnumValue	no	The direction in which to flip the printed image.
FontDownloading	FontDownloading_EnumValue	no	Controls how fonts are downloaded to the printer.
IgnoreSpread-Overrides	boolean	no	If true, ignores flattener spread overrides.
IncludeSlugTo-Print	boolean	no	If true, includes the slug area in the printed document.
Intent	RenderingIntent_EnumValue	no	The rendering intent. Note: Valid only when use color management is true.
MagentaAngle	double	no	The angle override for magenta ink. (Range: 0 to 360)
MagentaFrequency	double	no	The frequency override for magenta ink. (Range: 1 to 500)
MarkLineWeight	MarkLineWeight_EnumValue	no	The stroke weight (in points) for printer marks.
MarkOffset	double	no	The distance to offset the page marks from the edge of the page.
Negative	boolean	no	If true, prints the document as a negative.
OPIImage-Replacement	boolean	no	If true, prints graphics that are either OPI comments stored in imported EPS files or linked using OPI comments. For information on linking files using OPI comments, see omit EPS, omit PDF, or omit bitmaps.

Name	Type	Req	Description
OmitBitmaps	boolean	no	If true, replaces bitmap images with OPI links.
OmitEPS	boolean	no	If true, replaces EPS images with OPI links.
OmitPDF	boolean	no	If true, replaces PDF images with OPI links.
PPDFFile	string	no	The name of the PDF file.
PageInformation-Marks	boolean	no	If true, prints the filename, page number, current date and time, and color separation name.
PagePosition	PagePositions_EnumValue	no	The position of the page on the printing medium. Note: Valid only when tile is false.
PaperGap	double	no	The space between document pages on the printing medium.
PaperHeightRange	list { double ,xsd:double }	no	A list of the available paper heights.
PaperOffset	double	no	The amount of space to offset the page from the left edge of the imageable area.
PaperOffsetRange	list { double ,xsd:double }	no	A list of the paper offset ranges.
PaperSizeSelector	string	no	The paper size selector.
PaperTransverse	boolean	no	If true, uses transverse orientation.
PaperWidthRange	list { double ,xsd:double }	no	A list of the available paper widths.
PostScriptLevel	PostScriptLevels_EnumValue	no	The PostScript level of the printer.
PreserveColor-Numbers	boolean	no	If true, preserves uncalibrated color numbers.
PrintBlack	boolean	no	If true, prints the black ink. Note: Valid only when trapping is off.
PrintBlankPages	boolean	no	If true, prints blank pages. Note: Valid only when trapping is off.
PrintCyan	boolean	no	If true, prints the cyan ink. Note: Valid only when trapping is off.
PrintFile	string	no	The PostScript file to print to. Note: Valid only when the current printer is defined as postscript file.
PrintGuidesGrids	boolean	no	If true, prints visible guides and baseline grids. Note: Valid only when trapping is off.
PrintLayers	PrintLayer-Options_EnumValue	no	The layers to print.
PrintMagenta	boolean	no	If true, prints the magenta ink. Note: Valid only when trapping is off.
PrintMasterPages	boolean	no	If true, prints master pages.
PrintNonprinting	boolean	no	If true, prints non-printing objects. Note: Valid only when trapping is off.

Name	Type	Req	Description
PrintPage-Orientation	PrintPage-Orientation_Enum-Value	no	The orientation of the printed page.
PrintRecord	string	no	Cached data preserving the last print job (maintained for file round trip).
PrintResolution	double	no	The resolution of the print job.
PrintSpreads	boolean	no	If true, prints each spread with all spread pages on a single sheet. If false, prints spread pages as separate pages.
PrintTo	int	no	The index of the selected printer in the list of available printers.
PrintToDisk	boolean	no	If true, print the file to disk as PostScript.
PrintYellow	boolean	no	If true, prints the yellow ink. Note: Valid only when trapping is off.
RegistrationMarks	boolean	no	If true, prints small targets outside the page area for aligning color separations.
ReverseOrder	boolean	no	If true, prints pages in reverse order.
ScaleHeight	double	no	The amount (as a percentage) that the page height is scaled during printing. (Range: 0 to 1000) Note: Valid only when scale mode is scale width height.
ScaleMode	ScaleModes_Enum-Value	no	The policy for scaling the page. Note: Valid only when printing from Layout view.
ScaleProportional	boolean	no	If true, constrains the proportions of the scaling; uses the most recent value for either scale width or scale height to define both values. Note: Valid only when scale mode is scale width height.
ScaleWidth	double	no	The amount (as a percentage) that the page width is scaled during printing. (Range: 0 to 1000) Note: Valid only when scale mode is scale width height.
SendImageData	ImageDataTypes_EnumValue	no	The image data sent to the printer or file.
Separation-Screening	string	no	The screening to use when printing separations.
Sequence	Sequences_Enum-Value	no	The sequence of pages to print.
SimulateOverprint	boolean	no	If true, simulates the effects of overprinting spot inks with different neutral density values by converting spot colors to process colors for printing. Note: Not valid when the color output mode is defined to leave color profiles unchanged.
SourceSpace	SourceSpaces_EnumValue	no	The source of the color management system. Note: Valid only when use color management is true.
SpotAngle	double	no	The angle of a spot ink.

Name	Type	Req	Description
SpotFrequency	double	no	The screen frequency of a spot ink.
TextAsBlack	boolean	no	If true, prints all text as black unless text has the color None or Paper or a color value that equals white. If false, prints colored text, such as blue hyperlinks, in halftone patterns. Note: Valid only when trapping is off.
Thumbnails	boolean	no	If true, prints thumbnails. Note: Valid only when trapping is off and tile is false.
ThumbnailsPerPage	ThumbsPerPage_EnumValue	no	The number of thumbnails per page.
Tile	boolean	no	If true, tiles pages.
TilingOverlap	double	no	The amount of tiling overlap. Note: Valid only when tiling is true and tiling type is not manual.
TilingType	TilingTypes_EnumValue	no	The tiling type. Note: Valid only when tiling is true.
Trapping	Trapping_EnumValue	no	The type of trapping.
UseDocumentBleed-ToPrint	boolean	no	If true, uses the bleed area set for the document.
YellowAngle	double	no	The angle override for yellow ink. (Range: 0 to 360)
YellowFrequency	double	no	The frequency override for yellow ink. (Range: 1 to 500)

**Table 107. PrintPreference Properties Represented as Elements**

Name	Type	Req	Description
ActivePrinter-Preset	PrinterPreset-Types_EnumValue or string	no	The current printer preset, as a PrinterPresetTypes enumeration or a reference to the Self attribute of a printer preset.
CRD	ColorRenderingDictionary_EnumValue or string	no	The current CRD, as a ColorRenderingDictionary enumeration or a string containing the name of the CRD.
ImageablePaper-SizeRect	RectangleBounds-Type_TypeDef	no	A rectangle defining the imageable area of the paper, as a space-separated list of four values (in the order left, top, right, bottom).
MarkType	MarkTypes_EnumValue or string	no	The type of printer marks, either a MarkTypes enumeration or the name of a custom marks file.
PPD	PPDValues_EnumValue or string	no	The PPD, specified as a PPDValues enumeration or as the name of the PPD.
PageRange	PageRange_EnumValue or string	no	The pages to print, specified either as a PageRange enumeration or a string. To specify a range, separate page numbers in the string with a hyphen (-). To specify separate pages, separate page numbers in the string with a comma (,).

Name	Type	Req	Description
PaperHeight	PaperSize_Enum-Value or double	no	The paper height, as a PaperSize enumeration or double. Note: Valid only when paper size is custom or scale mode is scale width height.
PaperSize	PaperSizes_Enum-Value or string	no	The paper height, as a PaperSize enumeration or string (the name of the paper size).
PaperSizeRect	RectangleBounds-Type_TypeDef	no	A rectangle defining the size of the paper, as a space-separated list of four values (in the order left, top, right, bottom).
PaperWidth	PaperSize_Enum-Value or double	no	The paper width, as a PaperSize enumeration or double. Note: Valid only when paper size is custom or scale mode is scale width height.
Printer	Printer_EnumValue or string	no	The current printer, as a Printer enumeration or string (name of the printer).
Profile	Profile_EnumValue or string	no	The color profile, as a Profile enumeration or string (the name of the profile).
Screening	Screening_Enum-Value or string		The ink screening settings for composite gray output in PostScript or PDF format.

### 6.3.27 PrintBookletOption

The <PrintBookletOption> defines the layout options for the booklet created when you use the Print Booklet feature. The printing options for the booklet are defined by the <PrintBooklet-PrintPreference> element.

#### Schema Example 97. PrintBookletOption

```
PrintBookletOption_Object = element PrintBookletOption {
    attribute Self { xsd:string },
    attribute BookletType { BookletTypeOptions_EnumValue }?,
    attribute SpaceBetweenPages { xsd:double {minInclusive="0" maxInclusive="288"} }?,
    attribute BleedBetweenPages { xsd:double {minInclusive="0" maxInclusive="144"} }?,
    attribute Creep { xsd:double {minInclusive="-144" maxInclusive="144"} }?,
    attribute SignatureSize { SignatureSizeOptions_EnumValue }?,
    attribute TopMargin { xsd:double {minInclusive="0" maxInclusive="288"} }?,
    attribute BottomMargin { xsd:double {minInclusive="0" maxInclusive="288"} }?,
    attribute LeftMargin { xsd:double {minInclusive="0" maxInclusive="288"} }?,
    attribute RightMargin { xsd:double {minInclusive="0" maxInclusive="288"} }?,
    attribute AutoAdjustMargins { xsd:boolean }?,
    attribute MarginsUniformSize { xsd:boolean }?,
    attribute PrintBlankPrinterSpreads { xsd:boolean }?,
    element Properties {
        element PageRange {
            (enum_type, PageRange_EnumValue) |
            (string_type, xsd:string)
        }?
    }?
}
```

### 6.3.28 PrintBookletPrintPreference

The printing options for the Print Booklet are defined by the `<PrintBookletPrintPreference>` element. The `<PrintBookletOption>` defines the layout options for the booklet.

#### Schema Example 98. PrintBookletPrintPreference

```
PrintBookletPrintPreference_Object = element PrintBookletPrintPreference {
    attribute Self { xsd:string },
    attribute PrinterList { list { xsd:string * } }?,
    attribute PPDLList { list { xsd:string * } }?,
    attribute PaperSizeList { list { xsd:string * } }?,
    attribute ScreeningList { list { xsd:string * } }?,
    attribute PrintFile { xsd:string }?,
    attribute Copies { xsd:int }?,
    attribute Collating { xsd:boolean }?,
    attribute ReverseOrder { xsd:boolean }?,
    attribute PrintNonprinting { xsd:boolean }?,
    attribute PrintBlankPages { xsd:boolean }?,
    attribute PrintGuidesGrids { xsd:boolean }?,
    attribute PaperOffset { xsd:double }?,
    attribute PaperGap { xsd:double }?,
    attribute PaperTransverse { xsd:boolean }?,
    attribute PrintPageOrientation { PrintPageOrientation_EnumValue }?,
    attribute PagePosition { PagePositions_EnumValue }?,
    attribute ScaleMode { ScaleModes_EnumValue }?,
    attribute ScaleWidth { xsd:double }?,
    attribute ScaleHeight { xsd:double }?,
    attribute ScaleProportional { xsd:boolean }?,
    attribute PrintLayers { PrintLayerOptions_EnumValue }?,
    attribute AllPrinterMarks { xsd:boolean }?,
    attribute CropMarks { xsd:boolean }?,
    attribute BleedMarks { xsd:boolean }?,
    attribute RegistrationMarks { xsd:boolean }?,
    attribute ColorBars { xsd:boolean }?,
    attribute PageInformationMarks { xsd:boolean }?,
    attribute MarkLineWeight { MarkLineWeight_EnumValue }?,
    attribute MarkOffset { xsd:double }?,
    attribute UseDocumentBleedToPrint { xsd:boolean }?,
    attribute BleedTop { xsd:double {minInclusive="0" maxInclusive="432"} }?,
    attribute BleedBottom { xsd:double {minInclusive="0" maxInclusive="432"} }?,
    attribute BleedInside { xsd:double {minInclusive="0" maxInclusive="432"} }?,
    attribute BleedOutside { xsd:double {minInclusive="0" maxInclusive="432"} }?,
    attribute BleedChain { xsd:boolean }?,
    attribute ColorOutput { ColorOutputModes_EnumValue }?,
    attribute TextAsBlack { xsd:boolean }?,
    attribute Trapping { Trapping_EnumValue }?,
    attribute Flip { Flip_EnumValue }?,
    attribute Negative { xsd:boolean }?,
    attribute CompositeAngle { xsd:double }?,
    attribute CompositeFrequency { xsd:double }?,
    attribute SimulateOverprint { xsd:boolean }?,
    attribute PrintCyan { xsd:boolean }?,
    attribute CyanAngle { xsd:double }?,
    attribute CyanFrequency { xsd:double }?,
```

```

        attribute PrintMagenta { xsd:boolean }?,
        attribute MagentaAngle { xsd:double }?,
        attribute MagentaFrequency { xsd:double }?,
        attribute PrintYellow { xsd:boolean }?,
        attribute YellowAngle { xsd:double }?,
        attribute YellowFrequency { xsd:double }?,
        attribute PrintBlack { xsd:boolean }?,
        attribute BlackAngle { xsd:double }?,
        attribute BlackFrequency { xsd:double }?,
        attribute SendImageData { ImageDataTypes_EnumValue }?,
        attribute FontDownloading { FontDownloading_EnumValue }?,
        attribute DownloadPPDFonts { xsd:boolean }?,
        attribute PostScriptLevel { PostScriptLevels_EnumValue }?,
        attribute DataFormat { DataFormat_EnumValue }?,
        attribute SourceSpace { SourceSpaces_EnumValue }?,
        attribute Intent { RenderingIntent_EnumValue }?,
        attribute PreserveColorNumbers { xsd:boolean }?,
        attribute OPIImageReplacement { xsd:boolean }?,
        attribute OmitEPS { xsd:boolean }?,
        attribute OmitPDF { xsd:boolean }?,
        attribute OmitBitmaps { xsd:boolean }?,
        attribute FlattenerPresetName { xsd:string }?,
        attribute IgnoreSpreadOverrides { xsd:boolean }?,
        attribute BitmapPrinting { xsd:boolean }?,
        attribute BitmapResolution { xsd:int {minInclusive="72" maxInclusive="1200"} }?,
        attribute DeviceType { xsd:int }?,
        attribute PrintTo { xsd:int }?,
        attribute PPDFile { xsd:string }?,
        attribute PrintToDisk { xsd:boolean }?,
        attribute PrintRecord { xsd:string }?,
        attribute PrintResolution { xsd:double }?,
        attribute PaperSizeSelector { xsd:string }?,
        attribute PaperHeightRange { list { xsd:double ,xsd:double } }?,
        attribute PaperWidthRange { list { xsd:double ,xsd:double } }?,
        attribute PaperOffsetRange { list { xsd:double ,xsd:double } }?,
        attribute SeparationScreening { xsd:string }?,
        attribute CompositeScreening { xsd:string }?,
        attribute SpotAngle { xsd:double }?,
        attribute SpotFrequency { xsd:double }?,
        element Properties {
            element Printer {
                (enum_type, Printer_EnumValue ) |
                (string_type, xsd:string )
            }?&
            element PPD {
                (enum_type, PPDValues_EnumValue ) |
                (string_type, xsd:string )
            }?&
            element PaperSize {
                (enum_type, PaperSizes_EnumValue ) |
                (string_type, xsd:string )
            }?&
            element PaperHeight {
                (enum_type, PaperSize_EnumValue ) |

```

```

        (unit_type, xsd:double )
    }?&
    element PaperWidth {
        (enum_type, PaperSize_EnumValue ) |
        (unit_type, xsd:double )
    }?&
    element MarkType {
        (enum_type, MarkTypes_EnumValue ) |
        (string_type, xsd:string )
    }?&
    element Screening {
        (enum_type, Screeening_EnumValue ) |
        (string_type, xsd:string )
    }?&
    element Profile {
        (enum_type, Profile_EnumValue ) |
        (string_type, xsd:string )
    }?&
    element CRD {
        (enum_type, ColorRenderingDictionary_EnumValue ) |
        (string_type, xsd:string )
    }?&
    element ActivePrinterPreset {
        (enum_type, PrinterPresetTypes_EnumValue ) |
        (object_type, xsd:string )
    }?&
    element PaperSizeRect { RectangleBoundsType_TypeDef }?&
    element ImageablePaperSizeRect { RectangleBoundsType_TypeDef }?
}
?
}

```

### 6.3.29 MetadataPacketPreference

The <MetadataPacketPreference> element stores the XMP data for the document.

#### Schema Example 99. MetadataPacketPreference

```

MetadataPacketPreference_Object = element MetadataPacketPreference {
    attribute Self { xsd:string },
    element Properties {
        element Contents { xsd:string }?
    }
?
}

```

### 6.3.30 IndexOptions

The <IndexOptions> element stores settings for index creation. These options have no effect on indices already included in the IDML document.

#### Schema Example 100. IndexOptions

```

IndexOptions_Object = element IndexOptions {

```

```

        attribute Self { xsd:string },
        attribute Title { xsd:string }?,
        attribute TitleStyle { xsd:string }?,
        attribute ReplaceExistingIndex { xsd:boolean }?,
        attribute IncludeBookDocuments { xsd:boolean }?,
        attribute IncludeHiddenEntries { xsd:boolean }?,
        attribute IndexFormat { IndexFormat_EnumValue }?,
        attribute IncludeSectionHeadings { xsd:boolean }?,
        attribute IncludeEmptyIndexSections { xsd:boolean }?,
        attribute Level1Style { xsd:string }?,
        attribute Level2Style { xsd:string }?,
        attribute Level3Style { xsd:string }?,
        attribute Level4Style { xsd:string }?,
        attribute SectionHeadingStyle { xsd:string }?,
        attribute PageNumberStyle { xsd:string }?,
        attribute CrossReferenceStyle { xsd:string }?,
        attribute CrossReferenceTopicStyle { xsd:string }?,
        attribute FollowingTopicSeparator { xsd:string }?,
        attribute BetweenEntriesSeparator { xsd:string }?,
        attribute PageRangeSeparator { xsd:string }?,
        attribute BetweenPageNumbersSeparator { xsd:string }?,
        attribute BeforeCrossReferenceSeparator { xsd:string }?,
        attribute EntryEndSeparator { xsd:string }?
    }
}

```

### 6.3.31 IndexHeaderSetting

The <IndexHeaderSetting> element stores the strings used by the headers added to the index during index generation. These options have no effect on indices already included in the IDML document.

#### Schema Example 101. IndexHeaderSetting

```

IndexHeaderSetting_Object = element IndexHeaderSetting {
    attribute Self { xsd:string },
    attribute HeaderSetName { xsd:string }?,
    attribute HeaderSetLanguage { xsd:int }?,
    attribute IndexHeaderSetHandler { xsd:int }?,
    attribute IndexHeaderSetGroupValue { xsd:int }?,
    attribute IndexHeaderSetGroupOptionValue { xsd:int }?,
    element Properties {
        element ListOfIndexHeaderGroupType { element IndexHeaderGroupType {
            IndexHeaderGroupType_TypeDef }*
        }?
    }
}

```

### 6.3.32 PageItemDefault

The <PageItemDefault> element controls the default page item formatting for an InDesign document. Values that you specify here will apply to all page items that do not explicitly define

these attributes and elements. All of the properties of the <PageItemDefault> element can be found in the “Common Page Item Properites” section.

#### **Schema Example 102. PageItemDefault**

```
PageItemDefault_Object = element PageItemDefault {
    attribute Self { xsd:string },
    attribute AppliedGraphicObjectStyle { xsd:string }?,
    attribute AppliedTextObjectStyle { xsd:string }?,
    attribute AppliedGridObjectStyle { xsd:string }?,
    attribute FillColor { xsd:string }?,
    attribute FillTint { xsd:double }?,
    attribute StrokeWeight { xsd:double }?,
    attribute MiterLimit { xsd:double {minInclusive="1" maxInclusive="500"} }?,
    attribute EndCap { EndCap_EnumValue }?,
    attribute EndJoin { EndJoin_EnumValue }?,
    attribute StrokeType { xsd:string }?,
    attribute LeftLineEnd { ArrowHead_EnumValue }?,
    attribute RightLineEnd { ArrowHead_EnumValue }?,
    attribute StrokeColor { xsd:string }?,
    attribute StrokeTint { xsd:double }?,
    attribute CornerOption { CornerOptions_EnumValue }?,
    attribute CornerRadius { xsd:double }?,
    attribute GradientFillAngle { xsd:double }?,
    attribute GradientStrokeAngle { xsd:double }?,
    attribute OverprintStroke { xsd:boolean }?,
    attribute OverprintFill { xsd:boolean }?,
    attribute GapColor { xsd:string }?,
    attribute GapTint { xsd:double }?,
    attribute OverprintGap { xsd:boolean }?,
    attribute StrokeAlignment { StrokeAlignment_EnumValue }?,
    attribute Nonprinting { xsd:boolean }?,
    (
        TransparencySetting_Object?&
        StrokeTransparencySetting_Object?&
        FillTransparencySetting_Object?&
        ContentTransparencySetting_Object?
    )
}
```

#### **6.3.33 FrameFittingOption**

The <FrameFittingOption> element controls the default fitting behavior for all page items in a document. Values that you specify here will apply to all page items that do not explicitly define these attributes and elements. For more on frame fitting options, refer to the InDesign documentation.

#### **Schema Example 103. FrameFittingOption**

```
FrameFittingOption_Object = element FrameFittingOption {
    attribute Self { xsd:string },
    attribute LeftCrop { xsd:double }?,
    attribute TopCrop { xsd:double }?,
    attribute RightCrop { xsd:double }?,
    attribute BottomCrop { xsd:double }?,
```

```

        attribute FittingOnEmptyFrame { EmptyFrameFittingOptions_EnumValue }?,
        attribute FittingAlignment { AnchorPoint_EnumValue }?
    }
}

```

### 6.3.34 ButtonPreference

The <ButtonPreference> element controls the default fitting behavior for all buttons in a document. Values that you specify here will apply to all buttons that do not explicitly define these attributes and elements.

#### Schema Example 104. ButtonPreference

```

ButtonPreference_Object = element ButtonPreference {
    attribute Self { xsd:string },
    attribute Name { xsd:string }?
}

```

### 6.3.35 TinDocumentDDataObject

#### Schema Example 105. TinDocumentDataObject

```

TinDocumentDataObject_Object = element TinDocumentDataObject {
    element Properties {
        element GaijiRefMaps { text }?
    }
    ?
}

```

### 6.3.36 LayoutGridDataInformation

The <LayoutGridDataInformation> element controls the default layout grid for a document.

#### Schema Example 106. LayoutGridDataInformation

```

LayoutGridDataInformation_Object = element LayoutGridDataInformation {
    attribute Self { xsd:string },
    attribute FontStyle { xsd:string }?,
    attribute PointSize { xsd:double }?,
    attribute CharacterAki { xsd:double }?,
    attribute LineAki { xsd:double }?,
    attribute HorizontalScale { xsd:double }?,
    attribute VerticalScale { xsd:double }?,
    element Properties {
        element AppliedFont {
            (object_type, xsd:string) |
            (string_type, xsd:string)
        }?
    }
    ?
}

```

### 6.3.37 StoryGridDataInformation

The <StoryGridDataInformation> element controls the default story grid for a document.

#### Schema Example 107. StoryGridDataInformation

```
StoryGridDataInformation_Object = element StoryGridDataInformation {
    attribute Self { xsd:string },
    attribute FontStyle { xsd:string }?,
    attribute PointSize { xsd:double }?,
    attribute CharacterAki { xsd:double }?,
    attribute LineAki { xsd:double }?,
    attribute HorizontalScale { xsd:double }?,
    attribute VerticalScale { xsd:double }?,
    attribute LineAlignment { LineAlignment_EnumValue }?,
    attribute GridAlignment { GridAlignment_EnumValue }?,
    attribute CharacterAlignment { CharacterAlignment_EnumValue }?,
    attribute GridView { GridViewSettings_EnumValue }?,
    attribute CharacterCountLocation { CharacterCountLocation_EnumValue }?,
    attribute CharacterCountSize { xsd:double }?,
    element Properties {
        element AppliedFont {
            (object_type, xsd:string) |
            (string_type, xsd:string)
        }?
    }
}?
```

### 6.3.38 CjkGridPreference

#### Schema Example 108. CjkGridPreference

```
CjkGridPreference_Object = element CjkGridPreference {
    attribute Self { xsd:string },
    attribute ShowAllLayoutGrids { xsd:boolean }?,
    attribute ShowAllFrameGrids { xsd:boolean }?,
    attribute MinimumScale { xsd:double }?,
    attribute SnapToLayoutGrid { xsd:boolean }?,
    attribute ColorEveryNthCell { xsd:short }?,
    attribute SingleLineColorMode { xsd:boolean }?,
    attribute ICFMode { xsd:boolean }?,
    attribute UseCircularCells { xsd:boolean }?,
    attribute ShowCharacterCount { xsd:boolean }?,
    element Properties {
        element LayoutGridColorIndex { InDesignUIColorType_TypeDef }?
    }
}?
```

**Table 108. CjkGridPreference Properties Represented as Attributes**

Name	Type	Req	Description
ColorEveryNthCell	short	no	Applies the grid color to every nth cell, where n is the value of this property.
ICFMode	boolean	no	If true, uses ICF mode for grid cells. If false, uses virtual body mode.
MinimumScale	double	no	The view magnification (as a percentage) less than which grids do not appear. (Range: 5 to 4000)
ShowAllFrameGrids	boolean	no	If true, displays the frame (story) grids.
ShowAllLayout-Grids	boolean	no	If true, displays the layout grids.
ShowCharacter-Count	boolean	no	If true, displays the character count for the frame.
SingleLineColor-Mode	boolean	no	If true, applies the grid color from the edge of the line. If false, applies the grid color from the corner of the frame.
SnapToLayoutGrid	boolean	no	If true, objects snap to the layout grid.
UseCircularCells	boolean	no	If true, cell shape is circular. If false, cell shape is rectangular.

**Table 109. CjkGridPreference Properties Represented as Elements**

Name	Type	Req	Description
LayoutGridColor-Index	InDesignUIColor-Type	no	The color of the layout grid as a UIColor enumeration or an RGB color as a list of three <ListItem> elements (in the order R, G, B).

### 6.3.39 MojikumiUiPreference

The <MojikumiUiPreference> element controls the user interface for Mojikumi.

#### Schema Example 109. MojikumiUiPreference

```
MojikumiUiPreference_Object = element MojikumiUiPreference {
    attribute Self { xsd:string },
    attribute MojikumiUiSettings { xsd:short }?
}
```

**Table 110. MojikumiUiPreference Properties Represented as Attributes**

Name	Type	Req	Description
MojikumiUi-Settings	short	no	User interface settings for Mojikumi features.

### 6.3.40 ChapterNumberPreference

The <ChapterNumberPreference> element stores information used by chapter numbering text variables in the document.

#### Schema Example 110. ChapterNumberPreference

```
ChapterNumberPreference_Object = element ChapterNumberPreference {
    attribute Self { xsd:string },
    attribute ChapterNumber { xsd:int }?,
    attribute ChapterNumberSource { ChapterNumberSources_EnumValue }?,
    element Properties {
        element ChapterNumberFormat {
            (enum_type, NumberingStyle_EnumValue ) |
            (string_type, xsd:string )
        }?
    }?
}
```

**Table 111. ChapterNumberPreference Properties Represented as Attributes**

Name	Type	Req	Description
ChapterNumber	int	no	Chapter number.
ChapterNumber-Source	ChapterNumber-Sources_EnumValue	no	Source for generating the chapter number. Can be UserDefined, ContinueFromPrevious-Document, or SameAsPreviousDocument.

**Table 112. ChapterNumberPreference Properties Represented as Elements**

Name	Type	Req	Description
ChapterNumber-Format	NumberingStyle_EnumValue or string.	no	The chapter numbering type, as a NumberingStyle enumeration or as a string. The string corresponds to the options found on the Style pop-up menu in the Document Chapter Numbering section of the Numbering & Section Options dialog box in InDesign's user interface.

## 6.4 Styles

An InDesign document contains a number of style objects: notably paragraph styles, character styles, and object styles, but also including table styles, cell styles, and anchored object settings. In an IDML package, the `styles.xml` file in the Resources folder contains the styles used in the document.

### 6.4.1 Style Paths and the Self Attribute

When you create a reference to a style in an IDML package, you specify the location of the style element using a path-like syntax. For example, when you want to refer to a paragraph style named “Heading1” in the `<RootParagraphStyle>` element of a document, you use the value of its `Self` attribute:

```
\ParagraphStyles\Heading1
```

When a style appears within a style group other than the root style group, InDesign does not use a backslash as a path separator; instead, a colon is used. The colon appears in the `Name` attribute of the style, but is encoded as `%3a` when it appears in the `Self` attribute, as shown in the example below:

```
<ParagraphStyle Self="ParagraphStyle\Headings%3aHeading1"
Name="Headings:Heading1">
```

In the following example, a paragraph style group named “TableHeadings” has been created inside the “Headings” paragraph style group.

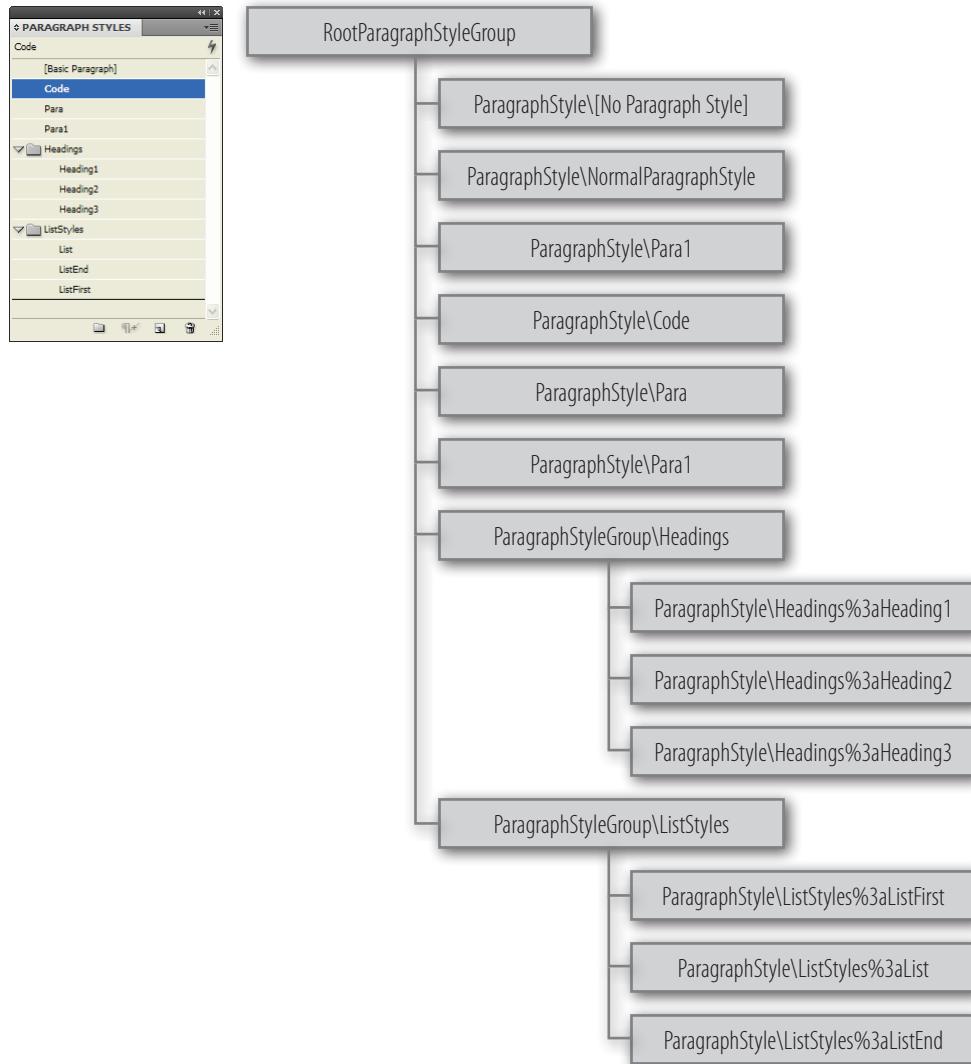
```
<ParagraphStyle Self="ParagraphStyle\Headings%3aTableHeadings%3aTableHeading1"
Name="Headings:TableHeadings:TableHeading1">
```

### 6.4.2 ParagraphStyles

A paragraph style includes both character and paragraph formatting attributes, and can be applied to a paragraph or range of paragraphs. Paragraph styles can define every formatting attribute that can be applied to a range of paragraphs, including character formatting. Paragraph styles in an InDesign document can be stored in paragraph styles groups for organizational purposes. In IDML, it is not necessary to define every formatting attribute of a paragraph style, but you should be aware that values that you do not include will be defined using the application’s defaults.

Each IDML package document contains a `<RootParagraphStyleGroup>` element, which, in turn, contains all of the `<ParagraphStyle>` and `<ParagraphStyleGroup>` elements in the IDML package. A `<ParagraphStyleGroup>` element can contain other `<ParagraphStyleGroup>` elements. The following figure shows the relationship between the elements in an example InDesign document.

Figure 54. Example RootParagraphStyleGroup, ParagraphStyle, and ParagraphStyleGroup Elements (Self attribute values shown)



The `<RootParagraphStyleGroup>` and `<ParagraphStyleGroup>` elements have the same structure.

#### Schema Example 111. ParagraphStyleGroup

```

ParagraphStyleGroup_Object = element ParagraphStyleGroup {
    attribute Self { xsd:string },
    attribute Name { xsd:string }?,
    element Properties {
        element Label { element KeyValuePair { KeyValuePair_TypeDef }* }?
    }?
}
?
,
(
    ParagraphStyle_Object*&
    ParagraphStyleGroup_Object*
)
  
```

```
}
```

InDesign documents always include two default paragraph styles, “Normal Paragraph Style” and “[No Paragraph Style].” The former style can be edited by the user (it appears as “Basic Paragraph Style” in the user interface); the latter contains default formatting and cannot be edited.

Many document designs feature hierarchies of styles sharing certain attributes. The headings and subheads, for example, often use the same font. You can easily create links between similar styles by creating a base, or parent, style. When you edit the parent style, the child styles will change as well. You can then edit the child styles to distinguish it from the parent style. The `BasedOn` attribute of a paragraph style refers to the parent style of a `<ParagraphStyle>` element.

The `NextStyle` attribute only affects the user interface, and automatically applies styles as the user types text. If, for example, your document’s design calls for the style “body text” to follow a heading style named “heading 1,” you can set the Next Style option for “heading 1” to “body text.” After you’ve typed a paragraph styled with “heading 1,” pressing Enter or Return starts a new paragraph styled with “body text.”

The following example shows an example `<ParagraphStyleGroup>` element inside the `<Root-ParagraphStyleGroup>` element. We have omitted the formatting specified in the individual `<ParagraphStyle>` elements for clarity.

#### IDML Example 77. ParagraphStyleGroup

```
<idPkg:Styles xmlns:idPkg="http://ns.adobe.com/AdobeInDesign/idml/1.0/packaging">
  <RootParagraphStyleGroup Self="u69">
    <ParagraphStyleGroup Self="ParagraphStyleGroup\Headings" Name="Headings">
      <ParagraphStyle Self="ParagraphStyle\Headings%3aHeading1"
        Name="Headings:Heading1">
        </ParagraphStyle>
      <ParagraphStyle Self="ParagraphStyle\Headings%3aHeading2"
        Name="Headings:Heading2">
        </ParagraphStyle>
      <ParagraphStyle Self="ParagraphStyle\Headings%3aHeading3"
        Name="Headings:Heading3">
        </ParagraphStyle>
    </ParagraphStyleGroup>
  </RootParagraphStyleGroup>
</idPkg:Styles>
```

#### Schema Example 112. ParagraphStyle

```
ParagraphStyle_Object = element ParagraphStyle {
  attribute Self { xsd:string },
  attribute Name { xsd:string },
  attribute Imported { xsd:boolean }?,
  attribute NextStyle { xsd:string }?,
  attribute FirstLineIndent { xsd:double }?,
  attribute LeftIndent { xsd:double }?,
  attribute RightIndent { xsd:double }?,
  attribute SpaceBefore { xsd:double }?,
  attribute SpaceAfter { xsd:double }?,
  attribute Justification { Justification_EnumValue }?,
  attribute SingleWordJustification { SingleWordJustification_EnumValue }?,
  attribute AutoLeading { xsd:double }?,
  attribute DropCapLines { xsd:short {minInclusive="0" maxInclusive="25"} }?,
```

```

attribute DropCapCharacters { xsd:short {minInclusive="0"
maxInclusive="150"} }?,
attribute KeepLinesTogether { xsd:boolean }?,
attribute KeepAllLinesTogether { xsd:boolean }?,
attribute KeepWithNext { xsd:short {minInclusive="0" maxInclusive="5"} }?,
attribute KeepFirstLines { xsd:short {minInclusive="1" maxInclusive="50"} }?,
attribute KeepLastLines { xsd:short {minInclusive="1" maxInclusive="50"} }?,
attribute StartParagraph { StartParagraph_EnumValue }?,
attribute Composer { xsd:string }?,
attribute MinimumWordSpacing { xsd:double }?,
attribute MaximumWordSpacing { xsd:double }?,
attribute DesiredWordSpacing { xsd:double }?,
attribute MinimumLetterSpacing { xsd:double }?,
attribute MaximumLetterSpacing { xsd:double }?,
attribute DesiredLetterSpacing { xsd:double }?,
attribute MinimumGlyphScaling { xsd:double }?,
attribute MaximumGlyphScaling { xsd:double }?,
attribute DesiredGlyphScaling { xsd:double }?,
attribute RuleAbove { xsd:boolean }?,
attribute RuleAboveOverprint { xsd:boolean }?,
attribute RuleAboveLineWeight { xsd:double }?,
attribute RuleAboveTint { xsd:double }?,
attribute RuleAboveOffset { xsd:double }?,
attribute RuleAboveLeftIndent { xsd:double }?,
attribute RuleAboveRightIndent { xsd:double }?,
attribute RuleAboveWidth { RuleWidth_EnumValue }?,
attribute RuleAboveGapTint { xsd:double }?,
attribute RuleAboveGapOverprint { xsd:boolean }?,
attribute RuleBelow { xsd:boolean }?,
attribute RuleBelowLineWeight { xsd:double }?,
attribute RuleBelowTint { xsd:double }?,
attribute RuleBelowOffset { xsd:double }?,
attribute RuleBelowLeftIndent { xsd:double }?,
attribute RuleBelowRightIndent { xsd:double }?,
attribute RuleBelowWidth { RuleWidth_EnumValue }?,
attribute RuleBelowGapTint { xsd:double }?,
attribute HyphenateCapitalizedWords { xsd:boolean }?,
attribute Hyphenation { xsd:boolean }?,
attribute HyphenateBeforeLast { xsd:short {minInclusive="1"
maxInclusive="15"} }?,
attribute HyphenateAfterFirst { xsd:short {minInclusive="1"
maxInclusive="15"} }?,
attribute HyphenateWordsLongerThan { xsd:short {minInclusive="3"
maxInclusive="25"} }?,
attribute HyphenateLadderLimit { xsd:short {minInclusive="0"
maxInclusive="25"} }?,
attribute HyphenationZone { xsd:double }?,
attribute HyphenWeight { xsd:short {minInclusive="0" maxInclusive="10"} }?,
attribute FontStyle { xsd:string }?,
attribute PointSize { xsd:double }?,
attribute KerningMethod { xsd:string }?,
attribute Tracking { xsd:double }?,
attribute Capitalization { Capitalization_EnumValue }?,
attribute Position { Position_EnumValue }?,

```

```

        attribute Underline { xsd:boolean }?,
        attribute StrikeThru { xsd:boolean }?,
        attribute Ligatures { xsd:boolean }?,
        attribute NoBreak { xsd:boolean }?,
        attribute HorizontalScale { xsd:double }?,
        attribute VerticalScale { xsd:double }?,
        attribute BaselineShift { xsd:double }?,
        attribute Skew { xsd:double }?,
        attribute FillTint { xsd:double }?,
        attribute StrokeTint { xsd:double }?,
        attribute StrokeWeight { xsd:double }?,
        attribute OverprintStroke { xsd:boolean }?,
        attribute OverprintFill { xsd:boolean }?,
        attribute OTFFigureStyle { OTFFigureStyle_EnumValue }?,
        attribute OTFOOrdinal { xsd:boolean }?,
        attribute OTFFraction { xsd:boolean }?,
        attribute OTFDiscretionaryLigature { xsd:boolean }?,
        attribute OTFTitling { xsd:boolean }?,
        attribute OTFContextualAlternate { xsd:boolean }?,
        attribute OTFSwash { xsd:boolean }?,
        attribute UnderlineTint { xsd:double }?,
        attribute UnderlineGapTint { xsd:double }?,
        attribute UnderlineOverprint { xsd:boolean }?,
        attribute UnderlineGapOverprint { xsd:boolean }?,
        attribute UnderlineOffset { xsd:double }?,
        attribute UnderlineWeight { xsd:double }?,
        attribute StrikeThroughTint { xsd:double }?,
        attribute StrikeThroughGapTint { xsd:double }?,
        attribute StrikeThroughOverprint { xsd:boolean }?,
        attribute StrikeThroughGapOverprint { xsd:boolean }?,
        attribute StrikeThroughOffset { xsd:double }?,
        attribute StrikeThroughWeight { xsd:double }?,
        attribute FillColor { xsd:string }?,
        attribute StrokeColor { xsd:string }?,
        attribute AppliedLanguage { xsd:string }?,
        attribute KeyboardShortcut { list { xsd:short ,xsd:short } }?,
        attribute LastLineIndent { xsd:double }?,
        attribute HyphenateLastWord { xsd:boolean }?,
        attribute OTFSashedZero { xsd:boolean }?,
        attribute OTFHistorical { xsd:boolean }?,
        attribute OTFStylisticSets { xsd:int }?,
        attribute GradientFillLength { xsd:double }?,
        attribute GradientFillAngle { xsd:double }?,
        attribute GradientStrokeLength { xsd:double }?,
        attribute GradientStrokeAngle { xsd:double }?,
        attribute GradientFillStart { UnitPointType_TypeDef }?,
        attribute GradientStrokeStart { UnitPointType_TypeDef }?,
        attribute RuleBelowOverprint { xsd:boolean }?,
        attribute RuleBelowGapOverprint { xsd:boolean }?,
        attribute DropcapDetail { xsd:int }?,
        attribute HyphenateAcrossColumns { xsd:boolean }?,
        attribute KeepRuleAboveInFrame { xsd:boolean }?,
        attribute IgnoreEdgeAlignment { xsd:boolean }?,
        attribute OTFMark { xsd:boolean }?,

```

```

attribute OTFLocale { xsd:boolean }?,
attribute PositionalForm { PositionalForms_EnumValue }?,
attribute ParagraphDirection { ParagraphDirection_EnumValue }?,
attribute ParagraphJustification { ParagraphJustification_EnumValue }?,
attribute MiterLimit { xsd:double {minInclusive="1" maxInclusive="500"} }?,
attribute StrokeAlignment { TextStrokeAlign_EnumValue }?,
attribute EndJoin { OutlineJoin_EnumValue }?,
attribute OTFOverlapSwash { xsd:boolean }?,
attribute OTFStylisticAlternate { xsd:boolean }?,
attribute OTFJustificationAlternate { xsd:boolean }?,
attribute OTFSretchedAlternate { xsd:boolean }?,
attribute CharacterDirection { CharacterDirection_EnumValue }?,
attribute KeyboardDirection { CharacterDirection_EnumValue }?,
attribute DigitsType { DigitsType_EnumValue }?,
attribute Kashidas { Kashidas_EnumValue }?,
attribute DiacriticPosition { DiacriticPosition_EnumValue }?,
attribute XOffsetDiacritic { xsd:double }?,
attribute YOffsetDiacritic { xsd:double }?,
attribute GotoNextX { GotoNextX_EnumValue }?,
attribute PageNumberType { PageNumberType_EnumValue }?,
attribute CharacterAlignment { CharacterAlignment_EnumValue }?,
attribute Tsume { xsd:double }?,
attribute LeadingAki { xsd:double }?,
attribute TrailingAki { xsd:double }?,
attribute CharacterRotation { xsd:double }?,
attribute Jidori { xsd:short }?,
attribute ShataiMagnification { xsd:double }?,
attribute ShataiDegreeAngle { xsd:double }?,
attribute ShataiAdjustRotation { xsd:boolean }?,
attribute ShataiAdjustTsume { xsd:boolean }?,
attribute Tatechuyoko { xsd:boolean }?,
attribute TatechuyokoXOffset { xsd:double }?,
attribute TatechuyokoYOffset { xsd:double }?,
attribute KentenTint { xsd:double }?,
attribute KentenStrokeTint { xsd:double }?,
attribute KentenWeight { xsd:double }?,
attribute KentenOverprintFill { AdornmentOverprint_EnumValue }?,
attribute KentenOverprintStroke { AdornmentOverprint_EnumValue }?,
attribute KentenKind { KentenCharacter_EnumValue }?,
attribute KentenPlacement { xsd:double }?,
attribute KentenAlignment { KentenAlignment_EnumValue }?,
attribute KentenPosition { RubyKentenPosition_EnumValue }?,
attribute KentenFontSize { xsd:double }?,
attribute KentenXScale { xsd:double }?,
attribute KentenYScale { xsd:double }?,
attribute KentenCustomCharacter { xsd:string }?,
attribute KentenCharacterSet { KentenCharacterSet_EnumValue }?,
attribute RubyTint { xsd:double }?,
attribute RubyWeight { xsd:double }?,
attribute RubyOverprintFill { AdornmentOverprint_EnumValue }?,
attribute RubyOverprintStroke { AdornmentOverprint_EnumValue }?,
attribute RubyStrokeTint { xsd:double }?,
attribute RubyFontSize { xsd:double }?,
attribute RubyOpenTypePro { xsd:boolean }?,

```

```

        attribute RubyXScale { xsd:double }?,
        attribute RubyYScale { xsd:double }?,
        attribute RubyType { RubyTypes_EnumValue }?,
        attribute RubyAlignment { RubyAlignments_EnumValue }?,
        attribute RubyPosition { RubyKentenPosition_EnumValue }?,
        attribute RubyXOffset { xsd:double }?,
        attribute RubyYOffset { xsd:double }?,
        attribute RubyParentSpacing { RubyParentSpacing_EnumValue }?,
        attribute RubyAutoAlign { xsd:boolean }?,
        attribute RubyOverhang { xsd:boolean }?,
        attribute RubyAutoScaling { xsd:boolean }?,
        attribute RubyParentScalingPercent { xsd:double }?,
        attribute RubyParentOverhangAmount { RubyOverhang_EnumValue }?,
        attribute Warichu { xsd:boolean }?,
        attribute WarichuSize { xsd:double }?,
        attribute WarichuLines { xsd:short }?,
        attribute WarichuLineSpacing { xsd:double }?,
        attribute WarichuAlignment { WarichuAlignment_EnumValue }?,
        attribute WarichuCharsAfterBreak { xsd:short }?,
        attribute WarichuCharsBeforeBreak { xsd:short }?,
        attribute OTFProportionalMetrics { xsd:boolean }?,
        attribute OTFHVKana { xsd:boolean }?,
        attribute OTFRomanItalics { xsd:boolean }?,
        attribute ScaleAffectsLineHeight { xsd:boolean }?,
        attribute CjkGridTracking { xsd:boolean }?,
        attribute GlyphForm { AlternateGlyphForms_EnumValue }?,
        attribute GridAlignFirstLineOnly { xsd:boolean }?,
        attribute GridAlignment { GridAlignment_EnumValue }?,
        attribute GridGyoudori { xsd:short }?,
        attribute AutoTcy { xsd:short }?,
        attribute AutoTcyIncludeRoman { xsd:boolean }?,
        attribute KinsokuType { KinsokuType_EnumValue }?,
        attribute KinsokuHangType { KinsokuHangTypes_EnumValue }?,
        attribute BunriKinshi { xsd:boolean }?,
        attribute Rensuuji { xsd:boolean }?,
        attribute RotateSingleByteCharacters { xsd:boolean }?,
        attribute LeadingModel { LeadingModel_EnumValue }?,
        attribute RubyAutoTcyDigits { xsd:short }?,
        attribute RubyAutoTcyIncludeRoman { xsd:boolean }?,
        attribute RubyAutoTcyAutoScale { xsd:boolean }?,
        attribute TreatIdeographicSpaceAsSpace { xsd:boolean }?,
        attribute AllowArbitraryHyphenation { xsd:boolean }?,
        attribute ParagraphGyoudori { xsd:boolean }?,
        attribute NumberingExpression { xsd:string }?,
        attribute BulletsTextAfter { xsd:string }?,
        attribute NumberingLevel { xsd:int }?,
        attribute NumberingContinue { xsd:boolean }?,
        attribute NumberingStartAt { xsd:int }?,
        attribute NumberingApplyRestartPolicy { xsd:boolean }?,
        attribute BulletsAlignment { ListAlignment_EnumValue }?,
        attribute NumberingAlignment { ListAlignment_EnumValue }?,
        attribute BulletsAndNumberingListType { ListType_EnumValue }?,
        element Properties {
            element BasedOn {

```

```

(object_type, xsd:string) |
(string_type, xsd:string)
} ?&
element BalanceRaggedLines {
  (bool_type, xsd:boolean) |
  (enum_type, BalanceLinesStyle_EnumValue)
} ?&
element RuleAboveColor {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element RuleAboveGapColor {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element RuleAboveType {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element RuleBelowColor {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element RuleBelowGapColor {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element RuleBelowType {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element AllNestedStyles { list_type, element ListItem {
  record_type,
  (
    element AppliedCharacterStyle { object_type, xsd:string } &
    element Delimiter {
      (string_type, xsd:string) |
      (enum_type, NestedStyleDelimiters_EnumValue)
    } &
    element Repetition { long_type, xsd:int } &
    element Inclusive { bool_type, xsd:boolean })
  } *
} ?&
element TabList { list_type, element ListItem {
  record_type,
  (
    element Alignment { enum_type, TabStopAlignment_EnumValue } &
    element AlignmentCharacter { string_type, xsd:string } &
    element Leader { string_type, xsd:string } &
    element Position { unit_type, xsd:double })
  } *
} ?&
element AppliedFont {
  (object_type, xsd:string) |

```

```

        (string_type, xsd:string )
    }?&
    element Leading {
        (unit_type, xsd:double ) |
        (enum_type, Leading_EnumValue )
    }?&
    element UnderlineColor {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element UnderlineGapColor {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element UnderlineType {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element StrikeThroughColor {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element StrikeThroughGapColor {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element StrikeThroughType {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element PreviewColor {
        (InDesignUIColorType_TypeDef ) |
        (enum_type, NothingEnum_EnumValue )
    }?&
    element KentenFillColor {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element KentenStrokeColor {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element KentenFont {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element KentenFontStyle {
        (string_type, xsd:string ) |
        (enum_type, NothingEnum_EnumValue )
    }?&
    element RubyFill {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&

```

```

element RubyStroke {
    (object_type, xsd:string) |
    (string_type, xsd:string)
} ?&
element RubyFont {
    (object_type, xsd:string) |
    (string_type, xsd:string)
} ?&
element RubyFontStyle {
    (string_type, xsd:string) |
    (enum_type, NothingEnum_EnumValue)
} ?&
element KinsokuSet {
    (object_type, xsd:string) |
    (enum_type, KinsokuSet_EnumValue) |
    (string_type, xsd:string)
} ?&
element Mojikumi {
    (object_type, xsd:string) |
    (string_type, xsd:string) |
    (enum_type, MojikumiTableDefaults_EnumValue)
} ?&
element BulletsFont {
    (object_type, xsd:string) |
    (string_type, xsd:string) |
    (enum_type, AutoEnum_EnumValue)
} ?&
element BulletsFontStyle {
    (string_type, xsd:string) |
    (enum_type, NothingEnum_EnumValue) |
    (enum_type, AutoEnum_EnumValue)
} ?&
element BulletsCharacterStyle {
    (object_type, xsd:string) |
    (string_type, xsd:string)
} ?&
element NumberingCharacterStyle {
    (object_type, xsd:string) |
    (string_type, xsd:string)
} ?&
element AppliedNumberingList {
    (object_type, xsd:string) |
    (string_type, xsd:string)
} ?&
element NumberingFormat {
    (enum_type, NumberingStyle_EnumValue) |
    (string_type, xsd:string)
} ?&
element NumberingRestartPolicies {
    attribute RestartPolicy { RestartPolicy_EnumValue },
    attribute LowerLevel { xsd:int },
    attribute UpperLevel { xsd:int }
} ?&
element BulletChar {

```

```

        attribute BulletCharacterType { BulletCharacterType_EnumValue },
        attribute BulletCharacterValue { xsd:int }
    }?&
    element Label { element KeyValuePair { KeyValuePair_TypeDef }* }
    ?
    ,
    (
        NestedLineStyle_Object*&
        NestedGrepStyle_Object*
    )
}

```

Most of the attributes and elements of a <ParagraphStyle> element are shared with all other text elements. Refer to “Common Text Properties.” The following tables describe the attributes and elements of a <ParagraphStyle> element that are not shared with all text objects.

**Table 113. Paragraph Style Properties Represented as Attributes**

Name	Type	Req	Description
Imported	Boolean	no	If true, the paragraph style was imported from another file.
KeyboardShortcut	string	no	The keyboard shortcut for the style.
NextStyle	string	no	A reference to the next paragraph style for the paragraph style, as a unique ID (as the value of the Self attribute of the <ParagraphStyle> element). The next paragraph style determines the style that is applied to the next paragraph when you enter a return at the end of a paragraph of this style.

**Table 114. Paragraph Style Properties Represented as Elements**

Name	Type	Req	Description
BasedOn	string	no	A reference to the paragraph style that this paragraph style is based on, as a unique ID (the value of the Self attribute of the <ParagraphStyle> element), or as a reference to the default “[No paragraph style]” element, in the form: \$ID/ [No paragraph style]. This is the default.

**IDML Example 78. BasedOn “[No paragraph style]”**

```

<ParagraphStyle Self="ParagraphStyle\Code" Name="Code" Imported="false" Next-
Style="ParagraphStyle\Code">
    <Properties>
        <BasedOn type="string">$ID/ [No paragraph style]</BasedOn>
    </Properties>
</ParagraphStyle>

```

**IDML Example 79. BasedOn Another Style**

```
<ParagraphStyle Self="ParagraphStyle\ListStyles%3aListFirst" Name="ListFirst"
```

```
Imported="false" NextStyle="ParagraphStyle\ListStyles%3aList">
<Properties>
    <BasedOn type="object">ParagraphStyle\ListStyles%3aList</BasedOn>
</Properties>
</ParagraphStyle>
```

### 6.4.3 Nested Styles

Nested styles are a feature of InDesign's paragraph formatting, and can be applied as local formatting or as part of a paragraph style. Using nested styles, you can specify character-level formatting for one or more ranges of text within a paragraph. Nested styles have two parts: a delimiter that sets the end of the text range affected, and a character style to apply to the range of text defined by the delimiter.

Nested styles are especially useful for run-in headings. For example, you can apply one character style to the first letter in a paragraph and another character style that takes effect through the first colon (:). For each nested style, you can define a character that ends the style, such as a tab character or the end of a word.

The nested style schema is included in a number of text objects (most notably the `<ParagraphStyle>` and `<ParagraphStyleRange>` elements).

#### Schema Example 113. NestedStyle

```
element AllNestedStyles { list_type, element ListItem {
    record_type,
    (
        element AppliedCharacterStyle { object_type, xsd:string }&
        element Delimiter {
            (string_type, xsd:string ) |
            (enum_type, NestedStyleDelimiters_EnumValue )
        }&
        element Repetition { long_type, xsd:int }&
        element Inclusive { bool_type, xsd:boolean })
    }*
}*&
```

**Table 115. NestedStyle Properties Represented as Elements**

Name	Type	Req	Description
AppliedCharacter-Style	string	yes	The character style applied to the nested style, as a reference to the Self attribute of the character style.
Delimiter	string or Nested-StyleDelimiters_EnumValue	yes	The delimiter character for the nested style.
Repetition	int	yes	The number of instances of the delimiter required to complete the nested style.
Inclusive	boolean	yes	If true, the formatting of the nested style is applied to the delimiter character.

The following example shows how a character style and a paragraph style can work together to format text using a nested style.

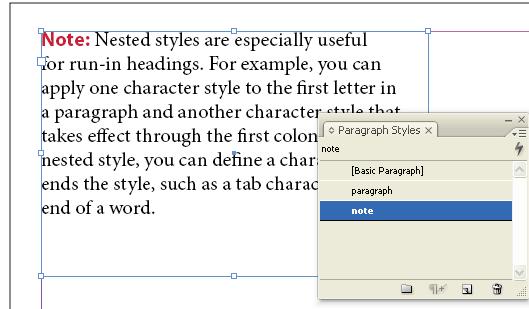
**IDML Example 80. NestedStyle**

```

<CharacterStyle Self="CharacterStyle>Note" Name="Note"
FillColor="Color/C=15 M=100 Y=100 K=0" FontStyle="Bold">
<Properties>
  <BasedOn type="string">$ID/[No character style]</BasedOn>
  <PreviewColor type="enumeration">Nothing</PreviewColor>
  <AppliedFont type="string">Myriad Pro</AppliedFont>
</Properties>
</CharacterStyle>
<ParagraphStyle Self="ParagraphStyle/note" Name="note"
NextStyle="ParagraphStyle/paragraph">
<Properties>
  <BasedOn type="object">ParagraphStyle/paragraph</BasedOn>
  <PreviewColor type="enumeration">Nothing</PreviewColor>
  <AllNestedStyles type="list">
    <ListItem type="record">
      <AppliedCharacterStyle type="object">CharacterStyle/Note
      </AppliedCharacterStyle>
      <Delimiter type="string">:</Delimiter>
      <Repetition type="long">1</Repetition>
      <Inclusive type="boolean">true</Inclusive>
    </ListItem>
  </AllNestedStyles>
</Properties>
</ParagraphStyle>

```

Figure 55. Nested Style

**6.4.4 Character Styles**

A character style is a collection of character formatting attributes that can be applied to text in a single step. Unlike paragraph styles, character styles do not necessarily define every formatting attribute that can be applied to a range of text. Instead, character styles typically contain only formatting which differs from that of the surrounding text.

Character styles in IDML are represented the same way as paragraph styles: each document contains a `<RootCharacterStyleGroup>` element, which, in turn, contains all of the `<CharacterStyle>` and `<CharacterStyleGroup>` elements used in the document described by the IDML package. The `<RootCharacterStyleGroup>` and `<CharacterStyleGroup>` elements have the same structure.

**Schema Example 114. CharacterStyleGroup**

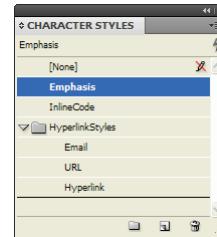
```
CharacterStyleGroup_Object = element CharacterStyleGroup {
    attribute Self { xsd:string },
    attribute Name { xsd:string }?,
    element Properties {
        element Label { element KeyValuePair { KeyValuePair_TypeDef }* }
    }?
}
?
'
(
    CharacterStyle_Object*&
    CharacterStyleGroup_Object*
)
}
```

**IDML Example 81. CharacterStyleGroup**

```
<RootCharacterStyleGroup Self="u6a">
    <CharacterStyle Self="CharacterStyle\[No character style]" Imported="false" Name="$ID/[No character style]"/>
    <CharacterStyle Self="CharacterStyle\Emphasis" Imported="false" Name="Emphasis" FontStyle="Italic">
        <Properties>
            <BasedOn type="string">$ID/[No character style]</BasedOn>
        </Properties>
    </CharacterStyle>
    <CharacterStyle Self="CharacterStyle\cInlineCode" Imported="false" Name="InlineCode" FontStyle="Regular">
        <Properties>
            <BasedOn type="string">$ID/[No character style]</BasedOn>
            <AppliedFont type="string">Droid Sans Mono</AppliedFont>
        </Properties>
    </CharacterStyle>
    <CharacterStyleGroup Self="CharacterStyleGroup\HyperlinkStyles" Name="$ID/HyperlinkStyles">
        <CharacterStyle Self="CharacterStyle\HyperlinkStyles%3aEmail" Imported="false" Name="HyperlinkStyles:Email">
            <Properties>
                <BasedOn type="object">CharacterStyle\HyperlinkStyles\Hyperlink</BasedOn>
            </Properties>
        </CharacterStyle>
        <CharacterStyle Self="CharacterStyle\HyperlinkStyles\URL" Imported="false" Name="HyperlinkStyles:URL">
            <Properties>
                <BasedOn type="object">CharacterStyle\HyperlinkStyles\Hyperlink</BasedOn>
            </Properties>
        </CharacterStyle>
        <CharacterStyle Self="CharacterStyle\HyperlinkStyles\Hyperlink" Imported="false" Name="HyperlinkStyles:Hyperlink">
            <Properties>
                <BasedOn type="string">$ID/[No character style]</BasedOn>
            </Properties>
        </CharacterStyle>
    </CharacterStyleGroup>
```

```
</RootCharacterStyleGroup>
```

Figure 56. Character Styles

**Schema Example 115. CharacterStyle**

```
CharacterStyle_Object = element CharacterStyle {
    attribute Self { xsd:string },
    attribute Imported { xsd:boolean }?,
    attribute FontStyle { xsd:string }?,
    attribute PointSize { xsd:double }?,
    attribute KerningMethod { xsd:string }?,
    attribute Tracking { xsd:double }?,
    attribute Capitalization { Capitalization_EnumValue }?,
    attribute Position { Position_EnumValue }?,
    attribute Underline { xsd:boolean }?,
    attribute StrikeThru { xsd:boolean }?,
    attribute Ligatures { xsd:boolean }?,
    attribute NoBreak { xsd:boolean }?,
    attribute HorizontalScale { xsd:double }?,
    attribute VerticalScale { xsd:double }?,
    attribute BaselineShift { xsd:double }?,
    attribute Skew { xsd:double }?,
    attribute FillTint { xsd:double }?,
    attribute StrokeTint { xsd:double }?,
    attribute StrokeWeight { xsd:double }?,
    attribute OverprintStroke { xsd:boolean }?,
    attribute OverprintFill { xsd:boolean }?,
    attribute OTFFigureStyle { OTFFigureStyle_EnumValue }?,
    attribute OTFOrdinal { xsd:boolean }?,
    attribute OTFFraction { xsd:boolean }?,
    attribute OTFDiscretionaryLigature { xsd:boolean }?,
    attribute OTFTitling { xsd:boolean }?,
    attribute OTFContextualAlternate { xsd:boolean }?,
    attribute OTFSwash { xsd:boolean }?,
    attribute UnderlineTint { xsd:double }?,
    attribute UnderlineGapTint { xsd:double }?,
    attribute UnderlineOverprint { xsd:boolean }?,
    attribute UnderlineGapOverprint { xsd:boolean }?,
    attribute UnderlineOffset { xsd:double }?,
    attribute UnderlineWeight { xsd:double }?,
    attribute StrikeThroughTint { xsd:double }?,
    attribute StrikeThroughGapTint { xsd:double }?,
    attribute StrikeThroughOverprint { xsd:boolean }?,
    attribute StrikeThroughGapOverprint { xsd:boolean }?,
    attribute StrikeThroughOffset { xsd:double }?,
    attribute StrikeThroughWeight { xsd:double }?,
```

```

attribute FillColor { xsd:string }?,
attribute StrokeColor { xsd:string }?,
attribute AppliedLanguage { xsd:string }?,
attribute KeyboardShortcut { list { xsd:short ,xsd:short } }?,
attribute OTFSashedZero { xsd:boolean }?,
attribute OTFHistorical { xsd:boolean }?,
attribute OTFStylisticSets { xsd:int }?,
attribute GradientFillLength { xsd:double }?,
attribute GradientFillAngle { xsd:double }?,
attribute GradientStrokeLength { xsd:double }?,
attribute GradientStrokeAngle { xsd:double }?,
attribute GradientFillStart { UnitPointType_TypeDef }?,
attribute GradientStrokeStart { UnitPointType_TypeDef }?,
attribute OTFMark { xsd:boolean }?,
attribute OTFLocale { xsd:boolean }?,
attribute PositionalForm { PositionalForms_EnumValue }?,
attribute Name { xsd:string },
attribute MiterLimit { xsd:double {minInclusive="1" maxInclusive="500"} }?,
attribute StrokeAlignment { TextStrokeAlign_EnumValue }?,
attribute EndJoin { OutlineJoin_EnumValue }?,
attribute OTFOverlapSwash { xsd:boolean }?,
attribute OTFStylisticAlternate { xsd:boolean }?,
attribute OTFJustificationAlternate { xsd:boolean }?,
attribute OTFStretchedAlternate { xsd:boolean }?,
attribute CharacterDirection { CharacterDirection_EnumValue }?,
attribute KeyboardDirection { CharacterDirection_EnumValue }?,
attribute DigitsType { DigitsType_EnumValue }?,
attribute Kashidas { Kashidas_EnumValue }?,
attribute DiacriticPosition { DiacriticPosition_EnumValue }?,
attribute XOffsetDiacritic { xsd:double }?,
attribute YOffsetDiacritic { xsd:double }?,
attribute GotoNextX { GotoNextX_EnumValue }?,
attribute PageNumberType { PageNumberType_EnumValue }?,
attribute CharacterAlignment { CharacterAlignment_EnumValue }?,
attribute Tsume { xsd:double }?,
attribute LeadingAki { xsd:double }?,
attribute TrailingAki { xsd:double }?,
attribute CharacterRotation { xsd:double }?,
attribute Jidori { xsd:short }?,
attribute ShataiMagnification { xsd:double }?,
attribute ShataiDegreeAngle { xsd:double }?,
attribute ShataiAdjustRotation { xsd:boolean }?,
attribute ShataiAdjustTsume { xsd:boolean }?,
attribute Tatechuyoko { xsd:boolean }?,
attribute TatechuyokoXOffset { xsd:double }?,
attribute TatechuyokoYOffset { xsd:double }?,
attribute KentenTint { xsd:double }?,
attribute KentenStrokeTint { xsd:double }?,
attribute KentenWeight { xsd:double }?,
attribute KentenOverprintFill { AdornmentOverprint_EnumValue }?,
attribute KentenOverprintStroke { AdornmentOverprint_EnumValue }?,
attribute KentenKind { KentenCharacter_EnumValue }?,
attribute KentenPlacement { xsd:double }?,
attribute KentenAlignment { KentenAlignment_EnumValue }?,

```

```

        attribute KentenPosition { RubyKentenPosition_EnumValue }?,
        attribute KentenFontSize { xsd:double }?,
        attribute KentenXScale { xsd:double }?,
        attribute KentenYScale { xsd:double }?,
        attribute KentenCustomCharacter { xsd:string }?,
        attribute KentenCharacterSet { KentenCharacterSet_EnumValue }?,
        attribute RubyTint { xsd:double }?,
        attribute RubyWeight { xsd:double }?,
        attribute RubyOverprintFill { AdornmentOverprint_EnumValue }?,
        attribute RubyOverprintStroke { AdornmentOverprint_EnumValue }?,
        attribute RubyStrokeTint { xsd:double }?,
        attribute RubyFontSize { xsd:double }?,
        attribute RubyOpenTypePro { xsd:boolean }?,
        attribute RubyXScale { xsd:double }?,
        attribute RubyYScale { xsd:double }?,
        attribute RubyType { RubyTypes_EnumValue }?,
        attribute RubyAlignment { RubyAlignments_EnumValue }?,
        attribute RubyPosition { RubyKentenPosition_EnumValue }?,
        attribute RubyXOffset { xsd:double }?,
        attribute RubyYOffset { xsd:double }?,
        attribute RubyParentSpacing { RubyParentSpacing_EnumValue }?,
        attribute RubyAutoAlign { xsd:boolean }?,
        attribute RubyOverhang { xsd:boolean }?,
        attribute RubyAutoScaling { xsd:boolean }?,
        attribute RubyParentScalingPercent { xsd:double }?,
        attribute RubyParentOverhangAmount { RubyOverhang_EnumValue }?,
        attribute Warichu { xsd:boolean }?,
        attribute WarichuSize { xsd:double }?,
        attribute WarichuLines { xsd:short }?,
        attribute WarichuLineSpacing { xsd:double }?,
        attribute WarichuAlignment { WarichuAlignment_EnumValue }?,
        attribute WarichuCharsAfterBreak { xsd:short }?,
        attribute WarichuCharsBeforeBreak { xsd:short }?,
        attribute OTFProportionalMetrics { xsd:boolean }?,
        attribute OTFHVKana { xsd:boolean }?,
        attribute OTFRomanItalics { xsd:boolean }?,
        attribute ScaleAffectsLineHeight { xsd:boolean }?,
        attribute CjkGridTracking { xsd:boolean }?,
        attribute GlyphForm { AlternateGlyphForms_EnumValue }?,
        attribute RubyAutoTcyDigits { xsd:short }?,
        attribute RubyAutoTcyIncludeRoman { xsd:boolean }?,
        attribute RubyAutoTcyAutoScale { xsd:boolean }?,
        element Properties {
            element BasedOn {
                (object_type, xsd:string) |
                (string_type, xsd:string)
            }?&
            element AppliedFont {
                (object_type, xsd:string) |
                (string_type, xsd:string)
            }?&
            element Leading {
                (unit_type, xsd:double) |
                (enum_type, Leading_EnumValue)
            }
        }
    
```

```

} ?&
element UnderlineColor {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element UnderlineGapColor {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element UnderlineType {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element StrikeThroughColor {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element StrikeThroughGapColor {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element StrikeThroughType {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element PreviewColor {
  (InDesignUIColorType_TypeDef) |
  (enum_type, NothingEnum_EnumValue)
} ?&
element KentenFillColor {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element KentenStrokeColor {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element KentenFont {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element KentenFontStyle {
  (string_type, xsd:string) |
  (enum_type, NothingEnum_EnumValue)
} ?&
element RubyFill {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element RubyStroke {
  (object_type, xsd:string) |
  (string_type, xsd:string)
} ?&
element RubyFont {

```

```

        (object_type, xsd:string) |
        (string_type, xsd:string)
    }?&
    element RubyFontStyle {
        (string_type, xsd:string) |
        (enum_type, NothingEnum_EnumValue)
    }?&
    element Label { element KeyValuePair { KeyValuePair_TypeDef }*
    }?
}
?
}
}

```

Most of the attributes and elements of a <CharacterStyle> element are shared with all other text elements. Refer to “Common Text Properties.” The following tables describe the attributes and elements of a <CharacterStyle> element that are not shared with all text objects.

**Table 116. Character Style Properties Represented as Attributes**

Name	Type	Req	Description
Imported	Boolean	no	If true, the paragraph style was imported from another file.
KeyboardShortcut	string	no	The keyboard shortcut for the style.
Name	string	yes	The name of the character style.

**Table 117. Character Style Properties Represented as Elements**

Name	Type	Req	Description
BasedOn	string	no	A reference to the character style that this character style is based on, as a unique ID (the value of the Self attribute of the <CharacterStyle> element), or as a reference to the default “[No character style]” element, in the form: \$ID/ [No character style]. This is the default.

#### 6.4.5 TableStyles

A table style is a collection of table formatting attributes, such as table borders and row and column strokes, that can be applied in a single step. Table styles can be organized into table style groups. In an IDML package, <TableStyle> elements are saved inside <TableStyleGroup> elements. All <TableStyle> and <TableStyleGroup> elements exist with the <RootTableStyleGroup> element. <TableStyleGroup> elements can contain both <TableStyle> elements and <TableStyleGroup> elements.

##### Schema Example 116. TableStyleGroup

```

TableStyleGroup_Object = element TableStyleGroup {
    attribute Self { xsd:string },
    attribute Name { xsd:string },
    element Properties {
        element Label { element KeyValuePair { KeyValuePair_TypeDef }*
        }?
    }
}

```

```

}
?

,
(
  TableStyle_Object*&
  TableStyleGroup_Object*
)
}
}

```

**Schema Example 117. TableStyle**

```

TableStyle_Object = element TableStyle {
  attribute Self { xsd:string },
  attribute Name { xsd:string },
  attribute StrokeOrder { StrokeOrderTypes_EnumValue }?,
  attribute TopBorderStrokeWeight { xsd:double }?,
  attribute TopBorderStrokeType { xsd:string }?,
  attribute TopBorderStrokeColor { xsd:string }?,
  attribute TopBorderStrokeTint { xsd:double }?,
  attribute TopBorderStrokeOverprint { xsd:boolean }?,
  attribute TopBorderStrokeGapColor { xsd:string }?,
  attribute TopBorderStrokeGapTint { xsd:double }?,
  attribute TopBorderStrokeGapOverprint { xsd:boolean }?,
  attribute LeftBorderStrokeWeight { xsd:double }?,
  attribute LeftBorderStrokeType { xsd:string }?,
  attribute LeftBorderStrokeColor { xsd:string }?,
  attribute LeftBorderStrokeTint { xsd:double }?,
  attribute LeftBorderStrokeOverprint { xsd:boolean }?,
  attribute LeftBorderStrokeGapColor { xsd:string }?,
  attribute LeftBorderStrokeGapTint { xsd:double }?,
  attribute LeftBorderStrokeGapOverprint { xsd:boolean }?,
  attribute BottomBorderStrokeWeight { xsd:double }?,
  attribute BottomBorderStrokeType { xsd:string }?,
  attribute BottomBorderStrokeColor { xsd:string }?,
  attribute BottomBorderStrokeTint { xsd:double }?,
  attribute BottomBorderStrokeOverprint { xsd:boolean }?,
  attribute BottomBorderStrokeGapColor { xsd:string }?,
  attribute BottomBorderStrokeGapTint { xsd:double }?,
  attribute BottomBorderStrokeGapOverprint { xsd:boolean }?,
  attribute RightBorderStrokeWeight { xsd:double }?,
  attribute RightBorderStrokeType { xsd:string }?,
  attribute RightBorderStrokeColor { xsd:string }?,
  attribute RightBorderStrokeTint { xsd:double }?,
  attribute RightBorderStrokeOverprint { xsd:boolean }?,
  attribute RightBorderStrokeGapColor { xsd:string }?,
  attribute RightBorderStrokeGapTint { xsd:double }?,
  attribute RightBorderStrokeGapOverprint { xsd:boolean }?,
  attribute SpaceBefore { xsd:double }?,
  attribute SpaceAfter { xsd:double }?,
  attribute SkipFirstAlternatingStrokeRows { xsd:int }?,
  attribute SkipLastAlternatingStrokeRows { xsd:int }?,
  attribute StartRowStrokeCount { xsd:int }?,
  attribute StartRowStrokeColor { xsd:string }?,
  attribute StartRowStrokeWeight { xsd:double }?,
  attribute StartRowStrokeType { xsd:string }?
}

```

```

        attribute StartRowStrokeTint { xsd:double }?,
        attribute StartRowStrokeGapOverprint { xsd:boolean }?,
        attribute StartRowStrokeGapColor { xsd:string }?,
        attribute StartRowStrokeGapTint { xsd:double }?,
        attribute StartRowStrokeOverprint { xsd:boolean }?,
        attribute EndRowStrokeCount { xsd:int }?,
        attribute EndRowStrokeColor { xsd:string }?,
        attribute EndRowStrokeWeight { xsd:double }?,
        attribute EndRowStrokeType { xsd:string }?,
        attribute EndRowStrokeTint { xsd:double }?,
        attribute EndRowStrokeOverprint { xsd:boolean }?,
        attribute EndRowStrokeGapColor { xsd:string }?,
        attribute EndRowStrokeGapTint { xsd:double }?,
        attribute EndRowStrokeGapOverprint { xsd:boolean }?,
        attribute SkipFirstAlternatingStrokeColumns { xsd:int }?,
        attribute SkipLastAlternatingStrokeColumns { xsd:int }?,
        attribute StartColumnStrokeCount { xsd:int }?,
        attribute StartColumnStrokeColor { xsd:string }?,
        attribute StartColumnStrokeWeight { xsd:double }?,
        attribute StartColumnStrokeType { xsd:string }?,
        attribute StartColumnStrokeTint { xsd:double }?,
        attribute StartColumnStrokeOverprint { xsd:boolean }?,
        attribute StartColumnStrokeGapColor { xsd:string }?,
        attribute StartColumnStrokeGapTint { xsd:double }?,
        attribute StartColumnStrokeGapOverprint { xsd:boolean }?,
        attribute EndColumnStrokeCount { xsd:int }?,
        attribute EndColumnStrokeColor { xsd:string }?,
        attribute EndColumnStrokeWeight { xsd:double }?,
        attribute EndColumnLineStyle { xsd:string }?,
        attribute EndColumnStrokeTint { xsd:double }?,
        attribute EndColumnStrokeOverprint { xsd:boolean }?,
        attribute EndColumnStrokeGapColor { xsd:string }?,
        attribute EndColumnStrokeGapTint { xsd:double }?,
        attribute EndColumnStrokeGapOverprint { xsd:boolean }?,
        attribute ColumnFillsPriority { xsd:boolean }?,
        attribute SkipFirstAlternatingFillRows { xsd:int }?,
        attribute SkipLastAlternatingFillRows { xsd:int }?,
        attribute StartRowFillColor { xsd:string }?,
        attribute StartRowFillCount { xsd:int }?,
        attribute StartRowFillTint { xsd:double }?,
        attribute StartRowFillOverprint { xsd:boolean }?,
        attribute EndRowFillCount { xsd:int }?,
        attribute EndRowFillColor { xsd:string }?,
        attribute EndRowFillTint { xsd:double }?,
        attribute EndRowFillOverprint { xsd:boolean }?,
        attribute SkipFirstAlternatingFillColumns { xsd:int }?,
        attribute SkipLastAlternatingFillColumns { xsd:int }?,
        attribute StartColumnFillCount { xsd:int }?,
        attribute StartColumnFillColor { xsd:string }?,
        attribute StartColumnFillTint { xsd:double }?,
        attribute StartColumnFillOverprint { xsd:boolean }?,
        attribute EndColumnFillCount { xsd:int }?,
        attribute EndColumnFillColor { xsd:string }?,
        attribute EndColumnFillTint { xsd:double }?,

```

```

attribute EndColumnFillOverprint { xsd:boolean }?,
attribute HeaderRegionSameAsBodyRegion { xsd:boolean }?,
attribute FooterRegionSameAsBodyRegion { xsd:boolean }?,
attribute LeftColumnRegionSameAsBodyRegion { xsd:boolean }?,
attribute RightColumnRegionSameAsBodyRegion { xsd:boolean }?,
attribute HeaderRegionCellStyle { xsd:string }?,
attribute FooterRegionCellStyle { xsd:string }?,
attribute LeftColumnRegionCellStyle { xsd:string }?,
attribute RightColumnRegionCellStyle { xsd:string }?,
attribute BodyRegionCellStyle { xsd:string }?,
attribute KeyboardShortcut { list { xsd:short ,xsd:short } }?,
element Properties {
    element BasedOn {
        (object_type, xsd:string ) |
        (string_type, xsd:string )
    }?&
    element Label { element KeyValuePair { KeyValuePair_TypeDef }* }
}?
}
?
}

```

**Table 118. TableStyle Properties Represented as Attributes**

Name	Type	Req	Description
BodyRegionCell-Style	string	no	The cell style of the body (i.e., not header or footer) region.
BottomBorder-StrokeColor	string	no	The color, specified as a swatch (color, gradient, tint, or mixed ink), of the bottom border stroke.
BottomBorder-StrokeGapColor	string	no	The gap color, specified as a swatch (color, gradient, tint, or mixed ink), of the bottom border stroke. Note: Valid only when bottom border stroke type is not solid.
BottomBorder-StrokeGap-Overprint	boolean	no	If true, the gap of the bottom border stroke will overprint. Note: Valid only when bottom border stroke type is not solid.
BottomBorder-StrokeGapTint	double	no	The tint (as a percentage) of the gap color of the bottom border stroke. (Range: 0 to 100) Note: Valid only when bottom border stroke type is not solid.
BottomBorder-StrokeOverprint	boolean	no	If true, the bottom border stroke will overprint.
BottomBorder-StrokeTint	double	no	The tint (as a percentage) of the bottom border stroke. (Range: 0 to 100)
BottomBorder-StrokeType	string	no	The stroke type of the bottom border.
BottomBorder-StrokeWeight	double	no	The stroke weight of the bottom border stroke.
ColumnFills-Priority	boolean	no	If true, hides alternating row fills. If false, hides alternating column fills.

Name	Type	Req	Description
EndColumnFill-Color	string	no	The fill color, specified as a swatch (color, gradient, tint, or mixed ink), of columns in the second alternating fill group. Note: Valid when alternating fills are defined for table columns.
EndColumnFill-Count	int	no	The number of columns in the second alternating fills group. Note: Valid when alternating fills are defined for table columns.
EndColumnFill-Overprint	boolean	no	If true, the columns in the second alternating fills group will overprint. Note: Valid when alternating fills are defined for table columns.
EndColumnFillTint	double	no	The tint (as a percentage) of the columns in the second alternating fills group. (Range: 0 to 100) Note: Valid when alternating fills are defined for table columns.
EndColumnLineStyle	string	no	The stroke type of columns in the second alternating strokes group.
EndColumnStroke-Color	string	no	The stroke color, specified as a swatch (color, gradient, tint, or mixed ink), of column borders in the second alternating column strokes group. Note: Valid when end column stroke count is 1 or greater.
EndColumnStroke-Count	int	no	The number of columns in the second alternating column strokes group.
EndColumnStroke-GapColor	string	no	The stroke gap color, specified as a swatch (color, gradient, tint, or mixed ink), of column borders in the second alternating column strokes group. Note: Valid when end column stroke count is 1 or greater.
EndColumnStroke-GapOverprint	boolean	no	If true, the gap of the column border stroke in the second alternating column strokes group will overprint. Note: Valid when end column stroke count is 1 or greater.
EndColumnStroke-GapTint	double	no	The tint (as a percentage) of the gap color of column borders in the second alternating column strokes group. (Range: 0 to 100) Note: Valid when end column stroke count is 1 or greater.
EndColumnStroke-Overprint	boolean	no	If true, the column borders in the second alternating column strokes group will overprint. Note: Valid when end column stroke count is 1 or greater.
EndColumnStroke-Tint	double	no	The tint (as a percentage) of column borders in the second alternating column strokes group. (Range: 0 to 100) Note: Valid when end column stroke count is 1 or greater.
EndColumnStroke-Weight	double	no	The stroke weight of column borders in the second alternating column strokes group. Note: Valid when end column stroke count is 1 or greater.

Name	Type	Req	Description
EndRowFillColor	string	no	The fill color, specified as a swatch (color, gradient, tint, or mixed ink), of rows in the second alternating fills group. Note: Valid when alternating fills are defined for table rows.
EndRowFillCount	int	no	The number of rows in the second alternating fills group. Note: Valid when alternating fills are defined for table rows.
EndRowFill-Overprint	boolean	no	If true, the rows in the second alternating fills group will overprint. Note: Valid when alternating fills are defined for table rows.
EndRowFillTint	double	no	The tint (as a percentage) of the rows in the second alternating fills group. (Range: 0 to 100) Note: Valid when alternating fills are defined for table rows.
EndRowStrokeColor	string	no	The stroke color, specified as a swatch (color, gradient, tint, or mixed ink), of row borders in the second alternating row strokes group. Note: Valid when end row stroke count is 1 or greater.
EndRowStrokeCount	int	no	The number of rows in the second alternating row strokes group.
EndRowStrokeGap-Color	string	no	The gap color, specified as a swatch (color, gradient, tint, or mixed ink), of row borders in the second alternating rows group. Note: Valid when end row stroke count is 1 or greater.
EndRowStrokeGap-Overprint	boolean	no	If true, the gap of the row borders in the second alternating rows group will overprint. Note: Valid when end row stroke count is 1 or greater.
EndRowStrokeGap-Tint	double	no	The tint (as a percentage) of the gap color of rows in the second alternating strokes group. (Range: 0 to 100) Note: Valid when end row stroke count is 1 or greater and end row stroke type is not solid.
EndRowStroke-Overprint	boolean	no	If true, the rows in the second alternating rows group will overprint. Note: Valid when end row stroke count is 1 or greater.
EndRowStrokeTint	double	no	The tint (as a percentage) of the row borders in the second alternating strokes group. (Range: 0 to 100) Note: Valid when end row stroke count is 1 or greater.
EndRowStrokeType	string	no	The stroke type of rows in the second alternating strokes group.
EndRowStroke-Weight	double	no	The stroke weight of row borders in the second alternating row strokes group. Note: Valid when end row stroke count is 1 or greater.
FooterRegionCellStyle	string	no	The cell style of the footer region.
FooterRegionSameAsBodyRegion	boolean	no	If true, uses the cell style of the body region for the footer region.

Name	Type	Req	Description
HeaderRegionCellStyle	string	no	The cell style of the header region.
HeaderRegionSameAsBodyRegion	boolean	no	If true, use the cell style of the body region for the header region.
KeyboardShortcut		no	
LeftBorderStroke-Color	string	no	The color, specified as a swatch (color, gradient, tint, or mixed ink), of the left border stroke.
LeftBorderStroke-GapColor	string	no	The gap color, specified as a swatch (color, gradient, tint, or mixed ink), of the left border stroke. Note: Valid only when left border stroke type is not solid.
LeftBorderStroke-GapOverprint	boolean	no	If true, the gap of the left border stroke will overprint. Note: Valid only when left border stroke type is not solid.
LeftBorderStroke-GapTint	double	no	The tint (as a percentage) of the gap color of the left border stroke. (Range: 0 to 100) Note: Valid only when left border stroke type is not solid.
LeftBorderStroke-Overprint	boolean	no	If true, the left border stroke will overprint.
LeftBorderStroke-Tint	double	no	The tint (as a percentage) of the left border stroke. (Range: 0 to 100)
LeftBorderStroke-Type	string	no	The stroke type of the left border.
LeftBorderStroke-Weight	double	no	The stroke weight of the left border stroke.
LeftColumnRegion-CellStyle	string	no	The cell style of the left column region.
LeftColumnRegion-SameAsBodyRegion	boolean	no	If true, uses the cell style of the body region for the left column region.
Name	string	yes	The name of the TableStyle.
RightBorder-StrokeColor	string	no	The color, specified as a swatch (color, gradient, tint, or mixed ink), of the right border stroke.
RightBorder-StrokeGapColor	string	no	The gap color, specified as a swatch (color, gradient, tint, or mixed ink), of the right border stroke. Note: Valid only when right border stroke type is not solid.
RightBorder-StrokeGap-Overprint	boolean	no	If true, the gap color of the right border stroke will overprint. Note: Valid only when right border stroke type is not solid.
RightBorder-StrokeGapTint	double	no	The tint (as a percentage) of the gap color of the right border stroke. (Range: 0 to 100) Note: Valid only when right border stroke type is not solid.
RightBorder-StrokeOverprint	boolean	no	If true, the right border stroke will overprint.
RightBorder-StrokeTint	double	no	The tint (as a percentage) of the right border stroke. (Range: 0 to 100)

Name	Type	Req	Description
RightBorder-StrokeType	string	no	The stroke type of the right border.
RightBorder-StrokeWeight	double	no	The stroke weight of the right border stroke.
RightColumn-RegionCellStyle	string	no	The cell style of the right column region.
RightColumn-RegionSameAsBody-Region	boolean	no	If true, uses the cell style of the body region for the right column region.
SkipFirst-AlternatingFill-Columns	int	no	The number of columns on the left side of the table to skip before applying the column fill color. Note: Valid when alternating fills are defined for table columns.
SkipFirst-AlternatingFill-Rows	int	no	The number of body rows at the beginning of the table to skip before applying the row fill color. Note: Valid when alternating fills are defined for table rows. For information on body rows, see body row count.
SkipFirst-Alternating-StrokeColumns	int	no	The number of columns on the left of the table in which to skip border stroke formatting. Note: Valid when start column stroke count is 1 or greater and/or end column stroke count is 1 or greater.
SkipFirst-Alternating-StrokeRows	int	no	The number of body rows at the beginning of the table in which to skip border stroke formatting. Note: Valid when start row stroke count is 1 or greater and/or end row stroke count is 1 or greater. For information on body rows, see body row count.
SkipLast-AlternatingFill-Columns	int	no	The number columns on the right side of the table in which to not apply the column fill color. Note: Valid when alternating fills are defined for table columns.
SkipLast-AlternatingFill-Rows	int	no	The number of body rows at the end of the table in which to not apply the row fill color. Note: Valid when alternating fills are defined for table rows. For information on body rows, see body row count.
SkipLast-Alternating-StrokeColumns	int	no	The number of columns on the right side of the table in which to skip border stroke formatting. Note: Valid when start column stroke count is 1 or greater and/or end column stroke count is 1 or greater.
SkipLast-Alternating-StrokeRows	int	no	The number of body rows at the end of the table in which to skip border stroke formatting. Note: Valid when start row stroke count is 1 or greater and/or end row stroke count is 1 or greater. For information on body rows, see body row count.
SpaceAfter	double	no	The space below the table.

Name	Type	Req	Description
SpaceBefore	double	no	The space above the table.
StartColumnFill-Color	string	no	The fill color, specified as a swatch (color, gradient, tint, or mixed ink), of columns in the first alternating fills group. Note: Valid when alternating fills are defined for table columns.
StartColumnFill-Count	int	no	The number of columns in the first alternating fills group. Note: Valid when alternating fills are defined for table columns.
StartColumnFill-Overprint	boolean	no	If true, the columns in the first alternating fills group will overprint. Note: Valid when alternating fills are defined for table columns.
StartColumnFill-Tint	double	no	The tint (as a percentage) of the columns in the first alternating fills group. (Range: 0 to 100) Note: Valid when alternating fills are defined for table columns.
StartColumn-StrokeColor	string	no	The stroke color, specified as a swatch (color, gradient, tint, or mixed ink), of column borders in the first alternating column strokes group.
StartColumn-StrokeCount	int	no	The number of columns in the first alternating column strokes group.
StartColumn-StrokeGapColor	string	no	The stroke gap color, specified as a swatch (color, gradient, tint, or mixed ink), of column borders in the first alternating column strokes group. Note: Valid when start column stroke count is 1 or greater.
StartColumn-StrokeGap-Overprint	boolean	no	If true, the gap of the column borders in the first alternating column strokes group will overprint. Note: Valid when start column stroke count is 1 or greater.
StartColumn-StrokeGapTint	double	no	The tint (as a percentage) of the gap color of column borders in the first alternating column strokes group. (Range: 0 to 100) Note: Valid when start column stroke count is 1 or greater.
StartColumn-StrokeOverprint	boolean	no	If true, the column borders in the first alternating column strokes group will overprint. Note: Valid when start column stroke count is 1 or greater.
StartColumn-StrokeTint	double	no	The tint (as a percentage) of column borders in the first alternating column strokes group. (Range: 0 to 100) Note: Valid when start column stroke count is 1 or greater.
StartColumn-StrokeType	string	no	The stroke type of columns in the first alternating strokes group.
StartColumn-StrokeWeight	double	no	The stroke weight of column borders in the first alternating column strokes group. Note: Valid when start column stroke count is 1 or greater.

Name	Type	Req	Description
StartRowFillColor	string	no	The fill color, specified as a swatch (color, gradient, tint, or mixed ink), of rows in the first alternating fills group. Note: Valid when alternating fills are defined for table rows.
StartRowFillCount	int	no	The number of rows in the first alternating fills group. Note: Valid when alternating fills are defined for table rows.
StartRowFill-Overprint	boolean	no	If true, the rows in the first alternating fills group will overprint. Note: Valid when alternating fills are defined for table rows.
StartRowFillTint	double	no	The tint (as a percentage) of the rows in the first alternating fills group. (Range: 0 to 100) Note: Valid when alternating fills are defined for table rows.
StartRowStroke-Color	string	no	The color, specified as a swatch (color, gradient, tint, or mixed ink), of row borders in the first alternating row strokes group. Note: Valid when start row stroke count is 1 or greater.
StartRowStroke-Count	int	no	The number of rows in the first alternating row strokes group.
StartRowStroke-GapColor	string	no	The stroke gap color of row borders in the first alternating row strokes group, specified as a swatch (color, gradient, tint, or mixed ink). Note: Valid when start row stroke count is 1 or greater.
StartRowStroke-GapOverprint	boolean	no	If true, the gap color of the row border stroke in the first alternating row strokes group will overprint. Note: Valid when start row stroke count is 1 or greater.
StartRowStroke-GapTint	double	no	The tint (as a percentage) of the gap color of row borders in the first alternating rows group. (Range: 0 to 100) Note: Valid when start row stroke count is 1 or greater.
StartRowStroke-Overprint	boolean	no	If true, the row borders in the first alternating row strokes group will overprint. Note: Valid when start row stroke count is 1 or greater.
StartRowStroke-Tint	double	no	The tint (as a percentage) of the borders in the first alternating row strokes group. (Range: 0 to 100) Note: Valid when start row stroke count is 1 or greater.
StartRowStroke-Type	string	no	The stroke type of rows in the first alternating strokes group.
StartRowStroke-Weight	double	no	The stroke weight of row borders in the first alternating row strokes group. Note: Valid when start row stroke count is 1 or greater.

Name	Type	Req	Description
StrokeOrder	StrokeOrderTypes_EnumValue	no	The order in which to display row and column strokes at corners. Can be RowOnTop (Places row strokes in front of column strokes), ColumnOnTop (Places column strokes in front of row strokes), BestJoins (Places row strokes in front of column strokes when row and column strokes are different colors; joins striped strokes and connects crossing points), or InDesign2Compatibility (Places row strokes in front when row and column strokes are different colors; joins striped strokes only at points where strokes cross in a T-shape).
TopBorderStroke-Color	string	no	The color, specified as a swatch (color, gradient, tint, or mixed ink), of the table's top border stroke.
TopBorderStroke-GapColor	string	no	The gap color, specified as a swatch (color, gradient, tint, or mixed ink), of the table's top border stroke. Note: Valid only when top border stroke type is not solid.
TopBorderStroke-GapOverprint	boolean	no	If true, the gap of the top border stroke will overprint. Note: Valid only when top border stroke type is not solid.
TopBorderStroke-GapTint	double	no	The tint (as a percentage) of the gap color of the table's top border stroke. (Range: 0 to 100) Note: Valid only when top border stroke type is not solid.
TopBorderStroke-Overprint	boolean	no	If true, the top border strokes will overprint.
TopBorderStroke-Tint	double	no	The tint (as a percentage) of the table's top border stroke. (Range: 0 to 100)
TopBorderStroke-Type	string	no	The stroke type of the top border.
TopBorderStroke-Weight	double	no	The stroke weight of the table's top border stroke.

**Table 119. TableStyle Properties Represented as Elements**

Name	Type	Req	Description
BasedOn	string	no	The table style that the table style is based on. Can be either: <code>type = "object"</code> A reference to the table style this table style is based on, as a unique ID (as the value of the <code>Self</code> attribute of the <code>&lt;TableStyle&gt;</code> element). or <code>type = "string"</code> A reference to the default “[No table style]” element, in the form: \$ID/[No table style]. This is the default.

#### 6.4.6 CellStyles

A cell style includes formatting such as cell insets, paragraph styles, and strokes and fills. Cell styles can be organized into cell style groups. In an IDML package, `<CellStyle>` elements are saved inside `<CellStyleGroup>` elements. All `<CellStyle>` and `<CellStyleGroup>` elements exist with the `<RootCellStyleGroup>` element. `<CellStyleGroup>` elements can contain both `<CellStyle>` elements and `<CellStyleGroup>` elements.

##### Schema Example 118. `CellStyleGroup`

```
CellStyleGroup_Object = element CellStyleGroup {
    attribute Self { xsd:string },
    attribute Name { xsd:string },
    element Properties {
        element Label { element KeyValuePair { KeyValuePair_TypeDef }* }
    }?
    ?
    ,
    (
        CellStyle_Object*&
        CellStyleGroup_Object*
    )
}
```

##### Schema Example 119. `CellStyle`

```
CellStyle_Object = element CellStyle {
    attribute Self { xsd:string },
    attribute AppliedParagraphStyle { xsd:string }?,
    attribute GradientFillLength { xsd:double }?,
    attribute GradientFillAngle { xsd:double }?,
    attribute GradientFillStart { UnitPointType_TypeDef }?,
    attribute TopInset { xsd:double }?,
    attribute LeftInset { xsd:double }?,
    attribute BottomInset { xsd:double }?,
    attribute RightInset { xsd:double }?,
    attribute FillColor { xsd:string }?,
    attribute FillTint { xsd:double }?,
    attribute OverprintFill { xsd:boolean }?,
    attribute TopLeftDiagonalLine { xsd:boolean }?,
    attribute TopRightDiagonalLine { xsd:boolean }?,
    attribute DiagonalLineInFront { xsd:boolean }?,
    attribute DiagonalLineStrokeWeight { xsd:double }?,
    attribute DiagonalLineStrokeType { xsd:string }?,
    attribute DiagonalLineStrokeColor { xsd:string }?,
    attribute DiagonalLineStrokeTint { xsd:double }?,
    attribute DiagonalLineStrokeOverprint { xsd:boolean }?,
    attribute DiagonalLineStrokeGapColor { xsd:string }?,
    attribute DiagonalLineStrokeGapTint { xsd:double }?,
    attribute DiagonalLineStrokeGapOverprint { xsd:boolean }?,
    attribute ClipContentToCell { xsd:boolean }?,
    attribute FirstBaselineOffset { FirstBaseline_EnumValue }?,
    attribute VerticalJustification { VerticalJustification_EnumValue }?,
    attribute ParagraphSpacingLimit { xsd:double }?,
    attribute MinimumFirstBaselineOffset { xsd:double {minInclusive="0" max-
```

```

Inclusive="8640"} }?,
    attribute RotationAngle { xsd:double }?,
    attribute LeftEdgeStrokeWeight { xsd:double }?,
    attribute LeftEdgeStrokeType { xsd:string }?,
    attribute LeftEdgeStrokeColor { xsd:string }?,
    attribute LeftEdgeStrokeTint { xsd:double }?,
    attribute LeftEdgeStrokeOverprint { xsd:boolean }?,
    attribute LeftEdgeStrokeGapColor { xsd:string }?,
    attribute LeftEdgeStrokeGapTint { xsd:double }?,
    attribute LeftEdgeStrokeGapOverprint { xsd:boolean }?,
    attribute TopEdgeStrokeWeight { xsd:double }?,
    attribute TopEdgeStrokeType { xsd:string }?,
    attribute TopEdgeStrokeColor { xsd:string }?,
    attribute TopEdgeStrokeTint { xsd:double }?,
    attribute TopEdgeStrokeOverprint { xsd:boolean }?,
    attribute TopEdgeStrokeGapColor { xsd:string }?,
    attribute TopEdgeStrokeGapTint { xsd:double }?,
    attribute TopEdgeStrokeGapOverprint { xsd:boolean }?,
    attribute RightEdgeStrokeWeight { xsd:double }?,
    attribute RightEdgeStrokeType { xsd:string }?,
    attribute RightEdgeStrokeColor { xsd:string }?,
    attribute RightEdgeStrokeTint { xsd:double }?,
    attribute RightEdgeStrokeOverprint { xsd:boolean }?,
    attribute RightEdgeStrokeGapColor { xsd:string }?,
    attribute RightEdgeStrokeGapTint { xsd:double }?,
    attribute RightEdgeStrokeGapOverprint { xsd:boolean }?,
    attribute BottomEdgeStrokeWeight { xsd:double }?,
    attribute BottomEdgeStrokeType { xsd:string }?,
    attribute BottomEdgeStrokeColor { xsd:string }?,
    attribute BottomEdgeStrokeTint { xsd:double }?,
    attribute BottomEdgeStrokeOverprint { xsd:boolean }?,
    attribute BottomEdgeStrokeGapColor { xsd:string }?,
    attribute BottomEdgeStrokeGapTint { xsd:double }?,
    attribute BottomEdgeStrokeGapOverprint { xsd:boolean }?,
    attribute KeyboardShortcut { list { xsd:short ,xsd:short } }?,
    attribute RotationRunsAgainstStory { xsd:boolean }?,
    attribute Name { xsd:string },
    element Properties {
        element BasedOn {
            (object_type, xsd:string ) |
            (string_type, xsd:string )
        }?&
        element Label { element KeyValuePair { KeyValuePair_TypeDef }* }
    }?
}
?
}

```

Most of the properties of a cell style are shared with the properties of a cell, see “Cell Properties Represented as Attributes.”

**Table 120. CellStyle Properties Represented as Elements**

Name	Type	Req	Description
BasedOn	string	no	A reference to the cell style that this cell style is based on, as a unique ID (the value of the Self attribute of the <CellStyle> element), or as a reference to the default “[No cell style]” element, in the form: \$ID/[No cell style]. This is the default.

#### 6.4.7 Object Styles

Just as you use paragraph and character styles to quickly format text, you can use object styles to quickly format graphics and frames. Object styles include settings for stroke, color, transparency, drop shadows, paragraph styles, text wrap, and more. You can assign different transparency effects for the object, fill, stroke, and text. Use object styles, rather than applying extensive local formatting, to format page items.

You can apply object styles to objects, groups, and frames (including text frames). A style can either clear and replace all object settings or it can replace only specific settings, leaving other settings unchanged. You control which settings the style affects by including or excluding a category of settings in the definition.

Object styles in an InDesign document can be organized into object style groups.

All InDesign documents contain a set of default object styles: “None,” “[Normal Graphics Frame],” “[Normal Text Frame],” and “[Normal Grid].” Apart from “None,” which cannot be edited, all of the default styles can be edited by the user, so you should define these object styles in your IDML if you plan to use them to format objects. The default object style “None” is an object style that does not apply any formatting. Use this style when you want to apply local formatting to page items.

For a description of the default objects styles, refer to the InDesign documentation.

In an IDML package, the `styles.xml` document contains a `<RootObjectStyleGroup>` element, which, in turn, contains all `<ObjectStyle>` and `<ObjectStyleGroup>` elements. An `<ObjectStyleGroup>` element can contain other `<ObjectStyleGroup>` elements.

##### Schema Example 120. ObjectStyleGroup

```
ObjectStyleGroup_Object = element ObjectStyleGroup {
    attribute Self { xsd:string },
    attribute Name { xsd:string },
    element Properties {
        element Label { element KeyValuePair { KeyValuePair_TypeDef }* }
        ?
    }
    ?
    (
        ObjectStyle_Object*&
        ObjectStyleGroup_Object*
    )
}
```

**Schema Example 121. ObjectStyle**

```

ObjectStyle_Object = element ObjectStyle {
    attribute Self { xsd:string },
    attribute Name { xsd:string },
    attribute AppliedParagraphStyle { xsd:string }?,
    attribute ApplyNextParagraphStyle { xsd:boolean }?,
    attribute EnableFill { xsd:boolean }?,
    attribute EnableStroke { xsd:boolean }?,
    attribute EnableParagraphStyle { xsd:boolean }?,
    attribute EnableTextFrameGeneralOptions { xsd:boolean }?,
    attribute EnableTextFrameBaselineOptions { xsd:boolean }?,
    attribute EnableStoryOptions { xsd:boolean }?,
    attribute EnableTextWrapAndOthers { xsd:boolean }?,
    attribute EnableAnchoredObjectOptions { xsd:boolean }?,
    attribute FillColor { xsd:string }?,
    attribute FillTint { xsd:double }?,
    attribute OverprintFill { xsd:boolean }?,
    attribute StrokeWeight { xsd:double }?,
    attribute MiterLimit { xsd:double {minInclusive="1" maxInclusive="500"} }?,
    attribute EndCap { EndCap_EnumValue }?,
    attribute EndJoin { EndJoin_EnumValue }?,
    attribute StrokeType { xsd:string }?,
    attribute StrokeCornerAdjustment { StrokeCornerAdjustment_EnumValue }?,
    attribute StrokeDashAndGap { list { xsd:double * } }?,
    attribute LeftLineEnd { ArrowHead_EnumValue }?,
    attribute RightLineEnd { ArrowHead_EnumValue }?,
    attribute StrokeColor { xsd:string }?,
    attribute StrokeTint { xsd:double }?,
    attribute CornerRadius { xsd:double }?,
    attribute OverprintStroke { xsd:boolean }?,
    attribute GapColor { xsd:string }?,
    attribute GapTint { xsd:double }?,
    attribute OverprintGap { xsd:boolean }?,
    attribute StrokeAlignment { StrokeAlignment_EnumValue }?,
    attribute Nonprinting { xsd:boolean }?,
    attribute GradientFillAngle { xsd:double }?,
    attribute GradientStrokeAngle { xsd:double }?,
    attribute AppliedNamedGrid { xsd:string }?,
    attribute KeyboardShortcut { list { xsd:short ,xsd:short } }?,
    attribute EnableFrameFittingOptions { xsd:boolean }?,
    attribute CornerOption { CornerOptions_EnumValue }?,
    attribute EnableStrokeAndCornerOptions { xsd:boolean }?,
    element Properties {
        element BasedOn {
            (object_type, xsd:string ) |
            (string_type, xsd:string )
        }?&
        element Label { element KeyValuePair { KeyValuePair_TypeDef }* }
    }?
}
?
,
(
    TextFramePreference_Object?&

```

```

BaselineFrameGridOption_Object?&
AnchoredObjectSetting_Object?&
TextWrapPreference_Object?&
StoryPreference_Object?&
FrameFittingOption_Object?&
TransparencySetting_Object?&
StrokeTransparencySetting_Object?&
FillTransparencySetting_Object?&
ContentTransparencySetting_Object?&
ObjectStyleObjectEffectsCategorySettings_Object?&
ObjectStyleStrokeEffectsCategorySettings_Object?&
ObjectStyleFillEffectsCategorySettings_Object?&
ObjectStyleContentEffectsCategorySettings_Object?
)
}

```

Most of the attributes and elements of an object style are shared with other elements. For more on the <TextFramePreference>, <BaselineFrameGridOption>, <TextWrapPreference>, <FrameFittingOption>, <TransparencySetting>, <StrokeTransparencySetting>, and <ContentTransparencySetting> elements, refer to the corresponding sections in the “Spreads and Master Spreads” section. For more on the <AnchoredObjectSetting> and <StoryPreference> elements, see the corresponding sections in the “Stories” section.

The <ObjectStyleObjectEffectsCategorySettings>, <ObjectStyleStrokeEffectsCategorySettings>, <ObjectStyleFillEffectsCategorySettings>, and <ObjectStyleContentEffectsCategorySettings> elements are unique to an <ObjectStyle> element. These elements contain attributes that enable (when set to true) or disable (when false) sections of the <ObjectStyle>. To specify that the drop shadow settings of an object style does not affect the fill of a page item, for example you would set the value of the `EnableDropShadow` attribute of the <ObjectStyleStrokeEffectsCategorySettings> element to false. When you do this, applying the object style will have no effect on the fill of the page item.

#### **Schema Example 122. ObjectStyleObjectEffectsCategorySettings**

```

ObjectStyleObjectEffectsCategorySettings_Object = element ObjectStyleObject-
EffectsCategorySettings {
    attribute Self { xsd:string },
    attribute EnableTransparency { xsd:boolean }?,
    attribute EnableDropShadow { xsd:boolean }?,
    attribute EnableFeather { xsd:boolean }?,
    attribute EnableInnerShadow { xsd:boolean }?,
    attribute EnableOuterGlow { xsd:boolean }?,
    attribute EnableInnerGlow { xsd:boolean }?,
    attribute EnableBevelEmboss { xsd:boolean }?,
    attribute EnableSatin { xsd:boolean }?,
    attribute EnableDirectionalFeather { xsd:boolean }?,
    attribute EnableGradientFeather { xsd:boolean }?
}

```

#### **Schema Example 123. ObjectStyleStrokeEffectsCategorySettings**

```

ObjectStyleStrokeEffectsCategorySettings_Object = element ObjectStyleStroke-
EffectsCategorySettings {
    attribute Self { xsd:string },
    attribute EnableTransparency { xsd:boolean }?,
    attribute EnableDropShadow { xsd:boolean }?,

```

```

        attribute EnableFeather { xsd:boolean }?,
        attribute EnableInnerShadow { xsd:boolean }?,
        attribute EnableOuterGlow { xsd:boolean }?,
        attribute EnableInnerGlow { xsd:boolean }?,
        attribute EnableBevelEmboss { xsd:boolean }?,
        attribute EnableSatin { xsd:boolean }?,
        attribute EnableDirectionalFeather { xsd:boolean }?,
        attribute EnableGradientFeather { xsd:boolean }?
    }
}

```

**Schema Example 124. ObjectStyleFillEffectsCategorySettings**

```

ObjectStyleFillEffectsCategorySettings_Object = element ObjectStyleFillEffects-
CategorySettings {
    attribute Self { xsd:string },
    attribute EnableTransparency { xsd:boolean }?,
    attribute EnableDropShadow { xsd:boolean }?,
    attribute EnableFeather { xsd:boolean }?,
    attribute EnableInnerShadow { xsd:boolean }?,
    attribute EnableOuterGlow { xsd:boolean }?,
    attribute EnableInnerGlow { xsd:boolean }?,
    attribute EnableBevelEmboss { xsd:boolean }?,
    attribute EnableSatin { xsd:boolean }?,
    attribute EnableDirectionalFeather { xsd:boolean }?,
    attribute EnableGradientFeather { xsd:boolean }?
}

```

**Schema Example 125. ObjectStyleContentEffectsCategorySettings**

```

ObjectStyleContentEffectsCategorySettings_Object = element ObjectStyleContent-
EffectsCategorySettings {
    attribute Self { xsd:string },
    attribute EnableTransparency { xsd:boolean }?,
    attribute EnableDropShadow { xsd:boolean }?,
    attribute EnableFeather { xsd:boolean }?,
    attribute EnableInnerShadow { xsd:boolean }?,
    attribute EnableOuterGlow { xsd:boolean }?,
    attribute EnableInnerGlow { xsd:boolean }?,
    attribute EnableBevelEmboss { xsd:boolean }?,
    attribute EnableSatin { xsd:boolean }?,
    attribute EnableDirectionalFeather { xsd:boolean }?,
    attribute EnableGradientFeather { xsd:boolean }?
}

```

Like character styles, object styles do not need to fully specify the formatting of an object. An object style might apply only a drop shadow, while leaving the fill color of the page item unchanged. The following example `<ObjectStyle>` element demonstrates this point—the only difference between this object style and the object style it is based on is that the `Mode` attribute of the `<TransparencySetting>` element is set to `Drop` (thereby turning the drop shadow on). For more information on the available attributes of the `<TransparencySetting>` element.

**IDML Example 82. ObjectStyle**

```

<ObjectStyle Self="ObjectStyle\DropShadow" Name="DropShadow">
    <Properties>
        <BasedOn type="object">ObjectStyle\[Normal Graphics Frame]</BasedOn>
    </Properties>

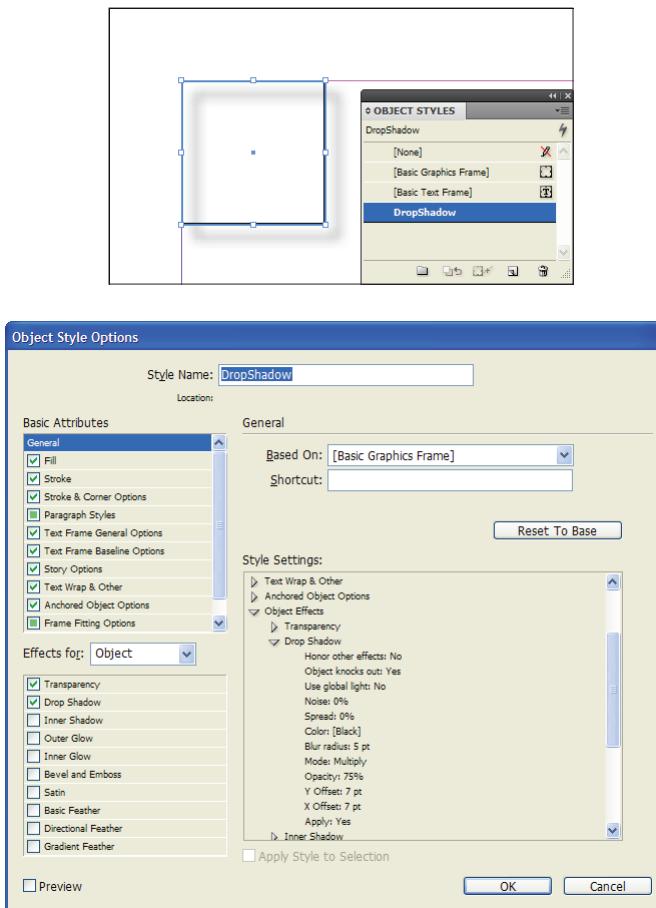
```

```

<TransparencySetting Self="ud4TransparencySetting1">
    <DropShadowSetting Self="ud4TransparencySetting1DropShadowSetting1"
        Mode="Drop"/>
</TransparencySetting>
</ObjectStyle>

```

Figure 57. ObjectStyle



#### 6.4.8 TOCStyles

A table of contents (TOC) can list the contents of an InDesign document. A document can contain multiple tables of contents—for example, a list of chapters and a list of illustrations. A “TOC style” defines the way that a specific TOC is created and formatted.

In an IDML document, `<TOCStyle>` elements contain attributes that define the properties of a TOC. Inside a `<TOCStyle>` element, `<TOCStyleElement>` elements define which paragraphs will appear in the TOC and the formatting of each paragraph.

##### Schema Example 126. TOCStyle

```

TOCStyle_Object = element TOCStyle {
    attribute Self { xsd:string },
    attribute TitleStyle { xsd:string }?,
    attribute Title { xsd:string }?,
    attribute Name { xsd:string },
    attribute RunIn { xsd:boolean }?,
}

```

```

        attribute IncludeHidden { xsd:boolean }?,
        attribute IncludeBookDocuments { xsd:boolean }?,
        attribute CreateBookmarks { xsd:boolean }?,
        attribute SetStoryDirection { HorizontalOrVertical_EnumValue }?,
        attribute NumberedParagraphs { NumberedParagraphsOptions_EnumValue }?,
        element Properties {
            element Label { element KeyValuePair { KeyValuePair_TypeDef }*
                }?
            ?
            ,
            (
                TOCStyleEntry_Object*
            )
        }
    
```

**Schema Example 127. TOCStyleEntry**

```

TOCStyleEntry_Object = element TOCStyleEntry {
    attribute Self { xsd:string },
    attribute Name { xsd:string }?,
    attribute Level { xsd:short }?,
    attribute PageNumberPosition { PageNumberPosition_EnumValue }?,
    attribute Separator { xsd:string }?,
    attribute SortAlphabet { xsd:boolean }?,
    element Properties {
        element FormatStyle {
            (object_type, xsd:string) |
            (string_type, xsd:string)
        }?&
        element PageNumberStyle {
            (object_type, xsd:string) |
            (string_type, xsd:string)
        }?&
        element SeparatorStyle {
            (object_type, xsd:string) |
            (string_type, xsd:string)
        }?
    }
    ?
}

```

**Table 121. TOCStyle Properties Represented as Attributes**

Name	Type	Req	Description
CreateBookmarks	boolean	no	If true, creates bookmarks for TOC entries.
IncludeBook- Documents	boolean	no	If true, includes the entire book in the TOC. If false, includes only the TOC entries in the current document. Note: Valid when the current document is part of a book.

Name	Type	Req	Description
IncludeHidden	boolean	no	If true, the TOC includes entries from text on hidden layers (this applies to the next TOC generated by a user, not to a TOC that has already been placed in a document).
Name	string	no	The name of the TOCStyle.
Numbered-Paragraphs	Numbered-Paragraphs-Options_EnumValue	no	The format for importing numbered paragraphs into the TOC. Can be <code>IncludeFullParagraph</code> , <code>IncludeNumbersOnly</code> , or <code>ExcludeNumbers</code> .
RunIn	boolean	no	If true, the lowest-level TOC entries are placed on the same line as the previous entry.
SetStoryDirection	HorizontalOrVertical_EnumValue	no	The table of contents story direction. Can be <code>LeftToRightDirection</code> or <code>RightToLeftDirection</code> .
Title	string	no	The TOC title.
TitleStyle	string	no	The paragraph style applied to the TOC title.

**Table 122. TOCStyleEntry Properties Represented as Attributes**

Name	Type	Req	Description
Level	short	no	The indent level of the entry in the TOC.
Name	string	no	The name of the TOCStyleEntry.
PageNumber-Position	PageNumber-Position_EnumValue	no	The page number placement for the TOC entry style. Can be <code>AfterEntry</code> , <code>BeforeEntry</code> , or <code>None</code> .
Self	string	yes	The unique ID of the object.
Separator	string	no	The string to insert between the entry text and the page numbers.
SortAlphabet	boolean	no	If true, sorts the TOC entries alphabetically.

**Table 123. TOCStyleEntry Properties Represented as Elements**

Name	Type	Req	Description
FormatStyle	string	no	The paragraph style applied to the TOC entry. Can be either: <code>type = "object"</code> A reference to a paragraph style as a unique ID (the value of the <code>Self</code> attribute of the <code>&lt;ParagraphStyle&gt;</code> element). or <code>type = "string"</code> A reference to the default “[No paragraph style]” element, in the form: <code>\$ID/[No paragraph style]</code> . This is the default.

PageNumberStyle	string	no	The character style applied to the page number of the entry. Can be either: type = "object" A reference to a character style as a unique ID (the value of the Self attribute of the <CharacterStyle> element). or type = "string" A reference to the default "[No character style]" element, in the form: \$ID/[No character style]. This is the default.
SeparatorStyle	string	no	The character style applied to the separator of the entry. Can be either: type = "object" A reference to a character style as a unique ID (the value of the Self attribute of the <CharacterStyle> element). or type = "string" A reference to the default "[No character style]" element, in the form: \$ID/[No character style]. This is the default.

**IDML Example 83. TOCStyle**

```

<TOCstyle Self="TOCStyle\kDefaultTOCStyleName" TitleStyle="ParagraphStyle\k[No
paragraph style]" Title="Contents" Name="$ID/DefaultTOCStyleName" RunIn="false"
IncludeHidden="false" IncludeBookDocuments="false" CreateBookmarks="true" Set-
StoryDirection="Horizontal" NumberedParagraphs="IncludeFullParagraph"/>
<TOCstyle Self="TOCStyle\cExampleTOCStyle" TitleStyle="ParagraphStyle\cContents-
Title" Title="Contents" Name="ExampleTOCStyle" RunIn="false" IncludeHidden="false"
IncludeBookDocuments="false" CreateBookmarks="true" NumberedParagraphs="Include-
FullParagraph">
  <TOCStyleEntry Self="ue4TOCStyleEntry0" Name="Headings:Heading1" Level="1"
  PageNumberPosition="AfterEntry" Separator="$ID/^t" SortAlphabet="false">
    <Properties>
      <FormatStyle type="object">ParagraphStyle\TOCHeading1</FormatStyle>
      <PageNumberStyle type="object">CharacterStyle\PageNumber</PageNumberStyle>
      <SeparatorStyle type="object">CharacterStyle\DotLeader</SeparatorStyle>
    </Properties>
  </TOCStyleEntry>
  <TOCStyleEntry Self="ue4TOCStyleEntry1" Name="Headings:Heading2" Level="2"
  PageNumberPosition="AfterEntry" Separator="$ID/^t" SortAlphabet="false">
    <Properties>
      <FormatStyle type="object">ParagraphStyle\TOCHeading2</FormatStyle>
      <PageNumberStyle type="object">CharacterStyle\PageNumber</PageNumberStyle>
      <SeparatorStyle type="object">CharacterStyle\DotLeader</SeparatorStyle>
    </Properties>
  </TOCStyleEntry>
  <TOCStyleEntry Self="ue4TOCStyleEntry2" Name="Headings:Heading3" Level="3"
  PageNumberPosition="AfterEntry" Separator="$ID/^t" SortAlphabet="false">
    <Properties>
      <FormatStyle type="object">ParagraphStyle\TOCHeading3</FormatStyle>
      <PageNumberStyle type="object">CharacterStyle\PageNumber</PageNumberStyle>
    </Properties>
  </TOCStyleEntry>
</TOCstyle>

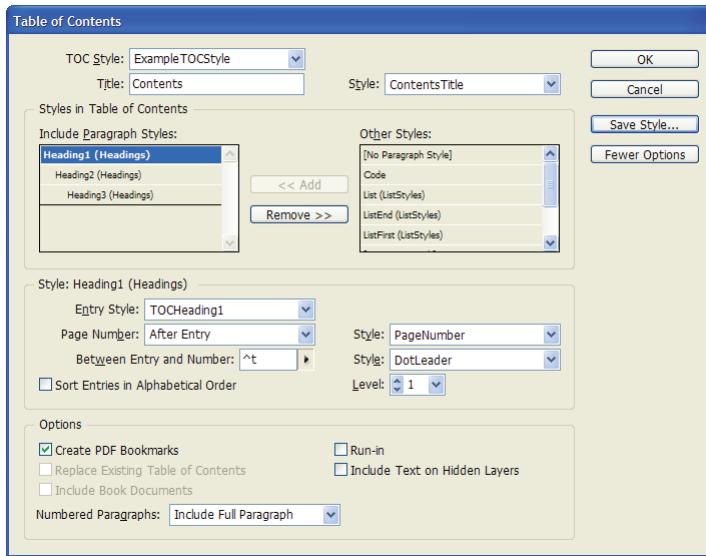
```

```

<SeparatorStyle type="object">CharacterStyle\DotLeader</SeparatorStyle>
</Properties>
</TOCStyleEntry>
</TOCStyle>

```

Figure 58. TOCStyle



#### 6.4.9 TrapPresets

A trap preset is a collection of trapping settings you can apply to a page or range of pages in an InDesign document. Trap presets can be applied to the entire document, to ranges of pages, or to individual pages. If no trap preset is applied, InDesign will use the [Default] trap preset to trap a document.

Trap presets in an IDML package are stored as <TrapPreset> elements.

##### Schema Example 128. TrapPreset

```

TrapPreset_Object = element TrapPreset {
    attribute Self { xsd:string },
    attribute Name { xsd:string },
    attribute DefaultTrapWidth { xsd:double {minInclusive="0" maxInclusive="8"} }?,
    attribute BlackWidth { xsd:double {minInclusive="0" maxInclusive="8"} }?,
    attribute TrapJoin { EndJoin_EnumValue }?,
    attribute TrapEnd { TrapEndTypes_EnumValue }?,
    attribute ObjectsToImages { xsd:boolean }?,
    attribute ImagesToImages { xsd:boolean }?,
    attribute InternalImages { xsd:boolean }?,
    attribute OneBitImages { xsd:boolean }?,
    attribute ImagePlacement { TrapImagePlacementTypes_EnumValue }?,
    attribute StepThreshold { xsd:double {minInclusive="1" maxInclusive="100"} }?,
    attribute BlackColorThreshold { xsd:double {minInclusive="0" maxInclusive="100"} }?,
    attribute BlackDensity { xsd:double {minInclusive="0" maxInclusive="10"} }?,
    attribute SlidingTrapThreshold { xsd:double {minInclusive="0" max-
Inclusive="100"} }?,
}

```

```

        attribute ColorReduction { xsd:double {minInclusive="0" maxInclusive="100"} }?,
        element Properties {
            element Label { element KeyValuePair { KeyValuePair_TypeDef }* }
            ?
        }
        ?
    }
}

```

**IDML Example 84. TrapPreset**

```
<TrapPreset Self="TrapPreset\ExampleTrapPreset" Name="ExampleTrapPreset" Default-
TrapWidth="0.25" BlackWidth="0.5" TrapJoin="MiterEndJoin" TrapEnd="MiterTrap-
Ends" ObjectsToImages="true" ImagesToImages="true" InternalImages="false"
OneBitImages="true" ImagePlacement="CenterEdges" StepThreshold="10" Black-
ColorThreshold="100" BlackDensity="1.6" SlidingTrapThreshold="70" Color-
Reduction="100"/>
```

**Table 124. TrapPreset Properties Represented as Attributes**

Name	Type	Req	Description
BlackColor- Threshold	double	no	The minimum amount (as a percentage) of black ink required before the black width setting is applied. (Range: 0 to 100)
BlackDensity	double	no	The neutral density value at or above which an ink is considered black. (Range: .001 to 10)
BlackWidth	double	no	The black width. (Range: 0.0 to 8.0)
ColorReduction	double	no	The degree (as a percentage) to which components from abutting colors are used to reduce the trap color. (Range: 0 to 100) Note: 0% creates a trap whose neutral density is equal to the neutral density of the darker color.
DefaultTrapWidth	double	no	The default width for trapping all colors except those involving solid black. (Range: 0.0 to 8.0)
ImagePlacement	TrapImage- PlacementTypes_- EnumValue	no	The trap placement between vector objects and bitmap images. Can be CenterEdges, Choke, ImageNeutralDensity, or ImagesOver- Spread.
ImagesToImages	boolean	no	If true, turns on trapping along the boundary of overlapping or abutting bitmap images.
InternalImages	boolean	no	If true, turns on trapping among colors within individual bitmap images.
Name	string	no	The name of the trap preset.
ObjectsToImages	boolean	no	If true, ensures that vector objects overlap bitmap images.
OneBitImages	boolean	no	If true, ensures that one-bit images trap to abutting objects.
SlidingTrap- Threshold	double	no	The difference (as a percentage) between the neutral densities of abutting colors at which the trap is moved from the darker side of a color edge toward the centerline. (Range: 0 to 100)

Name	Type	Req	Description
StepThreshold	double	no	The amount (as a percentage) that components of abutting colors must vary before a trap is created. (Range: 1 to 100)
TrapEnd	TrapEndTypes_EnumValue	no	The shape to use at the intersection of three-way traps. Can be MiterTrapEnds or OverlapTrapEnds.
TrapJoin	EndJoin_EnumValue	no	The join type of the trap preset. Can be MiterEndJoin, RoundEndJoin, or BevelEndJoin.

## 6.5 XML Elements

This section discusses the way that XML elements in the XML structure of an InDesign document are represented in an IDML document.

### 6.5.1 BackingStory.xml

The `BackingStory.xml` file in an IDML package contains any XML content in the XML structure of the InDesign document that has not been placed in the layout.

#### Schema Example 129. BackingStory

```
BackingStory_File = element idPkg:BackingStory {
    XmlStory_Object*
}
```

The following example shows the `<idPkg:BackingStory>` element for an IDML package in which all XML content has been placed in the layout (see the `MappingStyleToTags` example in the “Stories” section). The XML structure contains one child of the root `<XMLElement>` element, which has been associated with the `<Story>` element in the IDML package with the `Self` attribute `ud8`.

#### IDML Example 85. BackingStory

```
<idPkg:BackingStory xmlns:idPkg="http://ns.adobe.com/AdobeInDesign/idml/1.0/
packaging">
<XmlStory Self="u9c">
    <ParagraphStyleRange AppliedParagraphStyle=
        "ParagraphStyle\kNormalParagraphStyle">
        <CharacterStyleRange AppliedCharacterStyle=
            "CharacterStyle\k[No character style]">
            <Content></Content>
            <XMLElement Self="di2" MarkupTag="XMLTag\cRoot">
                <XMLElement Self="di2i3" MarkupTag="XMLTag\cStory" XMLContent="ud8"/>
            </XMLElement>
        </CharacterStyleRange>
    </ParagraphStyleRange>
</XmlStory>
</idPkg:BackingStory>
```

### 6.5.2 XMLStory

An `<XMLStory>` element represents XML text content that has not yet been placed in a layout. The `<XMLStory>` element is identical to the `<Story>` element; refer to “Stories.” An `<XMLStory>` element can contain `<ParagraphStyleRange>` elements, `<CharacterStyleRange>` elements, `<Table>` elements, anchored frames, and all of the other elements that can appear in a `<Story>` element.

### 6.5.3 XMLElement

The <XMLElement> elements in an IDML file represent XML content in the structure of an InDesign document. For an example of an <XMLElement> element in text, see “Stories.” For an example of an <XMLElement> element associated with a page item, see “Spreads and Master Spreads.”

#### Schema Example 130. XMLElement

```
XMLElement_Object = element XMLElement {
    attribute Self { xsd:string },
    attribute MarkupTag { xsd:string }?,
    attribute XMLContent { xsd:string }?,
    attribute NoTextMarker { XMLNoTextMarker_EnumValue }?,
(
    XMLAttribute_Object*&
    DTD_Object*&
    XMLElement_Object*&
    XMLComment_Object*&
    XMLInstruction_Object*&
    TextFrame_Object*&
    Oval_Object*&
    Rectangle_Object*&
    GraphicLine_Object*&
    Polygon_Object*&
    Group_Object*&
    EPSText_Object*&
    FormField_Object*&
    Button_Object*&
    Table_Object*&
    Cell_Object*&
    Footnote_Object*&
    Note_Object*&
    Link_Object*
)
}
```

---

**Table 125. XMLElement Properties Represented as Attributes**

Name	Type	Req	Description
MarkupTag	string	no	A reference to the <XMLTag> of the XML element (as a reference to the <code>Self</code> attribute of the <XMLTag>).
XMLContent	string	no	A reference to the <Story> of the XML element (as a reference to the <code>Self</code> attribute of the <Story>). Only used by the top-level XML element in a Story.
NoTextMarker	XMLNoTextMarker_EnumValue	no	Used for <Table> and <Cell> elements (these elements do not have associated marker characters in the layout).

**Schema Example 131. XMLAttribute**

```
XMLAttribute_Object = element XMLAttribute {
    attribute Self { xsd:string },
    attribute Name { xsd:string },
    attribute Value { xsd:string }
}
```

---

**Table 126. XMLAttribute Properties Represented as Attributes**

Name	Type	Req	Description
Name	string	yes	The name of the attribute.
Value	string	yes	The value of the attribute.

**Schema Example 132. XMLInstruction**

```
XMLInstruction_Object = element XMLInstruction {
    attribute Self { xsd:string },
    attribute StoryOffset { xsd:string }?,
    attribute Target { xsd:string },
    attribute Data { xsd:string }?
}
```

---

**Table 127. XMLInstruction Properties Represented as Attributes**

Name	Type	Req	Description
StoryOffset	string	no	The location of the <XMLInstruction>, as a reference to the Self attribute of a <CharacterStyleRange> element.
Target	string	yes	The target of the <XMLInstruction>.
Data	string	no	The contents of the <XMLInstruction>.

**Schema Example 133. XMLComment**

```
XMLComment_Object = element XMLComment {
    attribute Self { xsd:string },
    attribute StoryOffset { xsd:string }?,
    attribute Value { xsd:string }?
}
```

---

**Table 128. XMLComment Properties Represented as Attributes**

Name	Type	Req	Description
StoryOffset	string	no	The location of the <XMLInstruction>, as a reference to the Self attribute of a <CharacterStyleRange> element.
Value	string	no	The contents of the <XMLComment>.

**Schema Example 134. DTD**

```

DTD_Object = element DTD {
    attribute Self { xsd:string },
    attribute StoryOffset { xsd:string }?,
    element Properties {
        element Contents {
            (string_type, xsd:string) |
            (enum_type, SpecialCharacters_EnumValue) |
            (object_type, xsd:string)
        }?
    }?
}
?
```

**Table 129. DTD Properties Represented as Attributes**

Name	Type	Req	Description
StoryOffset	string	no	The location of the <XMLInstruction>, as a reference to the Self attribute of a <CharacterStyleRange> element.

**Table 130. DTD Properties Represented as Elements**

Name	Type	Req	Description
Contents	string or SpecialCharacters enumeration	no	The contents of the DTD.

## 6.6 Mapping

The <XMLExportMap> and <XMLImportMap> elements define the tag to style and style to tag mappings in an IDML document. For more on mapping XML tags to paragraph and character styles, refer to the InDesign onilne help.

**Schema Example 135. XMLExportMap**

```

XMLExportMap_Object = element XMLExportMap {
    attribute Self { xsd:string },
    attribute MarkupTag { xsd:string },
    attribute MappedStyle { xsd:string },
    attribute IncludeMasterPageStories { xsd:boolean }?,
    attribute IncludePasteboardStories { xsd:boolean }?,
    attribute IncludeEmptyStories { xsd:boolean }?
}
```

**Table 131. XMLExportMap Properties Represented as Attributes**

Name	Type	Req	Description
MarkupTag	string	yes	A reference to an <XMLTag> element in the IDML package (as a reference to its Self attribute).
MappedStyle	string	yes	A reference to a <ParagraphStyle> or <CharacterStyle> element in the IDML package (as a reference to its Self attribute).
IncludeMasterPageStories	boolean	no	If true, map stories that appear only on master spreads.
IncludePasteboardStories	boolean	no	If true, map stories that appear only on the pasteboard.
IncludeEmptyStories	boolean	no	If true, map the stories that have no content.

**IDML Example 86. XMLExportMap**

```
<XMLExportMap Self="dicd" MarkupTag="XMLTag\cheading_1"
  MappedStyle="ParagraphStyle\cheading 1" IncludeMasterPageStories="false"
  IncludePasteboardStories="false" IncludeEmptyStories="false"/>
```

**Schema Example 136. XMLImportMap**

```
XMLImportMap_Object = element XMLImportMap {
    attribute Self { xsd:string },
    attribute MarkupTag { xsd:string },
    attribute MappedStyle { xsd:string }
}
```

**Table 132. XMLImportMap Properties Represented as Attributes**

Name	Type	Req	Description
MarkupTag	string	yes	A reference to an <XMLTag> element in the IDML package (as a reference to its Self attribute).
MappedStyle	string	yes	A reference to a <ParagraphStyle> or <CharacterStyle> element in the IDML package (as a reference to its Self attribute).

**IDML Example 87. XMLImportMap**

```
<XMLImportMap Self="dicd" MarkupTag="XMLTag\cheading_1"
  MappedStyle="ParagraphStyle\cheading 1"/>
```

## 6.7 Tags

The `<XMLTag>` element represents an XML tag in an InDesign document. In an IDML package, tags are define in the `Tags.xml` file inside the XML folder. For more on creating and applying XML tags, refer to the InDesign online help.

### Schema Example 137. XMLTag

```
XMLTag_Object = element XMLTag {
    attribute Self { xsd:string },
    attribute Name { xsd:string },
    element Properties {
        element TagColor { InDesignUIColorType_TypeDef }?&
        element Label { element KeyValuePair { KeyValuePair_TypeDef }* }?
    }?
}
```

**Table 133. XMLTag Properties Represented as Attributes**

Name	Type	Req	Description
Name	string	yes	The name of the XML tag.

**Table 134. XMLTag Properties Represented as Elements**

Name	Type	Req	Description
TagColor	InDesignUIColor-Type	no	The color of the XML tag. TagColor can be a UIColor enumeration or an RGB color as a list of three <code>&lt;ListItem&gt;</code> elements (in the order R, G, B).

### IDML Example 88. XMLTag

```
<XMLTag Self="XMLTag\cbody_text" Name="body_text">
<Properties>
    <TagColor type="enumeration">Yellow</TagColor>
</Properties>
</XMLTag>
```

# Appendix A. UCF Container Format

## 1.1 Overview

### 1.1.1 Purpose and Scope

This appendix defines the Universal Container Format. UCF is a general-purpose container technology. It is based on the packaging principles of OCF, the OEBSP Container Format, created by the International Digital Publishing Forum. The OCF specification describes a general-purpose container technology in the context of encapsulating OEBPS publications. While the OCF specification anticipates that the general-purpose container technology it describes will ultimately be used in other bundling applications, the specification itself does not formally separate the generic technology from its use in the context of OEBPS. This goal of this appendix is to do just that, specifying a generic container format that can be used by many applications, where OCF itself is one application.

As a general container format, UCF collects a related set of files into a single-file container. UCF can be used to collect files in various document and data formats and for classes of applications. The single-file container enables easy transport of, management of, and random access to, the collection.

UCF defines rules for how to represent an abstract collection of files (the “abstract container”) into physical representation within a Zip archive (the “physical container”). The rules for Zip containers build upon and are backward compatible with the Zip technology used by Open Document Format (ODF) 1.0.

UCF is the RECOMMENDED single-file container technology for all Zip-based Adobe formats. It is designed to provide a set of lightweight constraints on the use of Zip. It includes a set of optional features including digital signatures and encryption. If an UCF-based file format includes this optional functionality, it should follow the UCF specifications for use of these features. For example, not all UCF-based formats will make use of digital signatures. However, if a format does include support for signatures, it should follow the UCF rules for signatures.

This appendix borrows heavily from both the UCF specification and the OCF specification. Where possible, the same language is used with the permission of the IDPF. Adobe and IDPF plan to work with OASIS to develop an open standard container format based on OCF (and most likely called Open Container Format) that will be the basis for future versions of OCF and the Open Document Format.

### 1.1.2 Definitions

#### **ASCII**

American Standard Code for Information Interchange – a 7-bit character encoding based on the English alphabet (ANSI X3.4-1986). When used in this document, ASCII refers to the printable

graphic characters in the range 33 (decimal) through 126 (decimal) and the nonprintable space character 32 (decimal).

### **IDML**

An XML representation of an InDesign document.

### **IRI**

Internationalized Resource Identifier (<http://www.ietf.org/rfc/rfc3987.txt>).

### **UCF**

The Universal Container Format defined by this specification.

### **UCF Container**

A container file that is compliant with the format defined in this specification.

### **UCF User Agent**

A combination of hardware and/or software that accepts documents or data packaged in an UCF Container and makes them available to consumer of the content.

### **ODF**

Open Document Format (<http://www.oasis-open.org/committees/download.php/12572/OpenDocument-v1.0-os.pdf>).

### **OEBPS**

Open eBook Publication Structure (<http://www.idpf.org/oebps/oebps1.2/index.htm>).

### **MIME**

Multipurpose Internet Mail Extensions (<http://www.ietf.org/rfc/rfc2045.txt>). “MIME media types” provide a standard methodology for specifying the content type of objects.

### **RFC**

Literally “Request For Comments,” but more generally a document published by the Internet Engineering Task Force (IETF). See <http://www.ietf.org/rfc.html>.

### ***Relax NG***

A schema language for XML (<http://www.relaxng.org/>).

### ***Rootfile***

The top-level file of a rendition of a publication; either the “root” from which all other components can be found or the lone file encapsulating the rendition. The OEBPS rootfile is the OEBPS Package file. A PDF file containing the PDF rendition could also be a rootfile.

### ***XML***

Extensible Markup Language (<http://www.w3.org/TR/xml/>).

### ***Zip***

A de facto industry standard bundling and compression format (<http://www.pkware.com/support/zip-application-note>).

#### **1.1.3 Relationship of UCF to Other Specifications**

UCF combines subsets and applications of other specifications. Together, these facilitate the construction, organization, presentation, and unambiguous interchange of electronic documents:

The OEBPS Container Format specification (<http://www.idpf.org/ocf/ocf1.0/>).

Zip format (<http://www.pkware.com/support/zip-application-note>)

The Extensible Markup Language (XML) 1.0 (Fourth Edition) specification (<http://www.w3.org/TR/xml/>)

The Namespaces in XML 1.0 (Second Edition) specification (<http://www.w3.org/TR/xml-names/>)

XML-Signature Syntax and Processing (<http://www.w3.org/TR/2002/REC-xmldsig-core-20020212>)

XML Encryption Syntax and Processing (<http://www.w3.org/TR/2002/REC-xmlenc-core-20021210>)

Extensible Metadata Platform (XMP) (<http://www.adobe.com/products/xmp/>)

The Unicode Consortium. The Unicode Standard, Version 5.0.0, defined by: The Unicode Standard, Version 5.0 (Boston, MA, Addison-Wesley, 2007. ISBN 0-321-48091-0), as updated from time to time by the publication of new versions. (See <http://www.unicode.org/unicode/standard/versions> for the latest version and additional information on versions of the standard and of the Unicode Character Database).

Particular MIME media types (<http://www.ietf.org/rfc/rfc4288.txt> and <http://www.iana.org/assignments/media-types/index.html>)

Open Document Format for Office Applications (Open Document) v1.0 (<http://www.oasis-open.org/committees/download.php/12572/OpenDocument-v1.0-os.pdf>)

## 1.2 Conformance

The keywords “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document MUST be interpreted as described in (<http://www.ietf.org/rfc/rfc2119.txt>).

This section defines conformance requirements for UCF.

### 1.2.1 Conforming Containers

The term “Conforming UCF Abstract Container” indicates an UCF Abstract Container (See Section 2.2) that conforms to all the relevant conformance criteria defined in this specification. The term “Conforming UCF Zip Container” indicates a Zip archive that conforms to the relevant Zip container conformance criteria (See Section 4) and which implements an instance of a Conforming UCF Abstract Container.

In addition to other conformance criteria defined in this specification, a Conforming UCF Abstract Container MUST meet the following conditions:

- All XML files defined by this specification MUST be well-formed (as defined in XML 1.0).
- All XML defined by this specification files MUST be compatible with the XML 1.0 specification (<http://www.w3.org/TR/2006/REC-xml-20060816>) and the Namespaces in XML specification (<http://www.w3.org/TR/REC-xml-names>).
- These conditions do not apply to files in the container that are not defined by this specification (files other than container.xml, manifest.xml, metadata.xml, signatures.xml, encryption.xml, and rights.xml).

### 1.2.2 Conforming User Agents

The term “Conforming UCF User Agent” indicates an UCF User Agent that supports all of the mandatory features defined by this specification.

An UCF User Agent that does not support all of the features defined in this specification MUST NOT claim to be a Conforming UCF User Agent and SHOULD provide readily available documentation of the subset of features it supports.

### 1.2.3 Future Directions

It is the intent of the contributors to this specification that subsequent versions of this specification continue in the directions established by the 1.0 release. Specifically:

- Future versions of this specification are expected to improve alignment with OASIS/ODF and IDPF/OCF.
- Any required functionality not present in relevant official standards shall be defined in a manner consistent with its eventual submission to an appropriate standards body as extensions to existing standards.

## 1.3 UCF Overview

### 1.3.1 UCF: A General Container Technology

UCF is designed as a general container technology. In particular, UCF is designed to be upwardly compatible with the container technology used in ODF 1.0 such that a future version of ODF might use UCF.

### 1.3.2 “Abstract Container” vs. “Physical Container”

An “Abstract Container” defines a file system model for the contents of the container. The file system model MUST have a single common root directory for all of the contents of the container. The special files REQUIRED by UCF MUST be included within the META-INF directory that is a direct child of the root directory.

A “Physical Container” holds the physical manifestation of an abstract container. UCF defines how an abstract container MUST be mapped to the following two physical container technologies:

- *File System Container* – The mapping of an Abstract Container to a file system within computer storage media on a specific platform (e.g., a hard disk on a computer or a data CD) MUST be a one-to-one mapping where each directory and file within the abstract container is represented as a directory or file within the file system. Section 3.3 defines a set of restrictions on file system names intended to allow files to be easily stored in most modern file systems.
- *Zip Container* - The mapping of an Abstract Container to a Zip archive is defined in Section 4.

If a user agent processed both types of physical container, the contents of an OCF container MUST be processed the same no matter whether using a File System Container or a Zip Container. In both cases, the UCF User Agent ultimately opens the rootfile, from which it can determine how to process the container.

## 1.4 UCF Container Contents

### 1.4.1 File and directory structure

The virtual file system for the UCF “Abstract Container” MUST have a single common root directory for all of the contents of the container.

The following file names in the root directory are reserved:

- “mimetype”
- “META-INF”

The “mimetype” file is discussed in Section 4. The META-INF/ directory contains the reserved files used by UCF. These reserved files are described in the following sections. All other files within the Abstract Container MAY be in any location descendant from the root directory except for “mimetype” at the root level or directly within the META-INF directory. An UCF based-format may include files in the META-INF directory as long as these files are within subdirectories in META-INF. The names of these subdirectories MUST NOT include the “.” character. This avoids conflicts with files specified by future versions of the UCF specification. Any file in the META-INF directory used by UCF will include a “.” character.

It is RECOMMENDED that the contents of individual documents or applications be stored within dedicated sub-directories to minimize potential file name collisions in the event that multiple renditions are used or that multiple publications per container are supported in future versions UCF.

#### **1.4.2 Relative IRIs for referring to other components**

Files within the Abstract Container refer to each other via Relative IRI References (<http://www.ietf.org/rfc/rfc3987.txt> and <http://www.ietf.org/rfc/rfc3986.txt>), no matter what is used for the physical container (e.g., File System Container or Zip Container). For example, if a file named “chapter1.html” refers to an image file named “image1.jpg” that is located in the same directory, then “chapter1.html” might contain the following as part of its content:

```

```

For Relative IRI References, the Base IRI (see RFC3986) is determined by the relevant language specifications for the given file formats. For example, the CSS specification defines how relative IRI references work in the context of CSS style sheets and property declarations.

Unlike many language specifications, the Base IRIs for all files within the META-INF/ directory use the root folder for the Abstract Container as the default Base IRI. For example, if META-INF/container.xml has the following content:

```
<?xml version="1.0"?>
<container version="1.0" xmlns="urn:oasis:names:tc:opendocument:xmlns:container">
  <rootfiles>
    <rootfile full-path="OEBPS/Great Expectations.opf"
      media-type="application/oebps-package+xml" />
  </rootfiles>
</container>
```

the path “OEBPS/Great Expectations.opf” is relative to the root directory for the Abstract Container and not relative to the META-INF/ directory.

If a Relative IRI Reference contains an absolute path (an IRI that has no schema or authority but begins with a “/”), the reference is resolved relative to the root directory of the Abstract Container. For example, in the example in Section 2.3.1, the IRI “/OEBPS/cover.html” will refer to the file OEBPS/cover.html no matter what file the IRI is found in.

### 1.4.3 File Names

The term File Name represents the name of any type of file, either a directory or an ordinary file within a directory within an Abstract Container. For a given directory within the Abstract Container, the Path Name is a string holding all directory names in the full path concatenated together with a “/” character separating the directory names. For a given file within the Abstract Container, the Path Name is the string holding all directory names concatenated together with a “/” character separating the directory names, followed by a “/” character and then the name of the file. The File Name restrictions described below are designed to allow directory names and file names to be used without modification on most commonly used operating systems. The UCF specification does not specify how a UCF User Agent that is unable to represent UCF conforming File Names would compensate for this incompatibility.

The following statements apply to Conforming UCF Content:

- File Names MUST be UTF-8 encoded with the restrictions below
- When represented as UTF-8, File Names MUST NOT exceed 255 bytes
- When represented as UTF-8, the Path Name for any directory or file within the Abstract Container MUST NOT exceed 65535 bytes
- File Names MUST NOT use the following characters (These characters are not be supported always across commonly used operating systems):
  - U+0022 “ QUOTATION MARK
  - U+002A \* ASTERISK
  - U+002E . FULL STOP, as the last character
  - U+002F / SOLIDUS
  - U+003A : COLON
  - U+003C < LESS-THAN SIGN
  - U+003E > GREATER-THAN SIGN
  - U+003F ? QUESTION MARK
  - U+005C \ REVERSE SOLIDUS
  - The C0 controls, U+0000 through U+001F and U+007F
- File Names are case sensitive.
- Two File Names within the same directory MUST NOT map to the same string following case normalization (<http://www.unicode.org/reports/tr21/tr21-5.html>). Two File Names that differ only in case are disallowed within the same directory.
- Two File Names within the same directory MUST NOT be canonically equivalent in the Unicode sense.

Note that some commercial Zip tools do not support the full Unicode range and may only support the ASCII range for File Names. Content creators who want to use Zip tools that have these restrictions MAY find it is best to restrict their File Names to the ASCII range. If the names of files can not be preserved during the unzipping process, it will be necessary to compensate for any name translation which took place when the files are referred to by URI from within the content.

#### **1.4.4 Container media type identification**

It is frequently necessary for applications to determine the media type of a file. This is usually accomplished by looking at the file extension of the file. This gives applications a quick way to determine the type of the file without looking inside the file. UCF Container files SHOULD use an extension specific to the kind of UCF Container it is.

Unfortunately, the identification of files through the use of file extensions is notoriously unreliable. As a result, it is desirable to have a more robust way of identifying files independent of their file names or extensions. One mechanism that has evolved for doing this is to require the placement of specific information at specific file offsets. A processing agent can then check a fixed location to determine if the file is a specific type of UCF Container.

The method that has evolved for doing this in Zip archives is the inclusion of an uncompressed, unencrypted file called “mimetype” as the first file in the Zip archive. The contents of this file are the media type of the file. UCF Containers MUST place the media type as an ASCII string in the “mimetype” file as the first file in the Zip archive. See Section 4 for more detail on this mechanism.

#### **1.4.5 META-INF**

All valid UCF Containers MAY include a directory called “META-INF” at the root level of the container file system. This directory contains the files specified below that describe the contents, metadata, signatures, encryption, rights and other information about the contained publication.

The semantics of the following files that MAY be present at the “META-INF/” level are specified. All other files found at the “META-INF/” level MUST be ignored by conformant UCF User Agents.

#### **1.4.6 Container – META-INF/container.xml (Optional)**

(This is normative.)

An UCF Container MAY include a file called “container.xml” within the “META-INF” directory at the root level of the container file system. If present, the container.xml file MAY identify the MIME type of, and path to, the rootfile for the container and any OPTIONAL alternate renditions included within the container. An UCF-based format MUST either require container.xml to identify the rootfile or specify a format-specific method for initiating processing of the container. The container.xml file MAY specify implicit relationships in the container, as described in Section 3.5.1.2.

The container.xml file MUST NOT be encrypted.

The container.xml file contains XML that uses the “urn:oasis:names:tc:opendocument:xmlns:container” namespace for all of its elements and attributes. The “version=”1.0”” attribute MUST be included for all containers that conform to this version of the specification.

A RELAX NG UCF schema describing the <container> element that MUST be the root element of container.xml can be found in the Appendix A.

### ***Rootfiles (Optional)***

The <rootfiles> element MUST contain at least one <rootfile> element.

Each <rootfile> element specifies the rootfile of a single rendition of the contained publication. A rootfile often includes an enumeration of the other files needed by the rendition.

The values of the full-path attributes MUST contain a “path component” (as defined by RFC3986) which MUST only take the form of a “path-rootless” (as defined by RFC3986). The path components are relative to the root of the container in which they are used.

Conforming UCF User Agents MUST ignore unrecognized elements (and their contents) and unrecognized attributes within a container.xml file, including unrecognized elements and unrecognized attributes from other namespaces.

Conforming container.xml files MUST be valid according to the RELAX NG UCF schema with the <container> element as the root element after removing all elements (and child nodes of these elements) and attributes from other namespaces.

### ***Relationships (Optional)***

Container.xml MAY include information that identifies implicit relationships between files in a container. Usually, one file explicitly refers to another file. For example, an SVG page description may reference an image. At times, it is convenient to indirectly reference a file. For example, one file may have metadata associated with it stored in a second file. Using an indirect association makes it possible to add metadata by simply creating a metadata resource and not changing the original resource or, in fact, any resource in the package.

Beyond convenience, indirect associations enable container files to be digitally signed while still permitting the addition of certain kinds of data such as metadata or annotations. For example, a document with no annotations can be signed. Annotations can then be added without invalidating the signature.

UCF provides a generic mechanism for establishing a relationship between two container files. A relationship specifies a relationship type and a mapping from a set of source names to target names. For any given file, this relationship can be used to determine the related file. However, UCF does not require that the related file exist.

The root <container> element MAY contain a child <relationships> element. This element MAY contain one or more child <relationship> elements that describe specific relationships. Each relationship element includes attributes type and target that specify a relationship type and a pattern that maps source to target names. The type is a qualified name. UCF defines one relationship type, “metadata.” UCF-based formats can add other types by specifying a namespace. The target pattern is a string that may include the following variables that are substituted when a relationship is resolved:

Variable	Definition	Example
path	the full path to the resource	/a/b/c.d
dir	the directory (without the filename)	/a/b
filename	the path without the directory	c.d
basename	the filename without the extension	c
ext	the filename extension	d

These variables are specified by enclosing their name in braces and preceding the opening brace with a “\$”. Braces may be omitted if the first character after the variable name is not a letter. A target is resolved by copying the ordinary text in the pattern and replacing the variables with their values.

Any file in the example container can have an associated metadata file with the same name as the source file followed by the “.xmp” extension.

#### ***Manifest – META-INF/manifest.xml (Optional)***

An OPTIONAL file with the reserved name “manifest.xml” within the “META-INF” directory at the root level of the container may appear in a valid UCF container. If present, the file’s content MUST be as defined in the ODF 1.0 manifest schema (<http://www.oasis-open.org/committees/download.php/12570/OpenDocument-manifest-schema-v1.0-os.rng>).

The manifest.xml file, if present, MUST NOT be encrypted.

#### ***Metadata – META-INF/metadata.xml (Optional)***

A file with the reserved name “metadata.xml” within the “META-INF” directory at the root level of the container file system may appear in a valid UCF container. This file, if present, MUST be used for container-level metadata. In version 1.0 of OCF, no such container-level metadata is specified.

If the “META-INF/metadata.xml” file exists, its contents MUST be valid XML with namespace-qualified elements to avoid collision with future versions of OCF that MAY specify a particular grammar and namespace for elements and attributes within this file.

Adobe-defined formats based on UCF MUST use XMP to specify metadata (<http://www.adobe.com/products/xmp/>).

#### ***Digital Signatures – META-INF/signatures.xml (Optional)***

An OPTIONAL “signatures.xml” file within the “META-INF” directory at the root level of the container file system holds digital signatures of the container and its contents. The contents of this file is not specified in UCF 1.0. However, a future revision of UCF will define the format for this file. See Appendix D for the likely definition.

### ***Encryption – META-INF/encryption.xml (Optional)***

An OPTIONAL “encryption.xml” file within the “META-INF” directory at the root level of the container file system holds all encryption information on the contents of the container. The contents of this file is not specified in UCF 1.0. However, a future revision of UCF will define the format for this file. See Appendix E for the likely definition.

### ***Rights Management – META-INF/rights.xml (Optional)***

An OPTIONAL file with the name “rights.xml” within the “META-INF” directory at the root level of the container file system is a reserved name in a valid UCF container. This location is reserved for digital rights management (DRM) information for trusted exchange of Publications among rights holders, intermediaries, and users. In version 1.0 of UCF, there is not a REQUIRED format for DRM information, but a future version of the UCF specification MAY specify a particular format for DRM information.

If the “META-INF/rights.xml” file exists, it MUST be a well-formed XML document which uses and conforms to XML Namespaces it uses, and its contents SHOULD be valid XML with namespace-qualified elements to avoid collision with future versions of UCF that MAY specify a particular format this file.

The rights.xml file MUST NOT be encrypted.

When the rights.xml file is not present, the UCF container provides no information indicating any part of the container is rights governed.

## **1.5 Zip Container**

UCF’s Zip Container supports the Zip format as specified by the application note at <http://www.pkware.com/support/zip-application-note>, but with the following constraints and clarifications:

Conforming UCF Zip Containers MUST NOT use the features in the Zip application note that allow Zip files to be split across multiple storage media. Conforming UCF User Agents MUST treat any UCF files that specify that the Zip file is split across multiple storage media as being in error.

Conforming UCF Zip Containers MUST only include uncompressed files or Flate-compressed files within the Zip archive. Conforming UCF User Agents MUST treat any UCF Containers that use compression techniques other than Flate as being in error.

Conforming UCF Zip Containers MAY use the Zip64 extensions and SHOULD only use those extensions when the content requires them. Conforming UCF User Agents MUST support the Zip64 extensions.

Conforming UCF Zip Containers MUST NOT use the encryption features defined by the Zip format; instead, encryption MUST be done using the features described in Section 3.5.5. Conforming UCF User Agents MUST treat any other UCF Zip Containers that use Zip encryption features as being in error.

It is not a requirement that Conforming UCF User Agents preserve information from an UCF Zip Container through load and save operations that do not map to corresponding representation within the UCF Abstract Container; in particular, a Conforming UCF User Agent does not have to preserve CRC values, comment fields or fields that hold file system information corresponding to a particular operating system (e.g., “External file attributes” and “Extra field”)

Conforming UCF Zip Containers MUST encode File System Names using UTF-8.

Here are some details about particular fields in the Zip archive:

On the local file header table, Conforming UCF Zip Containers MUST set the ‘version needed to extract’ fields to the values 10, 20 or 45 in order to match the maximum version level needed by the given file (e.g., 20 if Deflate is needed, 45 if Zip64 is needed). Conforming UCF User Agents MUST treat any other values as being in error.

On the local file header table, Conforming UCF Zip Containers MUST set the ‘compression’ method field to the values 0 or 8. Conforming UCF User Agents MUST treat any other values as being in error.

Conforming UCF User Agents MUST treat UCF Zip Containers with an “Archive decryption header” or an “Archive extra data record” as being in error.

The first file in the Zip Container MUST be a file by the ASCII name of ‘mimetype’ which holds the MIME type for the Zip Container (i.e., “application/epub+zip” as an ASCII string; no padding, white-space or case change). The file MUST be neither compressed nor encrypted and there MUST NOT be an extra field in its Zip header. If this is done, then the Zip Container offers convenient “magic number” support as described in RFC 2048 and the following will hold true:

The bytes “PK” will be at the beginning of the file

The bytes “mimetype” will be at position 30

The actual MIME type (i.e., the ASCII string “application/epub+zip”) will begin at position 38

## 1.6 RELAX NG UCF Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<choice xmlns="http://relaxng.org/ns/structure/1.0">
    <element name="container">
        <attribute name="version">
            <value>1.0</value>
        </attribute>
        <attribute name="xmlns">
            <value>urn:oasis:names:tc:opendocument:xmlns:container
        </value>
        </attribute>
        <optional>
            <element name="rootfiles">
                <oneOrMore>
                    <element name="rootfile">
                        <attribute name="full-path">
```

```

        <text/>
    </attribute>
    <attribute name="media-type">
        <text/>
    </attribute>
</element>
</oneOrMore>
</element>
</optional>
<optional>
    <element name="relationships">
        <oneOrMore>
            <element name="relationship">
                <attribute name="type">
                    <text/>
                </attribute>
                <attribute name="target">
                    <text/>
                </attribute>
            </element>
        </oneOrMore>
    </element>
</optional>
</element>

<element name="signatures">
    <attribute name="xmlns">
        <value>urn:oasis:names:tc:opendocument:xmlns:container</
value>
    </attribute>
    <oneOrMore>
        <element name="Signature" ns="http://www.w3.org/2001/04/xmldsig#">
            <externalRef
                href="http://www.w3.org/Signature/2002/07/xmldsig-core-schema.rng"/>
        </element>
    </oneOrMore>
</element>

<element name="encryption">
    <attribute name="xmlns">
        <value>urn:oasis:names:tc:opendocument:xmlns:container</
value>
    </attribute>
    <oneOrMore>
        <choice>
            <element name="EncryptedData"
ns="http://www.w3.org/2001/04/xmlenc#">
                <externalRef
                    href="http://www.w3.org/Encryption/2002/07/xenc-schema.rng"/>
            </element>
            <element name="EncryptedKey"
ns="http://www.w3.org/2001/04/xmlenc#">
                <externalRef
                    href="http://www.w3.org/Encryption/2002/07/xenc-schema.rng"/>
            </element>
        </choice>
    </oneOrMore>
</element>

```

```

        </element>
    </choice>
</oneOrMore>
</element>
```

</choice>

The following example demonstrates the use of this UCF format to contain a signed and encrypted OEBPS publication with an alternate PDF rendition within a Zip Container.

Ordered list of files in the Zip Container:

```

mimetype
META-INF/container.xml
META-INF/signatures.xml
META-INF/encryption.xml
OEBPS/As You Like It.opf
OEBPS/book.html
OEBPS/images/cover.png
PDF/As You Like It.pdf
```

The mimetype file:

application/epub+zip

The META-INF/container.xml file:

```

<?xml version="1.0"?>
<container version="1.0" xmlns="urn:oasis:names:tc:opendocument:xmlns:container">
    <rootfiles>
        <rootfile full-path="OEBPS/As You Like It.opf"
                  media-type="application/oebps-package+xml" />
        <rootfile full-path="OEBPS/As You Like It.pdf"
                  media-type="application/pdf" />
    </rootfiles>
</container>
```

The META-INF/signatures.xml file:

```

<?xml version="1.0"?>
<signatures xmlns="urn:oasis:names:tc:opendocument:xmlns:container">
    <Signature Id="AsYouLikeItSignature" xmlns="http://www.w3.org/2000/09/xmldsig#">

        <!-- SignedInfo is the information that is actually signed. In this case -->
        <!-- the SHA1 algorithm is used to sign the canonical form of the XML      -->
        <!-- documents enumerated in the Object element below                      -->
        <SignedInfo>
            <CanonicalizationMethod
                Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
            <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
            <Reference URI="#AsYouLikeIt">
                <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                <DigestValue>j6lw3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
            </Reference>
        </SignedInfo>
```

```

<!-- The signed value of the digest above using the DSA algorithm -->
<SignatureValue>MC0CFFrVLtRlk=...</SignatureValue>

<!-- The key to use to validate the signature -->
<KeyInfo>
  <KeyValue>
    <DSAKeyValue>
      <P>...</P><Q>...</Q><G>...</G><Y>...</Y>
    </DSAKeyValue>
  </KeyValue>
</KeyInfo>

<!-- The list documents to sign. Note that the canonical form of XML -->
<!-- documents is signed while the binary form of the other documents -->
<!-- is used -->
<Object>
  <Manifest Id="AsYouLikeIt">
    <Reference URI="OEBPS/As You Like It.opf">
      <Transforms>
        <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-
c14n-20010315"/>
      </Transforms>
    </Reference>
    <Reference URI="OEBPS/book.html">
      <Transforms>
        <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-
c14n-20010315"/>
      </Transforms>
    </Reference>
    <Reference URI="OEBPS/images/cover.png" />
    <Reference URI="PDF/As You Like It.pdf" />
  </Manifest>
</Object>

</Signature>
</signatures>

```

The META-INF/encryption.xml file:

```

<?xml version="1.0"?>
<encryption
  xmlns="urn:oasis:names:tc:opendocument:xmlns:container"
  xmlns:enc="http://www.w3.org/2001/04/xmlenc#"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#">

  <!-- The RSA encrypted AES-128 symmetric key used to encrypt the data -->
  <enc:EncryptedKey Id="EK">
    <enc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
    <ds:KeyInfo>
      <ds:KeyName>John Smith</ds:KeyName>
    </ds:KeyInfo>
    <enc:CipherData>
      <enc:CipherValue>xyzabc...</enc:CipherValue>
    </enc:CipherData>
  </enc:EncryptedKey>
</encryption>

```

```

        </enc:CipherData>
</enc:EncryptedKey>

<!-- Each EncryptedData block identifies a single document that has been -->
<!-- encrypted using the AES-128 algorithm. The data remains stored in it's -->
<!-- encrypted form in the original file within the container. -->
<enc:EncryptedData Id="ED1">
    <enc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes128"/>
    <ds:KeyInfo>
        <ds:RetrievalMethod URI="#EK"
            Type="http://www.w3.org/2001/04/xmlenc#EncryptedKey"/>
    </ds:KeyInfo>
    <enc:CipherData>
        <enc:CipherReference URI="OEBPS/book.html"/>
    </enc:CipherData>
</enc:EncryptedData>

<enc:EncryptedData Id="ED2">
    <enc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes128"/>
    <ds:KeyInfo>
        <ds:RetrievalMethod URI="#EK"
            Type="http://www.w3.org/2001/04/xmlenc#EncryptedKey"/>
    </ds:KeyInfo>
    <enc:CipherData>
        <enc:CipherReference URI="OEBPS/images/cover.png"/>
    </enc:CipherData>
</enc:EncryptedData>

<enc:EncryptedData Id="ED3">
    <enc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes128"/>
    <enc:KeyInfo>
        <enc:RetrievalMethod URI="#EK"
            Type="http://www.w3.org/2001/04/xmlenc#EncryptedKey"/>
    </enc:KeyInfo>
    <enc:CipherData>
        <enc:CipherReference URI="PDF/As You Like It.pdf"/>
    </enc:CipherData>
</enc:EncryptedData>

</encryption>

```

The OEBPS/As You Like It.opf file:

```

<?xml version="1.0"?>
<!DOCTYPE package PUBLIC "+//ISBN 0-9673008-1-9//DTD OEB 1.2 Package//EN"
 "http://openebook.org/dtds/oeb-1.2/oebpkg12.dtd">
<package unique-identifier="Package-ID">
    <metadata>
        <dc-metadata xmlns:dc="http://purl.org/dc/elements/1.0"
            xmlns:oebpackage="http://openebook.org/namespaces/oeb-
        package/1.0">
            <dc:Identifier id="Package-ID">ebook:guid-6B2DF0030656ED9D8</dc:Identifier>
            <dc>Title>As You Like It</dc>Title>
            <dc:Creator role="aut">William Shakespeare</dc:Creator>
        </dc-metadata>
    </metadata>
</package>

```

```
<dc:Identifier>0-7410-1455-6</dc:Identifier>
<dc:Subject></dc:Subject>
<dc>Type></dc>Type>
<dc:Date event="publication">3/24/2000</dc:Date>
<dc:Date event="copyright">1/1/9999</dc:Date>
<dc:Identifier scheme="ISBN">0-7410-1455-6</dc:Identifier>
<dc:Publisher>Project Gutenberg</dc:Publisher>
<dc:Language></dc:Language>
</dc-metadata>
</metadata>
<manifest>
  <item id="4915" href="book.html" media-type="text/x-oeb1-document"/>
  <item id="7184" href="images/cover.png" media-type="image/png" />
</manifest>
<spine>
  <itemref idref="4915"/>
</spine>
</package>
```

The OEBPS/book.html file:

This file would be binary and be encrypted. Its decrypted contents might look something like:

```
<?xml version="1.0" ?>
<!DOCTYPE html PUBLIC
  "+//ISBN 0-9673008-1-9//DTD OEB 1.2 Document//EN"
  "http://openebook.org/dtds/oeb-1.2/oebdoc12.dtd">
<html>
<head>

  ...

</head>
<body>

  ...



  ...

</body>
</html>
```

The OEBPS/images/cover.png file:

This file contains the encrypted binary bytes of the cover.png file.

The OEBPS/As You Like It.pdf file:

This file contains the encrypted binary bytes of the PDF file.

## 1.7 Comparison of UCF and OCF

As described in the introduction, UCF is OCF without the OEBPS dependencies. There are only a few significant differences:

The OCF specification states that the media type of the container must be application/epub+zip, while UCF specifies that UCF-based formats should choose an appropriate media type. OCF implicitly encourages the use of “+zip” to identify Zip-based formats.

OCF requires all XML documents in a container to be compatible with XML 1.1. UCF requires all XML documents to be compatible with XML 1.0. (The use of XML 1.0 follows generally recommended practice to use XML 1.0 rather than XML 1.1 unless XML 1.1 features are required.)

OCF forbids certain characters in names. In addition to those characters, UCF also disallows characters corresponding to the non-printing ASCII codes. While the OCF specifications lists forbidden characters by ASCII names, the UCF specification lists Unicode code points.

OCF requires that two file names in a container be unique after case normalization. UCF also requires that names be unique after character normalization using Unicode normalization form C (canonical decomposition followed by canonical composition).

OCF requires container.xml which in turn requires specification of a rootfile. UCF does not require container.xml. The rationale is that an UCF-based format can specify how to begin processing the container.

UCF adds file relationships, providing an application-independent method for storing and finding metadata associated with container files.

OCF specifies that the metadata.xml file must not be encrypted, while UCF allows this. Even when a publication is protected with encryption (usually to support digital rights management), the IDPF wants reading systems to be able to provide users with useful metadata. IDML (and possibly other UCF formats as well) has more general requirements and needs to leave this as an option for the container creator.

OCF encourages but does not require each publication (in UCF, generalized to document or application) to reside in its own directory within the container. This makes it easier to contain multiple renditions of the publication in the container.

The UCF specification has deferred the definition of encryption and digital signature features to a future revision. This will provide implementers more time to evaluate the proposed definition.

## 1.8 Digital Signatures

Support of digital signatures in UCF has been deferred until a future revision of the specification. However, it is likely that the specification of this feature will match the OCF specification, which follows:

An OPTIONAL “signatures.xml” file within the “META-INF” directory at the root level of the container file system holds digital signatures of the container and its contents. This file is an XML

document whose root element is <signatures>. The <signatures> element contains child elements of type <Signature> as defined by “XML-Signature Syntax and Processing” (<http://www.w3.org/TR/2002/REC-xmldsig-core-20020212>). Signatures can be applied to the publication and any alternate renditions as a whole or to parts of the publication and renditions. XML Signature can specify the signing of any kind of data, not just XML.

The signatusres.xml file MUST NOT be encrypted.

When the signatures.xml file is not present, the UCF container provides no information indicating any part of the container is digitally signed at the container level. It is however possible that digital signing exists within any optional alternate contained renditions.

A RELAX NG UCF schema describing the <signature> element that MUST be the root element of signatures.xml can be found in the Appendix A.

When an UCF agent creates a signature of data in a container, it SHOULD add the new signature as the last child <Signature> element of the <signatures> element in the signatures.xml file.

Each <Signature> in the signatures.xml file identifies by IRI the data to which the signature applies, using the XML Signature <Manifest> element and its <Reference> sub-elements. Individual contained files MAY be signed separately or together. Separately signing each file creates a digest value for the resource that can be validated independently. This approach MAY make a Signature element larger. If files are signed together, the set of signed files can be listed in a single XML Signature <Manifest> element and referred to by one or more <Signature> elements.

Any or all files in the container can be signed in their entirety with the exception of the signatures.xml file since that file will contain the computed signature information. Whether and how the signatures.xml file SHOULD be signed depends on the objective of the signer.

If the signer wants to allow signatures to be added or removed from the container without invalidating the signer’s signature, the signatures.xml file SHOULD NOT be signed.

If the signer wants any addition or removal of a signature to invalidate the signer’s signature, the Enveloped Signature transform (defined in Section 6.6.4 of XML Signature) can be used to sign the entire preexisting signature file excluding the <Signature> being created. This transform would sign all previous signatures, and it would become invalid if a subsequent signature was added to the package.

If the signer wants the removal of an existing signature to invalidate the signer’s signature but also wants to allow the addition of signatures, an XPath transform can be used to sign just the existing signatures. (This is only a suggestion. The particular XPath transform is not a part of UCF specification.)

XML-Signature does not associate any semantics with a signature, however an agent MAY include semantic information, for example, by adding information to the Signature element that describes the signature. XML Signature describes how additional information can be added to a signature (for example, by using the SignatureProperties element).

(This example is informative.)

The following XML expression shows the content of an example “signatures.xml” file, and is based on the examples found in Section 2 of “XML-Signature Syntax and Processing.” It contains one signature, and the signature applies to two resources, OEBFPS/book.html and OEBFPS/images/cover.jpeg, in the container.

```

<signatures>
  <Signature Id="MyFirstSignature" xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
      <CanonicalizationMethod
        Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
      <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
      <Reference URI="#Manifest1">
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>j6lw3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
      </Reference>
    </SignedInfo>
    <SignatureValue>MC0CFFrVLtRlk=...</SignatureValue>
    <KeyInfo>
      <KeyValue>
        <DSAKeyValue>
          <P>...</P><Q>...</Q><G>...</G><Y>...</Y>
        </DSAKeyValue>
      </KeyValue>
    </KeyInfo>
    <Object>
      <Manifest Id="Manifest1">
        <Reference URI="OEBFPS/book.xml">
          <Transforms>
            <Transform
              Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
          </Transforms>
        </Reference>
        <Reference URI="OEBFPS/images/cover.jpeg"/>
      </Manifest>
    </Object>
  </Signature>
</signatures>

```

## 1.9 Encryption

Support of encryption in UCF has been deferred until a future revision of the specification. With one exception, it is likely that the specification of this feature will match the OCF specification, which follows this paragraph. The exception is that it will be possible to specify that a particular algorithm does not require Flate compression of data before encryption.

An OPTIONAL “encryption.xml” file within the “META-INF” directory at the root level of the container file system holds all encryption information on the contents of the container. This file is an XML document whose root element is <encryption>. The <encryption> element contains child elements of type <EncryptedKey> and <EncryptedData> as defined by “XML Encryption Syntax and Processing” (<http://www.w3.org/TR/2002/REC-xmlenc-core-20021210>). Each EncryptedKey element describes how one or more container files are encrypted. Consequently, if any resource within the container is encrypted, “encryption.xml” MUST be present to indicate that the resource is encrypted and provide information on how it is encrypted.

An <EncryptedKey> element describes each encryption key used in the container, while an <EncryptedData> element describes each encrypted file. Each <EncryptedData> element refers to an <EncryptedKey> element, as described in XML Encryption.

A RELAX NG UCF schema describing the <encryption> element that MUST be the root element of encryption.xml can be found in the Appendix A.

When the encryption.xml file is not present, the UCF container provides no information indicating any part of the container is encrypted.

UCF encrypts individual files independently, trading off some security for improved performance, allowing the container contents to be incrementally decrypted. Encryption in this way still exposes the directory structure and file naming of the whole package.

UCF uses XML Encryption to provide a framework for encryption, allowing a variety of algorithms to be used. XML Encryption specifies a process for encrypting arbitrary data and representing the result in XML. Even though an UCF container MAY contain non-XML data, XML Encryption can be used to encrypt all data in an UCF container. UCF encryption supports only encryption of whole files. The encryption.xml file, if present, MUST NOT be encrypted.

Encrypted data replaces unencrypted data in an UCF container. For example, if an image named “photo.jpeg” is encrypted, the contents of the photo.jpeg resource SHOULD be replaced by its encrypted contents. When stored in a Zip container, files MUST be compressed before they are encrypted; Flate compression MUST be used. Within the Zip local file header and central directory, encrypted files SHOULD be listed as stored rather than Flate-compressed.

The following files MUST never be encrypted (regardless of whether default or specific encryption is requested):

- mimetype
- META-INF/container.xml
- META-INF/manifest.xml
- META-INF/metadata.xml
- META-INF/signatures.xml
- META-INF/encryption.xml
- META-INF/rights.xml

Signed resources MAY subsequently be encrypted by using the Decryption Transform for XML Signature. This feature enables an application such as an UCF agent to distinguish data that was encrypted before signing from data that was encrypted after signing. Only data that was encrypted after signing MUST be decrypted before computing the digest used to validate the signature.

(This example is informative.)

In the following example, adapted from Section 2.2.1 of “XML Encryption Syntax and Processing,” the resource image.jpeg is encrypted using a symmetric key algorithm (AES) and the symmetric key is further encrypted using an asymmetric key algorithm (RSA) with a key of John Smith.

```
<encryption
  xmlns ="urn:oasis:names:tc:opendocument:xmlns:container"
```

```
xmlns:enc="http://www.w3.org/2001/04/xmlenc#"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<enc:EncryptedKey Id="EK">
  <enc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
  <ds:KeyInfo>
    <ds:KeyName>John Smith</ds:KeyName>
  </ds:KeyInfo>
  <enc:CipherData>
    <enc:CipherValue>xyzabc</enc:CipherValue>
  </enc:CipherData>
</enc:EncryptedKey>
<enc:EncryptedData Id="ED1">
  <enc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes128"/>
  <ds:KeyInfo>
    <ds:RetrievalMethod URI="#EK"
      Type="http://www.w3.org/2001/04/xmlenc#EncryptedKey"/>
  </ds:KeyInfo>
  <enc:CipherData>
    <enc:CipherReference URI="image.jpeg"/>
  </enc:CipherData>
</enc:EncryptedData>
</encryption>
```

# Appendix B. IDML Variants

This appendix discusses the variants of IDML.

The following are variants of IDML that can be written by InDesign CS4 and InCopy CS4:

- InDesign Snippets (.idms files)
- InCopy ICML
- InCopy Assignment (.icma files)

All of the above file variants produce “standalone” XML files, rather than the Zip package format used by IDML.

Snippets and ICML files can be distinguished by the `SnippetType` attribute of an `<?aid>` element at the beginning of the file:

Snippet file      `<?aid SnippetType="PageItem"?>`

ICML file      `<?aid SnippetType="InCopyInterchange"?>`

An InCopy assignment file can be distinguished by the type attribute of an `<?aid>` element at the beginning of the file.

Assignment file    `<?aid type="assignment"?>`

## 1.1 InDesign Snippets (.idms files)

When you export the contents of a page as an InDesign snippet, InDesign creates a single XML file that contains the elements on the page and any style, color, layer, or other elements that might be needed to correctly re-create and format the elements in the snippet file when it is imported into an InDesign document.

InDesign snippet files are also used by InDesign’s Library feature—the library items themselves are InDesign snippets.

When you place (import) an InDesign Snippet into an InDesign document, InDesign makes a variety of decisions based on the formatting specified in the Snippet document and the formatting of the document itself. If a color used in a Snippet file already exists in the InDesign document, for example, InDesign applies the color in the document to the incoming snippet objects, rather than using the color definition in the Snippet file. This matching is based on the name of the color. If the color does not exist in the document, however, InDesign will add the color defined in the Snippet file to the list of colors in the document, and will apply that color to the incoming objects.

An InDesign Snippet file contains the same elements as you would find in `<Spread>` and `<Story>` elements in an IDML file, and also contains any `<ParagraphStyle>`, `<CharacterStyle>`, `<Color>`, and other elements required to support the formatting specified by the elements in the file. An

InDesign Snippet file does not contain most of the preferences and defaults elements found in an IDML document. The following table describes a number of points specific to InDesign snippets.

**Table 1. Snippet Export Policies**

Element	Export Policy
Text frames and stories	If all of the frames of a story are selected when you export the snippet, the entire story will be exported with the entire text contents, notes, tables, changes, etc., as child elements of the <Story> element. If all of the frames are not selected, then only the selected frames are exported. Text contents, notes, tables, changes, etc., that appear within the text frame are exported as child elements of the <TextFrame> element.
Hyperlink sources	Hyperlink sources are exported if the hyperlink source exists in/on an exported page item.
Hyperlink destinations	Hyperlink destinations are exported if the hyperlink destination exists in/on an exported page item.
Hyperlink page destinations	Not exported.
Hyperlinks	Hyperlinks are exported if the associated hyperlink source is exported. Hyperlink destination references are set to nothing if the hyperlink destination itself is not exported.
Bookmarks	Bookmarks are exported if the associated hyperlink destination is exported.
XML items	If the parent of the XML item (an element, processing instruction, comment, etc.) is not exported, the StoryOffset attribute of the XML item is removed. This way, the XML item will be added as the last child element of the root XML element when the snippet is placed in an InDesign document.

The following is an example InDesign Snippet file containing a simple rectangle (thumbnail image data was removed to save space).

#### **IDML Example 1. InDesign Snippet Example**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<?aid style="50" type="snippet" readerVersion="6.0" featureSet="257"
product="6.0(351)" ?>
<?aid SnippetType="PageItem"?>
<Document DOMVersion="6.0" Self="d">
  <Color Self="Color/Black" Model="Process" Space="CMYK"
    ColorValue="0 0 0 100" ColorOverride="Specialblack"
    AlternateSpace="NoAlternateColor" AlternateColorValue="" Name="Black"
    ColorEditable="false" ColorRemovable="false" Visible="true"
    SwatchCreatorID="7937"/>
  <Swatch Self="Swatch/None" Name="None" ColorEditable="false"
    ColorRemovable="false" Visible="true" SwatchCreatorID="7937"/>
  <StrokeStyle Self="StrokeStyle/$ID/Solid" Name="$ID/Solid"/>
<RootCharacterStyleGroup Self="u6a">
```

```

<CharacterStyle Self="CharacterStyle/$ID/[No character style]"
Imported="false" Name="$ID/[No character style]"/>
</RootCharacterStyleGroup>
<NumberingList Self="NumberingList/$ID/[Default]" Name="$ID/[Default]"
ContinueNumbersAcrossStories="false" ContinueNumbersAcrossDocuments="false"/>
<RootParagraphStyleGroup Self="u69">
  <ParagraphStyle Self="ParagraphStyle/$ID/[No paragraph style]"
Name="$ID/[No paragraph style]" Imported="false" FillColor="Color/Black"
FontStyle="Regular" PointSize="12" HorizontalScale="100"
KerningMethod="$ID/Metrics" Ligatures="true" PageNumberType="AutoPageNumber"
StrokeWeight="1" Tracking="0" Composer="HL Composer" DropCapCharacters="0"
DropCapLines="0" BaselineShift="0" Capitalization="Normal"
StrokeColor="Swatch/None" HyphenateLadderLimit="3" VerticalScale="100"
LeftIndent="0" RightIndent="0" FirstLineIndent="0" AutoLeading="120"
AppliedLanguage="$ID/English: USA" Hyphenation="true"
HyphenateAfterFirst="2" HyphenateBeforeLast="2"
HyphenateCapitalizedWords="true" HyphenateWordsLongerThan="5" NoBreak="false"
HyphenationZone="36" SpaceBefore="0" SpaceAfter="0" Underline="false"
OTFFigureStyle="Default" DesiredWordSpacing="100" MaximumWordSpacing="133"
MinimumWordSpacing="80" DesiredLetterSpacing="0" MaximumLetterSpacing="0"
MinimumLetterSpacing="0" DesiredGlyphScaling="100" MaximumGlyphScaling="100"
MinimumGlyphScaling="100" StartParagraph="Anywhere"
KeepAllLinesTogether="false" KeepWithNext="0" KeepFirstLines="2"
KeepLastLines="2" Position="Normal" StrikeThru="false"
CharacterAlignment="AlignEmCenter" KeepLinesTogether="false"
StrokeTint="-1" FillTint="-1" OverprintStroke="false" OverprintFill="false"
GradientStrokeAngle="0" GradientFillAngle="0" GradientStrokeLength="-1"
GradientFillLength="-1" GradientStrokeStart="0 0"
GradientFillStart="0 0" Skew="0" RuleAboveLineWeight="1" RuleAboveTint="-1"
RuleAboveOffset="0" RuleAboveLeftIndent="0" RuleAboveRightIndent="0"
RuleAboveWidth="ColumnWidth" RuleBelowLineWeight="1" RuleBelowTint="-1"
RuleBelowOffset="0" RuleBelowLeftIndent="0" RuleBelowRightIndent="0"
RuleBelowWidth="ColumnWidth" RuleAboveOverprint="false"
RuleBelowOverprint="false" RuleAbove="false" RuleBelow="false"
LastLineIndent="0" HyphenateLastWord="true" ParagraphBreakType="Anywhere"
SingleWordJustification="FullyJustified" OTFOrdinal="false"
OTFFraction="false" OTFDiscretionaryLigature="false" OTFTitling="false"
RuleAboveGapTint="-1" RuleAboveGapOverprint="false" RuleBelowGapTint="-1"
RuleBelowGapOverprint="false" Justification="LeftAlign"
DropcapDetail="0" PositionalForm="None" OTFMark="true" HyphenWeight="5"
OTFLocale="true" HyphenateAcrossColumns="true" KeepRuleAboveInFrame="false"
IgnoreEdgeAlignment="false" OTFSashedZero="false" OTFStylisticSets="0"
OTFHistorical="false" OTFContextualAlternate="true"
UnderlineGapOverprint="false" UnderlineGapTint="-1" UnderlineOffset="-9999"
UnderlineOverprint="false" UnderlineTint="-1" UnderlineWeight="-9999"
StrikeThroughGapOverprint="false" StrikeThroughGapTint="-1"
StrikeThroughOffset="-9999" StrikeThroughOverprint="false"
StrikeThroughTint="-1" StrikeThroughWeight="-9999" MiterLimit="4"
StrokeAlignment="OutsideAlignment" EndJoin="MiterEndJoin" OTFSwash="false"
Tsume="0" LeadingAki="-1" TrailingAki="-1"
KinsokuType="KinsokuPushInFirst" KinsokuHangType="None" BunriKinshi="true"
RubyOpenTypePro="true" RubyFontSize="-1" RubyAlignment="RubyJIS"
RubyType="PerCharacterRuby" RubyParentSpacing="RubyParent121Aki"
RubyXScale="100" RubyYScale="100" RubyXOffset="0" RubyYOffset="0"

```

```

RubyPosition="AboveRight" RubyAutoAlign="true"
RubyParentOverhangAmount="RubyOverhangOneRuby" RubyOverhang="false"
RubyAutoScaling="false" RubyParentScalingPercent="66" RubyTint="-1"
RubyOverprintFill="Auto" RubyStrokeTint="-1" RubyOverprintStroke="Auto"
RubyWeight="-1" KentenKind="None" KentenFontSize="-1" KentenXScale="100"
KentenYScale="100" KentenPlacement="0" KentenAlignment="AlignKentenCenter"
KentenPosition="AboveRight" KentenCustomCharacter=""
KentenCharacterSet="CharacterInput" KentenTint="-1"
KentenOverprintFill="Auto" KentenStrokeTint="-1" KentenOverprintStroke="Auto"
KentenWeight="-1" Tatechuyoko="false" TatechuyokoXOffset="0"
TatechuyokoYOffset="0" AutoTcy="0" AutoTcyIncludeRoman="false" Jidori="0"
GridGyoudori="0" GridAlignFirstLineOnly="false" GridAlignment="None"
CharacterRotation="0" RotateSingleByteCharacters="false" Rensuujji="true"
ShataiMagnification="0" ShataiDegreeAngle="4500" ShataiAdjustTsume="true"
ShataiAdjustRotation="false" Warichu="false" WarichuLines="2" WarichuSize="50"
WarichuLineSpacing="0" WarichuAlignment="Auto" WarichuCharsBeforeBreak="2"
WarichuCharsAfterBreak="2" OTFHVKana="false" OTFProportionalMetrics="false"
OTFRomanItalics="false" LeadingModel="LeadingModelAkiBelow"
ScaleAffectsLineHeight="false" ParagraphGyoudori="false"
CjkGridTracking="false" GlyphForm="None" RubyAutoTcyDigits="0"
RubyAutoTcyIncludeRoman="false" RubyAutoTcyAutoScale="true"
TreatIdeographicSpaceAsSpace="false" AllowArbitraryHyphenation="false"
BulletsAndNumberingListType="NoList" NumberingStartAt="1"
NumberingLevel="1" NumberingContinue="true" NumberingApplyRestartPolicy="true"
BulletsAlignment="LeftAlign" NumberingAlignment="LeftAlign"
NumberingExpression="^#.^t" BulletsTextAfter="^t" DigitsType="DefaultDigits"
Kashidas="DefaultKashidas" DiacriticPosition="OpentypePosition"
CharacterDirection="DefaultDirection"
ParagraphDirection="LeftToRightDirection"
ParagraphJustification="DefaultJustification" XOffsetDiacritic="0"
YOffsetDiacritic="0" OTFOverlapSwash="false" OTFStylisticAlternate="false"
OTFJustificationAlternate="false" OTFSretchedAlternate="false"
KeyboardDirection="DefaultDirection">
<Properties>
  <Leading type="enumeration">Auto</Leading>
  <AppliedFont type="string">Times New Roman</AppliedFont>
  <RuleAboveColor type="string">Text Color</RuleAboveColor>
  <RuleBelowColor type="string">Text Color</RuleBelowColor>
  <RuleAboveType type="object">StrokeStyle/$ID/Solid</RuleAboveType>
  <RuleBelowType type="object">StrokeStyle/$ID/Solid</RuleBelowType>
  <BalanceRaggedLines type="enumeration">NoBalancing</BalanceRaggedLines>
  <RuleAboveGapColor type="object">Swatch/None</RuleAboveGapColor>
  <RuleBelowGapColor type="object">Swatch/None</RuleBelowGapColor>
  <UnderlineColor type="string">Text Color</UnderlineColor>
  <UnderlineGapColor type="object">Swatch/None</UnderlineGapColor>
  <UnderlineType type="object">StrokeStyle/$ID/Solid</UnderlineType>
  <StrikeThroughColor type="string">Text Color</StrikeThroughColor>
  <StrikeThroughGapColor type="object">Swatch/None</StrikeThroughGapColor>
  <StrikeThroughType type="object">StrokeStyle/$ID/Solid
  </StrikeThroughType>
  <Mojikumi type="enumeration">Nothing</Mojikumi>
  <KinsokuSet type="enumeration">Nothing</KinsokuSet>
  <RubyFont type="string">$ID/</RubyFont>
  <RubyFontStyle type="enumeration">Nothing</RubyFontStyle>

```

```

<RubyFill type="string">Text Color</RubyFill>
<RubyStroke type="string">Text Color</RubyStroke>
<KentenFont type="string">$ID/<KentenFont>
<KentenFontStyle type="enumeration">Nothing</KentenFontStyle>
<KentenFillColor type="string">Text Color</KentenFillColor>
<KentenStrokeColor type="string">Text Color</KentenStrokeColor>
<BulletChar BulletCharacterType="UnicodeOnly"
BulletCharacterValue="8226"/>
<NumberingFormat type="string">1, 2, 3, 4...</NumberingFormat>
<BulletsFont type="string">$ID/<BulletsFont>
<BulletsFontStyle type="enumeration">Nothing</BulletsFontStyle>
<AppliedNumberingList type="object">NumberingList/$ID/[Default]
</AppliedNumberingList>
<NumberingRestartPolicies RestartPolicy="AnyPreviousLevel"
LowerLevel="0" UpperLevel="0"/>
<BulletsCharacterStyle type="object">CharacterStyle/$ID/[No character
style]</BulletsCharacterStyle>
<NumberingCharacterStyle type="object">CharacterStyle/$ID/[No character
style]</NumberingCharacterStyle>
</Properties>
</ParagraphStyle>
</RootParagraphStyleGroup>
<RootObjectStyleGroup Self="u83">
<ObjectStyle Self="ObjectStyle/$ID/[None]" Name="$ID/[None]"
AppliedParagraphStyle="ParagraphStyle/$ID/[No paragraph style]"
FillColor="Swatch/None" FillTint="-1" StrokeWeight="0" MiterLimit="4"
EndCap="ButtEndCap" EndJoin="MiterEndJoin" StrokeType="StrokeStyle/$ID/Solid"
LeftLineEnd="None" RightLineEnd="None" StrokeColor="Swatch/None"
StrokeTint="-1" CornerRadius="12" GapColor="Swatch/None" GapTint="-1"
StrokeAlignment="CenterAlignment" Nonprinting="false" GradientFillAngle="0"
GradientStrokeAngle="0" AppliedNamedGrid="n" CornerOption="None">
<TextFramePreference TextColumnCount="1" TextColumnGutter="12"
TextColumnFixedWidth="144" UseFixedColumnWidth="false"
FirstBaselineOffset="AscentOffset" MinimumFirstBaselineOffset="0"
VerticalJustification="TopAlign" VerticalThreshold="0" IgnoreWrap="false">
<Properties>
<InsetSpacing type="list">
<ListItem type="unit">0</ListItem>
<ListItem type="unit">0</ListItem>
<ListItem type="unit">0</ListItem>
<ListItem type="unit">0</ListItem>
</InsetSpacing>
</Properties>
</TextFramePreference>
<BaselineFrameGridOption UseCustomBaselineFrameGrid="false"
StartingOffsetForBaselineFrameGrid="0"
BaselineFrameGridRelativeOption="TopOfInset"
BaselineFrameGridIncrement="12">
<Properties>
<BaselineFrameGridColor type="enumeration">LightBlue
</BaselineFrameGridColor>
</Properties>
</BaselineFrameGridOption>
<AnchoredObjectSetting AnchoredPosition="InlinePosition"

```

```

    SpineRelative="false" LockPosition="false" PinPosition="true"
    AnchorPoint="BottomRightAnchor" HorizontalAlignment="LeftAlign"
    HorizontalReferencePoint="TextFrame" VerticalAlignment="BottomAlign"
    VerticalReferencePoint="LineBaseline" AnchorXoffset="0" AnchorYoffset="0"
    AnchorSpaceAbove="0"/>
    <TextWrapPreference Inverse="false" ApplyToMasterPageOnly="false"
        TextWrapSide="BothSides" TextWrapMode="None">
        <Properties>
            <TextWrapOffset Top="0" Left="0" Bottom="0" Right="0"/>
        </Properties>
        <ContourOption ContourType="SameAsClipping" IncludeInsideEdges="false"
            ContourPathName="$ID//"/>
    </TextWrapPreference>
    <StoryPreference OpticalMarginAlignment="false" OpticalMarginSize="12"
        FrameType="TextFrameType" StoryOrientation="Horizontal"
        StoryDirection="UnknownDirection"/>
    <FrameFittingOption LeftCrop="0" TopCrop="0" RightCrop="0" BottomCrop="0"
        FittingOnEmptyFrame="None" FittingAlignment="TopLeftAnchor"/>
</ObjectStyle>
</RootObjectStyleGroup>
<TransparencyDefaultContainerObject>
    <TransparencySetting>
        <BlendingSetting BlendMode="Normal" Opacity="100" KnockoutGroup="false"
            IsolateBlending="false"/>
        <DropShadowSetting Mode="None" BlendMode="Multiply" Opacity="75" XOffset="7"
            YOffset="7" Size="5" EffectColor="n" Noise="0" Spread="0"
            UseGlobalLight="false" KnockedOut="true" HonorOtherEffects="false"/>
        <FeatherSetting Mode="None" Width="9" CornerType="Diffusion" Noise="0"
            ChokeAmount="0"/>
        <InnerShadowSetting Applied="false" EffectColor="n" BlendMode="Multiply"
            Opacity="75" Angle="120" Distance="7" UseGlobalLight="false" ChokeAmount="0"
            Size="7" Noise="0"/>
        <OuterGlowSetting Applied="false" BlendMode="Screen" Opacity="75" Noise="0"
            EffectColor="n" Technique="Softer" Spread="0" Size="7"/>
        <InnerGlowSetting Applied="false" BlendMode="Screen" Opacity="75" Noise="0"
            EffectColor="n" Technique="Softer" Spread="0" Size="7"
            Source="EdgeSourced"/>
        <BevelAndEmbossSetting Applied="false" Style="InnerBevel"
            Technique="SmoothContour" Depth="100" Direction="Up" Size="7"
            Soften="0" Angle="120" Altitude="30" UseGlobalLight="false"
            HighlightColor="n" HighlightBlendMode="Screen" HighlightOpacity="75"
            ShadowColor="n" ShadowBlendMode="Multiply" ShadowOpacity="75"/>
        <SatinSetting Applied="false" EffectColor="n" BlendMode="Multiply"
            Opacity="50" Angle="120" Distance="7" Size="7" InvertEffect="false"/>
        <DirectionalFeatherSetting Applied="false" LeftWidth="0"
            RightWidth="0" TopWidth="0" BottomWidth="0" ChokeAmount="0" Angle="0"
            FollowShapeMode="LeadingEdge" Noise="0"/>
        <GradientFeatherSetting Applied="false" Type="Linear" Angle="0" Length="0"
            GradientStart="0 0" HiliteAngle="0" HiliteLength="0"/>
    </TransparencySetting>
    <StrokeTransparencySetting>
        <BlendingSetting BlendMode="Normal" Opacity="100" KnockoutGroup="false"
            IsolateBlending="false"/>
        <DropShadowSetting Mode="None" BlendMode="Multiply" Opacity="75" XOffset="7"

```

```

YOffset="7" Size="5" EffectColor="n" Noise="0" Spread="0"
UseGlobalLight="false" KnockedOut="true" HonorOtherEffects="false"/>
<FeatherSetting Mode="None" Width="9" CornerType="Diffusion" Noise="0"
ChokeAmount="0"/>
<InnerShadowSetting Applied="false" EffectColor="n" BlendMode="Multiply"
Opacity="75" Angle="120" Distance="7" UseGlobalLight="false" ChokeAmount="0"
Size="7" Noise="0"/>
<OuterGlowSetting Applied="false" BlendMode="Screen" Opacity="75" Noise="0"
EffectColor="n" Technique="Softer" Spread="0" Size="7"/>
<InnerGlowSetting Applied="false" BlendMode="Screen" Opacity="75" Noise="0"
EffectColor="n" Technique="Softer" Spread="0" Size="7"
Source="EdgeSourced"/>
<BevelAndEmbossSetting Applied="false" Style="InnerBevel"
Technique="SmoothContour" Depth="100" Direction="Up" Size="7"
Softnen="0" Angle="120" Altitude="30" UseGlobalLight="false"
HighlightColor="n" HighlightBlendMode="Screen" HighlightOpacity="75"
ShadowColor="n" ShadowBlendMode="Multiply" ShadowOpacity="75"/>
<SatinSetting Applied="false" EffectColor="n" BlendMode="Multiply"
Opacity="50" Angle="120" Distance="7" Size="7" InvertEffect="false"/>
<DirectionalFeatherSetting Applied="false" LeftWidth="0"
RightWidth="0" TopWidth="0" BottomWidth="0" ChokeAmount="0" Angle="0"
FollowShapeMode="LeadingEdge" Noise="0"/>
<GradientFeatherSetting Applied="false" Type="Linear" Angle="0" Length="0"
GradientStart="0 0" HiliteAngle="0" HiliteLength="0"/>
</StrokeTransparencySetting>
<FillTransparencySetting>
<BlendingSetting BlendMode="Normal" Opacity="100" KnockoutGroup="false"
IsolateBlending="false"/>
<DropShadowSetting Mode="None" BlendMode="Multiply" Opacity="75" XOffset="7"
YOffset="7" Size="5" EffectColor="n" Noise="0" Spread="0"
UseGlobalLight="false" KnockedOut="true" HonorOtherEffects="false"/>
<FeatherSetting Mode="None" Width="9" CornerType="Diffusion" Noise="0"
ChokeAmount="0"/>
<InnerShadowSetting Applied="false" EffectColor="n" BlendMode="Multiply"
Opacity="75" Angle="120" Distance="7" UseGlobalLight="false" ChokeAmount="0"
Size="7" Noise="0"/>
<OuterGlowSetting Applied="false" BlendMode="Screen" Opacity="75" Noise="0"
EffectColor="n" Technique="Softer" Spread="0" Size="7"/>
<InnerGlowSetting Applied="false" BlendMode="Screen" Opacity="75" Noise="0"
EffectColor="n" Technique="Softer" Spread="0" Size="7"
Source="EdgeSourced"/>
<BevelAndEmbossSetting Applied="false" Style="InnerBevel"
Technique="SmoothContour" Depth="100" Direction="Up" Size="7"
Softnen="0" Angle="120" Altitude="30" UseGlobalLight="false"
HighlightColor="n" HighlightBlendMode="Screen" HighlightOpacity="75"
ShadowColor="n" ShadowBlendMode="Multiply" ShadowOpacity="75"/>
<SatinSetting Applied="false" EffectColor="n" BlendMode="Multiply"
Opacity="50" Angle="120" Distance="7" Size="7" InvertEffect="false"/>
<DirectionalFeatherSetting Applied="false" LeftWidth="0"
RightWidth="0" TopWidth="0" BottomWidth="0" ChokeAmount="0" Angle="0"
FollowShapeMode="LeadingEdge" Noise="0"/>
<GradientFeatherSetting Applied="false" Type="Linear" Angle="0" Length="0"
GradientStart="0 0" HiliteAngle="0" HiliteLength="0"/>
</FillTransparencySetting>
```

```

<ContentTransparencySetting>
  <BlendingSetting BlendMode="Normal" Opacity="100" KnockoutGroup="false"
    IsolateBlending="false"/>
  <DropShadowSetting Mode="None" BlendMode="Multiply" Opacity="75" XOffset="7"
    YOffset="7" Size="5" EffectColor="n" Noise="0" Spread="0"
    UseGlobalLight="false" KnockedOut="true" HonorOtherEffects="false"/>
  <FeatherSetting Mode="None" Width="9" CornerType="Diffusion" Noise="0"
    ChokeAmount="0"/>
  <InnerShadowSetting Applied="false" EffectColor="n" BlendMode="Multiply"
    Opacity="75" Angle="120" Distance="7" UseGlobalLight="false" ChokeAmount="0"
    Size="7" Noise="0"/>
  <OuterGlowSetting Applied="false" BlendMode="Screen" Opacity="75" Noise="0"
    EffectColor="n" Technique="Softer" Spread="0" Size="7"/>
  <InnerGlowSetting Applied="false" BlendMode="Screen" Opacity="75" Noise="0"
    EffectColor="n" Technique="Softer" Spread="0" Size="7"
    Source="EdgeSourced"/>
  <BevelAndEmbossSetting Applied="false" Style="InnerBevel"
    Technique="SmoothContour" Depth="100" Direction="Up" Size="7"
    Soften="0" Angle="120" Altitude="30" UseGlobalLight="false"
    HighlightColor="n" HighlightBlendMode="Screen" HighlightOpacity="75"
    ShadowColor="n" ShadowBlendMode="Multiply" ShadowOpacity="75"/>
  <SatinSetting Applied="false" EffectColor="n" BlendMode="Multiply"
    Opacity="50" Angle="120" Distance="7" Size="7" InvertEffect="false"/>
  <DirectionalFeatherSetting Applied="false" LeftWidth="0"
    RightWidth="0" TopWidth="0" BottomWidth="0" ChokeAmount="0" Angle="0"
    FollowShapeMode="LeadingEdge" Noise="0"/>
  <GradientFeatherSetting Applied="false" Type="Linear" Angle="0" Length="0"
    GradientStart="0 0" HiliteAngle="0" HiliteLength="0"/>
</ContentTransparencySetting>
</TransparencyDefaultContainerObject>
<Layer Self="ub6" Name="Layer 1" Visible="true" Locked="false" IgnoreWrap="false"
  ShowGuides="true" LockGuides="false" UI="true" Expendable="true"
  Printable="true">
  <Properties>
    <LayerColor type="enumeration">LightBlue</LayerColor>
  </Properties>
</Layer>
<Spread Self="ub9">
  <Rectangle Self="ud0" StoryTitle="$ID/" ContentType="GraphicType"
    GradientFillStart="0 0" GradientFillLength="0" GradientFillAngle="0"
    GradientStrokeStart="0 0" GradientStrokeLength="0" GradientStrokeAngle="0"
    ItemLayer="ub6" Locked="false" LocalDisplaySetting="Default"
    GradientFillHiliteLength="0" GradientFillHiliteAngle="0"
    GradientStrokeHiliteLength="0" GradientStrokeHiliteAngle="0"
    AppliedObjectStyle="ObjectStyle/$ID/[None]" ItemTransform="1 0 0 1 0 0">
    <Properties>
      <PathGeometry>
        <GeometryPathType PathOpen="false">
          <PathPointArray>
            <PathPointType Anchor="72 -324" LeftDirection="72 -324"
              RightDirection="72 -324"/>
            <PathPointType Anchor="72 -252" LeftDirection="72 -252"
              RightDirection="72 -252"/>
            <PathPointType Anchor="144 -252" LeftDirection="144 -252"
              RightDirection="144 -252"/>
          </PathPointArray>
        </GeometryPathType>
      </PathGeometry>
    </Properties>
  </Rectangle>
</Spread>

```

```

        RightDirection="144 -252"/>
        <PathPointType Anchor="144 -324" LeftDirection="144 -324"
        RightDirection="144 -324"/>
    </PathPointArray>
</GeometryPathType>
</PathGeometry>
</Properties>
<TextWrapPreference Inverse="false" ApplyToMasterPageOnly="false"
TextWrapSide="BothSides" TextWrapMode="None">
<Properties>
    <TextWrapOffset Top="0" Left="0" Bottom="0" Right="0"/>
</Properties>
</TextWrapPreference>
<InCopyExportOption IncludeGraphicProxies="true"
IncludeAllResources="false"/>
</Rectangle>
</Spread>
<?xpacket begin="i>" id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="Adobe XMP Core 4.2.2-c063 53.352624,
2008/07/30-18:12:18      ">
    <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
        <rdf:Description rdf:about="">
            xmlns:dc="http://purl.org/dc/elements/1.1/">
            <dc:format>application/x-indesign</dc:format>
        </rdf:Description>
        <rdf:Description rdf:about="">
            xmlns:xmp="http://ns.adobe.com/xap/1.0/"
            xmlns:xmpGImg="http://ns.adobe.com/xap/1.0/g/img/">
            <xmp:CreatorTool>Adobe InDesign 6.0</xmp:CreatorTool>
            <xmp:CreateDate>2008-09-22T13:45:04-07:00</xmp:CreateDate>
            <xmp:MetadataDate>2008-09-22T13:45:04-07:00</xmp:MetadataDate>
            <xmp:ModifyDate>2008-09-22T13:45:04-07:00</xmp:ModifyDate>
            <xmp:Thumbnails>
                <rdf:Alt>
                    <rdf:li rdf:parseType="Resource">
                        <xmpGImg:format>JPEG</xmpGImg:format>
                        <xmpGImg:width>512</xmpGImg:width>
                        <xmpGImg:height>512</xmpGImg:height>
                        <xmpGImg:image></xmpGImg:image>
                    </rdf:li>
                </rdf:Alt>
            </xmp:Thumbnails>
        </rdf:Description>
        <rdf:Description rdf:about="">
            xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/"
            xmlns:stEvt="http://ns.adobe.com/xap/1.0/sType/ResourceEvent#">
            <xmpMM:InstanceID>xmp.id:60DDA861CE88DD11AB59D86C578039DD
            </xmpMM:InstanceID>
            <xmpMM:DocumentID>xmp.did:60DDA861CE88DD11AB59D86C578039DD
            </xmpMM:DocumentID>
            <xmpMM:OriginalDocumentID>xmp.did:60DDA861CE88DD11AB59D86C578039DD
            </xmpMM:OriginalDocumentID>
            <xmpMM:History>
                <rdf:Seq>

```

```

<rdf:li rdf:parseType="Resource">
  <stEvt:action>created</stEvt:action>
  <stEvt:instanceID>xmp.iid:60DDA861CE88DD11AB59D86C578039DD
  </stEvt:instanceID>
  <stEvt:when>2008-09-22T13:45:04-07:00</stEvt:when>
  <stEvt:softwareAgent>Adobe InDesign 6.0</stEvt:softwareAgent>
</rdf:li>
</rdf:Seq>
</xmpMM:History>
</rdf:Description>
</rdf:RDF>
</x:xmpmeta>
<?xpacket end="r"?>
</Document>

```

## 1.2 ICML

ICML is the native format for InCopy standalone documents. ICML is a standalone XML file, not a Zip archive as used by IDML. InCopy can open ICML files, InDesign can place ICML files in a layout, but cannot export as ICML.

Inside the ICML file, however, you'll find elements that are identical to the corresponding elements in an IDML file. At base, an ICML document contains a `<story>` element and `<ParagraphStyle>`, `<CharacterStyle>`, `<Color>`, and other elements required to support the formatting used in the `<story>` element.

For more on placing ICML documents in an InDesign document, refer to the online help.

The structure of a very simplified ICML document is shown below (in this example, “...” indicates that there can be more of the preceding element).

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<?aid style="50" type="snippet" readerVersion="6.0" featureSet="513"
product="6.0(329)" ?>
<?aid SnippetType="InCopyInterchange"?>
<Document DOMversion="6.0" Self="d">
  <Color/>
  ...
  <Ink/>
  ...
  <Swatch/>
  ...
  <StrokeStyle/>
  ...
  <FontFamily>
    <Font/>
  ...
  </FontFamily>
  ...
  <CompositeFont>
    <CompositeFontEntry>

```

```

...
</CompositeFont>
...
<RootCharacterStyleGroup>
  <CharacterStyle/>
...
</RootCharacterStyleGroup>
<NumberingList/>
<RootParagraphStyleGroup>
  <ParagraphStyle/>
...
</RootParagraphStyleGroup>
<RootCellStyleGroup>
  <CellStyle/>
...
</RootCellStyleGroup>
<RootTableStyleGroup>
  <TableStyle/>
...
</RootTableStyleGroup>
<ParagraphStyleGroup>
  <ParagraphStyle/>
...
</ParagraphStyleGroup>
...
<CharacterStyleGroup>
  <CharacterStyle/>
...
</CharacterStyleGroup>
...
<TableStyleGroup>
  <TableStyle/>
...
</TableStyleGroup>
...
<CellStyleGroup>
  <CellStyle/>
...
</CellStyleGroup>
...
<FootnoteOption/>
<ConditionalTextPreference/>
<TextVariable/>
...
<XMLTag/>
<CrossReferenceFormat>
  <BuildingBlock/>
</CrossReferenceFormat>
...
<Story>
  <StoryPreference/>
  <MetadataPacketPreference/>
  <InCopyExportOption/>
  <StandaloneDocumentPreference/>

```

```
<ParagraphStyleRange>
  <CharacterStyleRange>
    <Content></Content>
  </CharacterStyleRange>
  ...
</ParagraphStyleRange>
...
</Story>
</Document>
```

## 1.3 InCopy Assignments (ICMA)

ICMA is a variant of ICML that encapsulates the data needed to support an InCopy assignment. Because InCopy assignments are always closely bound to a specific InDesign layout, ICMA files contain almost all of the elements defined by the IDML specification. ICMA documents are standalone XML files, not Zip archives.

ICMA documents do not include various InDesign application defaults elements that appear in IDML. The following IDML elements are not included in an exported ICMA document: `<DataMerge>`, `<DataMergeOption>`, `<PrintPreference>`, `<IndexOptions>`, `<LayoutAdjustmentPreference>`, `<ButtonPreference>`, `<PolygonPreference>`, `<FlattenerSettings>`, `<TOCStyle>`, `<TrapStyle>`, `<XMLElement>`, and `<XMLViewPreference>`. ICMA documents do not include the XML structure associated with the stories in the assignment; that data is stored in the InDesign document containing the assignment.

# Appendix C. IDML Defaults

InDesign's IDML defaults are stored in the file Predef.iddx, which is stored inside the Default folder inside the Presets folder in the InDesign application folder. When opening and converting an IDML document, InDesign will use this file to provide default values for any elements or attributes that are not present in the IDML document. These default values ensure consistency when converting IDML files to InDesign documents.

Do not make changes to the Predef.iddx file.

The following shows the contents of the Predef.iddx file.

## 1.1.1 <Document> Attributes

The <Document> element contains all of the other elements in the Predef.iddx file.

Attribute Name	Value
DOMVersion	"6.0"
Self	"d"
StoryList	"u9b"
ZeroPoint	"0 0"
ActiveLayer	"ub6"
CMYKProfile	"U.S. Web Coated (SWOP) v2"
RGBProfile	"sRGB IEC61966-2.1"
SolidColorIntent	"UseColorSettings"
AfterBlendingIntent	"UseColorSettings"
DefaultImageIntent	"UseColorSettings"
RGBPolicy	"PreserveEmbeddedProfiles"
CMYKPolicy	"CombinationOfPreserveAndSafeCmyk"
AccurateLABSpots	"false"

## 1.1.2 Languages

The default languages are the following ("%3a" in the following listing represents a colon):

```
[No Language], English%3a USA, English%3a USA Medical, English%3a USA Legal,
French, Spanish%3a Castilian, Italian, English%3a UK, Swedish, Danish,
Norwegian%3a Bokmal, Portuguese, Portuguese%3a Brazilian, French%3a Canadian,
Norwegian%3a Nynorsk, Finnish, Catalan, Russian, Bulgarian, Czech, Polish,
Romanian, Greek, Turkish, Hungarian, English%3a Canadian, Slovak, Croatian,
Estonian, Latvian, Lithuanian, Slovenian, German%3a Traditional, German%3a
Reformed, de_DE_2006, Dutch, nl_NL_2005, German%3a Swiss, de_CH_2006, Ukrainian
```

## 1.1.3 Colors

The default colors are:

```
Black, Cyan, Magenta, Paper, Registration, Yellow, u7d, u7f
```

#### **1.1.4 Inks**

The default inks are:

Process Cyan, Process Magenta, Process Yellow, and Process Black.

#### **1.1.5 Swatches**

The only default swatch is None.

#### **1.1.6 Gradients**

The only default gradient is named "`u7e`", which includes the default colors "`u7f`" and `Black`.

#### **1.1.7 Stroke Styles**

The default stroke styles are:

`Triple_Stroke`, `ThickThinThick`, `ThinThickThin`, `ThickThick`, `ThickThin`, `ThinThick`, `ThinThin`, `Japanese Dots`, `White Diamond`, `Left Slant Hash`, `Right Slant Hash`, `Straight Hash`, `Wavy`, `Canned Dotted`, `Canned Dashed 3x2`, `Dashed`, and `Solid`

#### **1.1.8 Font Families**

Each font family contains a set of fonts. In the following listing, the indented listings are the names of the fonts within a given font family.

```

Times New Roman
Times New Roman Regular
Times New Roman Italic
Times New Roman Bold
Times New Roman Bold Italic
Myriad Pro
    Myriad Pro Condensed
    Myriad Pro Condensed Italic
    Myriad Pro Bold Condensed
    Myriad Pro Bold Condensed Italic
    Myriad Pro Regular
    Myriad Pro Italic
    Myriad Pro Semibold
    Myriad Pro Semibold Italic
    Myriad Pro Bold
    Myriad Pro Bold Italic
Minion Pro
    Minion Pro Bold Cond
    Minion Pro Bold Cond Italic
    Minion Pro Regular
    Minion Pro Italic
    Minion Pro Medium
    Minion Pro Medium Italic
    Minion Pro Semibold
    Minion Pro Semibold Italic
    Minion Pro Bold

```

Minion Pro Bold Italic  
 Kozuka Mincho Pro  
 Kozuka Mincho Pro EL  
 Kozuka Mincho Pro L  
 Kozuka Mincho Pro R  
 Kozuka Mincho Pro M  
 Kozuka Mincho Pro B  
 Kozuka Mincho Pro H

### 1.1.9 Character Styles

The defaults file contains a `<RootCharacterStyle>` element, which, in turn, contains a `<CharacterStyle>` element named [No character style].

#### **[No character style]**

---

**Table 1. Character Style Properties Represented as Attributes**

Attribute Name	Value
Imported	"false"
Name	"\$ID/[No character style]"

### 1.1.10 Paragraph Styles

The defaults file contains a `<RootParagraphStyle>` element, which, in turn, contains the following `<ParagraphStyle>` elements: [No paragraph style], Normal Paragraph Style.

#### **[No paragraph style]**

---

**Table 2. Paragraph Style Properties Represented as Attributes**

Attribute Name	Value
Imported	"false"
FillColor	"Color/Black"
FontStyle	"Regular"
PointSize	"12"
HorizontalScale	"100"
KerningMethod	"\$ID/Metrics"
Ligatures	"true"
PageNumberType	"AutoPageNumber"
StrokeWeight	"1"
Tracking	"0"
Composer	"HL Composer"
DropCapCharacters	"0"

Attribute Name	Value
DropCapLines	"0"
BaselineShift	"0"
Capitalization	"Normal"
StrokeColor	"Swatch/None"
HyphenateLadderLimit	"3"
VerticalScale	"100"
LeftIndent	"0"
RightIndent	"0"
FirstLineIndent	"0"
AutoLeading	"120"
AppliedLanguage	"\$ID/English: USA"
Hyphenation	"true"
HyphenateAfterFirst	"2"
HyphenateBeforeLast	"2"
HyphenateCapitalizedWords	"true"
HyphenateWordsLongerThan	"5"
NoBreak	"false"
HyphenationZone	"36"
SpaceBefore	"0"
SpaceAfter	"0"
Underline	"false"
OTFFigureStyle	"Default"
DesiredWordSpacing	"100"
MaximumWordSpacing	"133"
MinimumWordSpacing	"80"
DesiredLetterSpacing	"0"
MaximumLetterSpacing	"0"
MinimumLetterSpacing	"0"
DesiredGlyphScaling	"100"
MaximumGlyphScaling	"100"
MinimumGlyphScaling	"100"
StartParagraph	"Anywhere"
KeepAllLinesTogether	"false"
KeepWithNext	"0"
KeepFirstLines	"2"
KeepLastLines	"2"
Position	"Normal"
StrikeThru	"false"
CharacterAlignment	"AlignEmCenter"

Attribute Name	Value
KeepLinesTogether	"false"
StrokeTint	"-1"
FillTint	"-1"
OverprintStroke	"false"
OverprintFill	"false"
GradientStrokeAngle	"0"
GradientFillAngle	"0"
GradientStrokeLength	"-1"
GradientFillLength	"-1"
GradientStrokeStart	"0 0"
GradientFillStart	"0 0"
Skew	"0"
RuleAboveLineWeight	"1"
RuleAboveTint	"-1"
RuleAboveOffset	"0"
RuleAboveLeftIndent	"0"
RuleAboveRightIndent	"0"
RuleAboveWidth	"ColumnWidth"
RuleBelowLineWeight	"1"
RuleBelowTint	"-1"
RuleBelowOffset	"0"
RuleBelowLeftIndent	"0"
RuleBelowRightIndent	"0"
RuleBelowWidth	"ColumnWidth"
RuleAboveOverprint	"false"
RuleBelowOverprint	"false"
RuleAbove	"false"
RuleBelow	"false"
LastLineIndent	"0"
HyphenateLastWord	"true"
ParagraphBreakType	"Anywhere"
SingleWordJustification	"FullyJustified"
OTFOrdinal	"false"
OTFFraction	"false"
OTFDiscretionaryLigature	"false"
OTFTitling	"false"
RuleAboveGapTint	"-1"
RuleAboveGapOverprint	"false"
RuleBelowGapTint	"-1"

Attribute Name	Value
RuleBelowGapOverprint	"false"
Justification	"LeftAlign"
DropcapDetail	"0"
PositionalForm	"None"
OTFMark	"true"
HyphenWeight	"5"
OTFLocale	"true"
HyphenateAcrossColumns	"true"
KeepRuleAboveInFrame	"false"
IgnoreEdgeAlignment	"false"
OTFSashedZero	"false"
OTFStylisticSets	"0"
OTFHistorical	"false"
OTFContextualAlternate	"true"
UnderlineGapOverprint	"false"
UnderlineGapTint	"-1"
UnderlineOffset	"-9999"
UnderlineOverprint	"false"
UnderlineTint	"-1"
UnderlineWeight	"-9999"
StrikeThroughGapOverprint	"false"
StrikeThroughGapTint	"-1"
StrikeThroughOffset	"-9999"
StrikeThroughOverprint	"false"
StrikeThroughTint	"-1"
StrikeThroughWeight	"-9999"
MiterLimit	"4"
StrokeAlignment	"OutsideAlignment"
EndJoin	"MiterEndJoin"
OTFSwash	"false"
Tsume	"0"
LeadingAki	"-1"
TrailingAki	"-1"
KinsokuType	"KinsokuPushInFirst"
KinsokuHangType	"None"
BunriKinshi	"true"
RubyOpenTypePro	"true"
RubyFontSize	"-1"
RubyAlignment	"RubyJIS"

Attribute Name	Value
RubyType	"PerCharacterRuby"
RubyParentSpacing	"RubyParent121Aki"
RubyXScale	"100"
RubyYScale	"100"
RubyXOffset	"0"
RubyYOffset	"0"
RubyPosition	"AboveRight"
RubyAutoAlign	"true"
RubyParentOverhangAmount	"RubyOverhangOneRuby"
RubyOverhang	"false"
RubyAutoScaling	"false"
RubyParentScalingPercent	"66"
RubyTint	"-1"
RubyOverprintFill	"Auto"
RubyStrokeTint	"-1"
RubyOverprintStroke	"Auto"
RubyWeight	"-1"
KntenKind	"None"
KntenFontSize	"-1"
KntenXScale	"100"
KntenYScale	"100"
KntenPlacement	"0"
KntenAlignment	"AlignKntenCenter"
KntenPosition	"AboveRight"
KntenCustomCharacter	" "
KntenCharacterSet	"CharacterInput"
KntenTint	"-1"
KntenOverprintFill	"Auto"
KntenStrokeTint	"-1"
KntenOverprintStroke	"Auto"
KntenWeight	"-1"
Tatechuyoko	"false"
TatechuyokoXOffset	"0"
TatechuyokoYOffset	"0"
AutoTcy	"0"
AutoTcyIncludeRoman	"false"
Jidori	"0"
GridGyoudori	"0"
GridAlignFirstLineOnly	"false"

Attribute Name	Value
GridAlignment	"None"
CharacterRotation	"0"
RotateSingleByteCharacters	"false"
Rensuuji	"true"
ShataiMagnification	"0"
ShataiDegreeAngle	"4500"
ShataiAdjustTsume	"true"
ShataiAdjustRotation	"false"
Warichu	"false"
WarichuLines	"2"
WarichuSize	"50"
WarichuLineSpacing	"0"
WarichuAlignment	"Auto"
WarichuCharsBeforeBreak	"2"
WarichuCharsAfterBreak	"2"
OTFHVKana	"false"
OTFProportionalMetrics	"false"
OTFRomanItalics	"false"
LeadingModel	"LeadingModelAkiBelow"
ScaleAffectsLineHeight	"false"
ParagraphGyoudori	"false"
CjkGridTracking	"false"
GlyphForm	"None"
RubyAutoTcyDigits	"0"
RubyAutoTcyIncludeRoman	"false"
RubyAutoTcyAutoScale	"true"
TreatIdeographicSpaceAsSpace	"false"
AllowArbitraryHyphenation	"false"
BulletsAndNumberingListType	"NoList"
NumberingStartAt	"1"
NumberingLevel	"1"
NumberingContinue	"true"
NumberingApplyRestartPolicy	"true"
BulletsAlignment	"LeftAlign"
NumberingAlignment	"LeftAlign"
NumberingExpression	"^#.^t"
BulletsTextAfter	"^t"
DigitsType	"DefaultDigits"
Kashidas	"DefaultKashidas"

Attribute Name	Value
DiacriticPosition	"OpentypePosition"
CharacterDirection	"DefaultDirection"
ParagraphDirection	"LeftToRightDirection"
ParagraphJustification	"DefaultJustification"
XOffsetDiacritic	"0"
YOffsetDiacritic	"0"
OTFOverlapSwash	"false"
OTFStylisticAlternate	"false"
OTFJustificationAlternate	"false"
OTFStretchedAlternate	"false"
KeyboardDirection	"DefaultDirection">

**Table 3. Paragraph Properties Represented as Elements**

Element Name	Value
Leading	Auto
AppliedFont	Times New Roman
RuleAboveColor	Text Color
RuleBelowColor	Text Color
RuleAboveType	StrokeStyle/\$ID/Solid
RuleBelowType	StrokeStyle/\$ID/Solid
BalanceRaggedLines	NoBalancing
RuleAboveGapColor	Swatch/None
RuleBelowGapColor	Swatch/None
UnderlineColor	Text Color
UnderlineGapColor	Swatch/None
UnderlineType	StrokeStyle/\$ID/Solid
StrikeThroughColor	Text Color
StrikeThroughGapColor	Swatch/None
StrikeThroughType	StrokeStyle/\$ID/Solid
Mojikumi	Nothing
KinsokuSet	Nothing
RubyFont	\$ID/
RubyFontStyle	Nothing
RubyFill	Text Color
RubyStroke	Text Color
KentenFont	\$ID/
KentenFontStyle	Nothing
KentenFillColor	Text Color
KentenStrokeColor	Text Color

Element Name	Value
BulletChar	BulletCharacterType="UnicodeOnly" BulletCharacterValue="8226"
NumberingFormat	1, 2, 3, 4...
BulletsFont	\$ID/
BulletsFontSize	Nothing
AppliedNumberingList	NumberingList/\$ID/[Default]
NumberingRestartPolicies	RestartPolicy="AnyPreviousLevel" LowerLevel="0" UpperLevel="0"
BulletsCharacterStyle	CharacterStyle/\$ID/[No character style]
NumberingCharacterStyle	CharacterStyle/\$ID/[No character style]

### **NormalParagraphStyle**

**Table 4. Paragraph Properties Represented as Attributes**

Attribute Name	Value
Imported	"false"
NextStyle	"ParagraphStyle/\$ID/ NormalParagraphStyle"
KeyboardShortcut	"0 0"

**Table 5. Paragraph Properties Represented as Elements**

Element Name	Value
BasedOn	\$ID/[No paragraph style]
PreviewColor	Nothing

#### **1.1.11 TOC Styles**

The defaults file contains a single TOC style.

**Table 6. TOC Style Properties Represented as Attributes**

Attribute Name	Value
TitleStyle	"ParagraphStyle/\$ID/[No paragraph style]"
Title	"Contents"
Name	"\$ID/DefaultTOCStyleName"
RunIn	"false"
IncludeHidden	"false"
IncludeBookDocuments	"false"

Attribute Name	Value
CreateBookmarks	"true"
SetStoryDirection	"Horizontal"
NumberedParagraphs	"IncludeFullParagraph"

### 1.1.12 Cell Styles

The defaults file contains a single cell style, [None].

---

**Table 7. Cell Style Properties Represented as Attributes**

Attribute Name	Value
AppliedParagraphStyle	"ParagraphStyle/\$ID/ [No paragraph style]"
Name	"\$ID/ [None]"

### 1.1.13 Table Styles

The defaults file contains two table styles, [No table style] and [Basic Table].

#### **[No table style]**

---

**Table 8. Table Style Properties Represented as Attributes**

Attribute Name	Value
Name	"\$ID/ [No table style]"
StrokeOrder	"BestJoins"
TopBorderStrokeWeight	"1"
TopBorderStrokeType	"StrokeStyle/\$ID/Solid"
TopBorderStrokeColor	"Color/Black"
TopBorderStrokeTint	"100"
TopBorderStrokeOverprint	"false"
TopBorderStrokeGapColor	"Color/Paper"
TopBorderStrokeGapTint	"100"
TopBorderStrokeGapOverprint	"false"
LeftBorderStrokeWeight	"1"
LeftBorderStrokeType	"StrokeStyle/\$ID/Solid"
LeftBorderStrokeColor	"Color/Black"
LeftBorderStrokeTint	"100"
LeftBorderStrokeOverprint	"false"
LeftBorderStrokeGapColor	"Color/Paper"

Attribute Name	Value
LeftBorderStrokeGapTint	"100"
LeftBorderStrokeGapOverprint	"false"
BottomBorderStrokeWeight	"1"
BottomBorderStrokeType	"StrokeStyle/\$ID/Solid"
BottomBorderStrokeColor	"Color/Black"
BottomBorderStrokeTint	"100"
BottomBorderStrokeOverprint	"false"
BottomBorderStrokeGapColor	"Color/Paper"
BottomBorderStrokeGapTint	"100"
BottomBorderStrokeGapOverprint	"false"
RightBorderStrokeWeight	"1"
RightBorderStrokeType	"StrokeStyle/\$ID/Solid"
RightBorderStrokeColor	"Color/Black"
RightBorderStrokeTint	"100"
RightBorderStrokeOverprint	"false"
RightBorderStrokeGapColor	"Color/Paper"
RightBorderStrokeGapTint	"100"
RightBorderStrokeGapOverprint	"false"
SpaceBefore	"4"
SpaceAfter	"-4"
SkipFirstAlternatingStrokeRows	"0"
SkipLastAlternatingStrokeRows	"0"
StartRowStrokeCount	"0"
StartRowStrokeColor	"Color/Black"
StartRowStrokeWeight	"1"
StartRowStrokeType	"StrokeStyle/\$ID/Solid"
StartRowStrokeTint	"100"
StartRowStrokeGapOverprint	"false"
StartRowStrokeGapColor	"Color/Paper"
StartRowStrokeGapTint	"100"
StartRowStrokeOverprint	"false"
EndRowStrokeCount	"0"
EndRowStrokeColor	"Color/Black"
EndRowStrokeWeight	"0.25"
EndRowStrokeType	"StrokeStyle/\$ID/Solid"
EndRowStrokeTint	"100"
EndRowStrokeOverprint	"false"
EndRowStrokeGapColor	"Color/Paper"
EndRowStrokeGapTint	"100"

Attribute Name	Value
EndRowStrokeGapOverprint	"false"
SkipFirstAlternatingStrokeColumns	"0"
SkipLastAlternatingStrokeColumns	"0"
StartColumnStrokeCount	"0"
StartColumnStrokeColor	"Color/Black"
StartColumnStrokeWeight	"1"
StartColumnStrokeType	"StrokeStyle/\$ID/Solid"
StartColumnStrokeTint	"100"
StartColumnStrokeOverprint	"false"
StartColumnStrokeGapColor	"Color/Paper"
StartColumnStrokeGapTint	"100"
StartColumnStrokeGapOverprint	"false"
EndColumnStrokeCount	"0"
EndColumnStrokeColor	"Color/Black"
EndColumnStrokeWeight	"0.25"
EndColumnLineStyle	"StrokeStyle/\$ID/Solid"
EndColumnStrokeTint	"100"
EndColumnStrokeOverprint	"false"
EndColumnStrokeGapColor	"Color/Paper"
EndColumnStrokeGapTint	"100"
EndColumnStrokeGapOverprint	"false"
ColumnFillsPriority	"false"
SkipFirstAlternatingFillRows	"0"
SkipLastAlternatingFillRows	"0"
StartRowFillColor	"Color/Black"
StartRowFillCount	"0"
StartRowFillTint	"20"
StartRowFillOverprint	"false"
EndRowFillCount	"0"
EndRowFillColor	"Swatch/None"
EndRowFillTint	"100"
EndRowFillOverprint	"false"
SkipFirstAlternatingFillColumns	"0"
SkipLastAlternatingFillColumns	"0"
StartColumnFillCount	"0"
StartColumnFillColor	"Color/Black"
StartColumnFillTint	"20"
StartColumnFillOverprint	"false"
EndColumnFillCount	"0"

Attribute Name	Value
EndColumnFillColor	"Swatch/None"
EndColumnFillTint	"100"
EndColumnFillOverprint	"false"
HeaderRegionSameAsBodyRegion	"true"
FooterRegionSameAsBodyRegion	"true"
LeftColumnRegionSameAsBodyRegion	"true"
RightColumnRegionSameAsBodyRegion	"true"
HeaderRegionCellStyle	"n"
FooterRegionCellStyle	"n"
LeftColumnRegionCellStyle	"n"
RightColumnRegionCellStyle	"n"
BodyRegionCellStyle	"CellStyle/\$ID/[None]"/>

### [Basic Table]

**Table 9. Table Style Properties Represented as Attributes**

Attribute Name	Value
KeyboardShortcut	"0 0"
Name	"\$ID/[Basic Table]"

**Table 10. Table Style Properties Represented as Elements**

Element Name	Value
BasedOn	\$ID/[No table style]

### 1.1.14 Named Grids

The defaults file contains a single <NamedGrid> element, [Page Grid]. This element contains a <GridDataInformation> element, which, in turn, contains an <AppliedFont> element.

**Table 11. Table Style Properties Represented as Elements**

Attribute Name	Value
Name	"\$ID/[Page Grid]"

**Table 12. Grid Data Properties Represented as Attributes**

Attribute Name	Value
FontStyle	"Regular"
PointSize	"12"

Attribute Name	Value
CharacterAki	"0"
LineAki	"9"
HorizontalScale	"100"
VerticalScale	"100"
LineAlignment	"LeftOrTopLineJustify"
GridAlignment	"AlignEmCenter"
CharacterAlignment	"AlignEmCenter"

**Table 13. Grid Data Properties Represented as Elements**

Element Name	Value
AppliedFont	Times New Roman

### 1.1.15 Object Styles

The defaults file contains a series of <ObjectStyle> elements: [None], [Normal Graphics Frame], [Normal Text Frame], and [Normal Grid].

Each <ObjectStyle> element contains the following elements (some of these elements contain child elements; these child elements are indicated by an indent).

```
<TextFramePreference>
    <InsetSpacing>
<BaselineFrameGridOption>
    <BaselineFrameGridColor>
<AnchoredObjectSetting>
<TextWrapPreference>
    <TextWrapOffset>
    <ContourOption>
<StoryPreference>
<FrameFittingOption>
```

Some of the <ObjectStyle> elements also contain the following elements:

```
<ObjectStyleObjectEffectsCategorySettings>
<ObjectStyleStrokeEffectsCategorySettings>
<ObjectStyleFillEffectsCategorySettings>
<ObjectStyleContentEffectsCategorySettings>
```

**[None]**

**Table 14. Object Style Properties Represented as Attributes**

Attribute Name	Value
AppliedParagraphStyle	"ParagraphStyle/\$ID/ [No paragraph style]"

Attribute Name	Value
FillColor	"Swatch/None"
FillTint	"-1"
StrokeWeight	"0"
MiterLimit	"4"
EndCap	"ButtEndCap"
EndJoin	"MiterEndJoin"
StrokeType	"StrokeStyle/\$ID/Solid"
LeftLineEnd	"None"
RightLineEnd	"None"
StrokeColor	"Swatch/None"
StrokeTint	"-1"
CornerRadius	"12"
GapColor	"Swatch/None"
GapTint	"-1"
StrokeAlignment	"CenterAlignment"
Nonprinting	"false"
GradientFillAngle	"0"
GradientStrokeAngle	"0"
AppliedNamedGrid	"n"
CornerOption	"None"

**Table 15. Text Frame Preferences Properties Represented as Attributes**

Attribute Name	Value
TextColumnCount	"1"
TextColumnGutter	"12"
TextColumnFixedWidth	"144"
UseFixedColumnWidth	"false"
FirstBaselineOffset	"AscentOffset"
MinimumFirstBaselineOffset	"0"
VerticalJustification	"TopAlign"
VerticalThreshold	"0"
IgnoreWrap	"false"

**Table 16. Text Frame Preferences Properties Represented as Elements**

Element Name	Value
InsetSpacing	<List Item type="unit">0</List Item> <List Item type="unit">0</List Item> <List Item type="unit">0</List Item> <List Item type="unit">0</List Item>

**Table 17. Baseline Frame Grid Options Represented as Attributes**

Attribute Name	Value
UseCustomBaselineFrameGrid	"false"
StartingOffsetForBaselineFrameGrid	"0"
BaselineFrameGridRelativeOption	"TopOfInset"
BaselineFrameGridIncrement	"12"

**Table 18. Baseline Frame Grid Options Represented as Elements**

Element Name	Value
BaselineFrameGridColor	LightBlue

**Table 19. Anchored Object Settings Properties Represented as Attributes**

Attribute Name	Value
AnchoredPosition	"InlinePosition"
SpineRelative	"false"
LockPosition	"false"
PinPosition	"true"
AnchorPoint	"BottomRightAnchor"
HorizontalAlignment	"LeftAlign"
HorizontalReferencePoint	"TextFrame"
VerticalAlignment	"BottomAlign"
VerticalReferencePoint	"LineBaseline"
AnchorXoffset	"0"
AnchorYoffset	"0"
AnchorSpaceAbove	"0"

**Table 20. Text Wrap Preferences Properties Represented as Attributes**

Attribute Name	Value
Inverse	"false"
ApplyToMasterPageOnly	"false"
TextWrapSide	"BothSides"
TextWrapMode	"None"

**Table 21. Text Wrap Preferences Properties Represented as Elements**

Element Name	Value
TextWrapOffset	Top="0" Left="0" Bottom="0" Right="0"

**Table 22. Contour Option Properties Represented as Attributes**

Attribute Name	Value
ContourType	"SameAsClipping"
IncludeInsideEdges	"false"
ContourPathName	"\$ID/"

**Table 23. Story Preferences Properties Represented as Attributes**

Attribute Name	Value
OpticalMarginAlignment	"false"
OpticalMarginSize	"12"
FrameType	"TextFrameType"
StoryOrientation	"Horizontal"
StoryDirection	"UnknownDirection"

**Table 24. Frame Fitting Option Properties Represented as Attributes**

Attribute Name	Value
LeftCrop	"0"
TopCrop	"0"
RightCrop	"0"
BottomCrop	"0"
FittingOnEmptyFrame	"None"
FittingAlignment	"TopLeftAnchor"

**[Normal Graphics Frame]****Table 25. Object Style Properties Represented as Attributes**

Attribute Name	Value
Name	"\$ID/ [Normal Graphics Frame]"
AppliedParagraphStyle	"ParagraphStyle/\$ID/ [No paragraph style]"
ApplyNextParagraphStyle	"false"
EnableFill	"true"
EnableStroke	"true"
EnableParagraphStyle	"false"
EnableTextFrameGeneralOptions	"false"
EnableTextFrameBaselineOptions	"false"
EnableStoryOptions	"false"
EnableTextWrapAndOthers	"false"
EnableAnchoredObjectOptions	"false"

Attribute Name	Value
FillColor	"Swatch/None"
FillTint	"-1"
StrokeWeight	"1"
MiterLimit	"4"
EndCap	"ButtEndCap"
EndJoin	"MiterEndJoin"
StrokeType	"StrokeStyle/\$ID/Solid"
LeftLineEnd	"None"
RightLineEnd	"None"
StrokeColor	"Color/Black"
StrokeTint	"-1"
CornerRadius	"12"
OverprintStroke	"false"
GapColor	"Swatch/None"
GapTint	"-1"
StrokeAlignment	"CenterAlignment"
Nonprinting	"false"
GradientFillAngle	"0"
GradientStrokeAngle	"0"
AppliedNamedGrid	"n"
KeyboardShortcut	"0 0"
EnableFrameFittingOptions	"false"
CornerOption	"None"
EnableStrokeAndCornerOptions	"true"

**Table 26. Object Style Properties Represented as Elements**

Element Name	Value
BasedOn	\$ID/ [None]

**Table 27. Text Frame Preferences Properties Represented as Attributes**

Attribute Name	Value
TextColumnCount	"1"
TextColumnGutter	"12"
TextColumnFixedWidth	"144"
UseFixedColumnWidth	"false"
FirstBaselineOffset	"AscentOffset"
MinimumFirstBaselineOffset	"0"
VerticalJustification	"TopAlign"

Attribute Name	Value
VerticalThreshold	"0"
IgnoreWrap	"false"

**Table 28. Text Frame Preferences Properties Represented as Elements**

Element Name	Value
InsetSpacing	<List Item type="unit">0</List Item> <List Item type="unit">0</List Item> <List Item type="unit">0</List Item> <List Item type="unit">0</List Item>

**Table 29. Baseline Frame Grid Options Represented as Attributes**

Attribute Name	Value
UseCustomBaselineFrameGrid	"false"
StartingOffsetForBaselineFrameGrid	"0"
BaselineFrameGridRelativeOption	"TopOfInset"
BaselineFrameGridIncrement	"12"

**Table 30. Baseline Frame Grid Options Represented as Elements**

Element Name	Value
BaselineFrameGridColor	LightBlue

**Table 31. Anchored Object Settings Properties Represented as Attributes**

Attribute Name	Value
AnchoredPosition	"InlinePosition"
SpineRelative	"false"
LockPosition	"false"
PinPosition	"true"
AnchorPoint	"BottomRightAnchor"
HorizontalAlignment	"LeftAlign"
HorizontalReferencePoint	"TextFrame"
VerticalAlignment	"BottomAlign"
VerticalReferencePoint	"LineBaseline"
AnchorXoffset	"0"
AnchorYoffset	"0"
AnchorSpaceAbove	"0"

**Table 32. Text Wrap Preferences Properties Represented as Attributes**

Attribute Name	Value
Inverse	"false"
ApplyToMasterPageOnly	"false"
TextWrapSide	"BothSides"
TextWrapMode	"None"

**Table 33. Text Wrap Preferences Properties Represented as Elements**

Element Name	Value
TextWrapOffset	Top="0" Left="0" Bottom="0" Right="0"

**Table 34. Contour Option Properties Represented as Attributes**

Attribute Name	Value
ContourType	"SameAsClipping"
IncludeInsideEdges	"false"
ContourPathName	"\$ID/"

**Table 35. Story Preferences Properties Represented as Attributes**

Attribute Name	Value
OpticalMarginAlignment	"false"
OpticalMarginSize	"12"
FrameType	"TextFrameType"
StoryOrientation	"Horizontal"
StoryDirection	"UnknownDirection"

**Table 36. Frame Fitting Option Properties Represented as Attributes**

Attribute Name	Value
LeftCrop	"0"
TopCrop	"0"
RightCrop	"0"
BottomCrop	"0"
FittingOnEmptyFrame	"None"
FittingAlignment	"TopLeftAnchor"

**Table 37. Object Style Object Effects Category Settings Properties Represented as Attributes**

Attribute Name	Value
EnableTransparency	"true"

Attribute Name	Value
EnableDropShadow	"true"
EnableFeather	"true"
EnableInnerShadow	"true"
EnableOuterGlow	"true"
EnableInnerGlow	"true"
EnableBevelEmboss	"true"
EnableSatin	"true"
EnableDirectionalFeather	"true"
EnableGradientFeather	"true"

**Table 38. Object Style Stroke Effects Category Settings Properties Represented as Attributes**

Attribute Name	Value
EnableTransparency	"true"
EnableDropShadow	"true"
EnableFeather	"true"
EnableInnerShadow	"true"
EnableOuterGlow	"true"
EnableInnerGlow	"true"
EnableBevelEmboss	"true"
EnableSatin	"true"
EnableDirectionalFeather	"true"
EnableGradientFeather	"true"

**Table 39. Object Style Fill Effects Category Settings Properties Represented as Attributes**

Attribute Name	Value
EnableTransparency	"true"
EnableDropShadow	"true"
EnableFeather	"true"
EnableInnerShadow	"true"
EnableOuterGlow	"true"
EnableInnerGlow	"true"
EnableBevelEmboss	"true"
EnableSatin	"true"
EnableDirectionalFeather	"true"
EnableGradientFeather	"true"

**Table 40. Object Style Content Effects Category Settings Properties Represented as Attributes**

Attribute Name	Value
EnableTransparency	"true"
EnableDropShadow	"true"
EnableFeather	"true"
EnableInnerShadow	"true"
EnableOuterGlow	"true"
EnableInnerGlow	"true"
EnableBevelEmboss	"true"
EnableSatin	"true"
EnableDirectionalFeather	"true"
EnableGradientFeather	"true"

**[Normal Text Frame]****Table 41. Object Style Properties Represented as Attributes**

Attribute Name	Value
Name	"\$ID/ [Normal Text Frame]"
AppliedParagraphStyle	"ParagraphStyle/\$ID/ NormalParagraphStyle"
ApplyNextParagraphStyle	"false"
EnableFill	"true"
EnableStroke	"true"
EnableParagraphStyle	"false"
EnableTextFrameGeneralOptions	"true"
EnableTextFrameBaselineOptions	"true"
EnableStoryOptions	"false"
EnableTextWrapAndOthers	"false"
EnableAnchoredObjectOptions	"false"
FillColor	"Swatch/None"
FillTint	"-1"
StrokeWeight	"0"
MiterLimit	"4"
EndCap	"ButtEndCap"
EndJoin	"MiterEndJoin"
StrokeType	"StrokeStyle/\$ID/Solid"
LeftLineEnd	"None"
RightLineEnd	"None"
StrokeColor	"Swatch/None"

Attribute Name	Value
StrokeTint	"-1"
CornerRadius	"12"
GapColor	"Swatch/None"
GapTint	"-1"
StrokeAlignment	"CenterAlignment"
Nonprinting	"false"
GradientFillAngle	"0"
GradientStrokeAngle	"0"
AppliedNamedGrid	"n"
KeyboardShortcut	"0 0"
EnableFrameFittingOptions	"false"
CornerOption	"None"
EnableStrokeAndCornerOptions	"true"

**Table 42. Object Style Properties Represented as Elements**

Element Name	Value
BasedOn	\$ID/ [None]

**Table 43. Text Frame Preferences Properties Represented as Attributes**

Attribute Name	Value
TextColumnCount	"1"
TextColumnGutter	"12"
TextColumnFixedWidth	"144"
UseFixedColumnWidth	"false"
FirstBaselineOffset	"AscentOffset"
MinimumFirstBaselineOffset	"0"
VerticalJustification	"TopAlign"
VerticalThreshold	"0"
IgnoreWrap	"false"

**Table 44. Text Frame Preferences Properties Represented as Elements**

Element Name	Value
InsetSpacing	<List Item type="unit">0</List Item> <List Item type="unit">0</List Item> <List Item type="unit">0</List Item> <List Item type="unit">0</List Item>

**Table 45. Baseline Frame Grid Options Represented as Attributes**

Attribute Name	Value
UseCustomBaselineFrameGrid	"false"
StartingOffsetForBaselineFrameGrid	"0"
BaselineFrameGridRelativeOption	"TopOfInset"
BaselineFrameGridIncrement	"12"

**Table 46. Baseline Frame Grid Options Represented as Elements**

Element Name	Value
BaselineFrameGridColor	LightBlue

**Table 47. Anchored Object Settings Properties Represented as Attributes**

Attribute Name	Value
AnchoredPosition	"InlinePosition"
SpineRelative	"false"
LockPosition	"false"
PinPosition	"true"
AnchorPoint	"BottomRightAnchor"
HorizontalAlignment	"LeftAlign"
HorizontalReferencePoint	"TextFrame"
VerticalAlignment	"BottomAlign"
VerticalReferencePoint	"LineBaseline"
AnchorXoffset	"0"
AnchorYoffset	"0"
AnchorSpaceAbove	"0"

**Table 48. Text Wrap Preferences Properties Represented as Attributes**

Attribute Name	Value
Inverse	"false"
ApplyToMasterPageOnly	"false"
TextWrapSide	"BothSides"
TextWrapMode	"None"

**Table 49. Text Wrap Preferences Properties Represented as Elements**

Element Name	Value
TextWrapOffset	Top="0" Left="0" Bottom="0" Right="0"

**Table 50. Contour Option Properties Represented as Attributes**

Attribute Name	Value
ContourType	"SameAsClipping"
IncludeInsideEdges	"false"
ContourPathName	"\$ID/"

**Table 51. Story Preferences Properties Represented as Attributes**

Attribute Name	Value
OpticalMarginAlignment	"false"
OpticalMarginSize	"12"
FrameType	"TextFrameType"
StoryOrientation	"Horizontal"
StoryDirection	"UnknownDirection"

**Table 52. Frame Fitting Option Properties Represented as Attributes**

Attribute Name	Value
LeftCrop	"0"
TopCrop	"0"
RightCrop	"0"
BottomCrop	"0"
FittingOnEmptyFrame	"None"
FittingAlignment	"TopLeftAnchor"

**Table 53. Object Style Object Effects Category Settings Properties Represented as Attributes**

Attribute Name	Value
EnableTransparency	"true"
EnableDropShadow	"true"
EnableFeather	"true"
EnableInnerShadow	"true"
EnableOuterGlow	"true"
EnableInnerGlow	"true"
EnableBevelEmboss	"true"
EnableSatin	"true"
EnableDirectionalFeather	"true"
EnableGradientFeather	"true"

**Table 54. Object Style Stroke Effects Category Settings Properties Represented as Attributes**

Attribute Name	Value
EnableTransparency	"true"
EnableDropShadow	"true"
EnableFeather	"true"
EnableInnerShadow	"true"
EnableOuterGlow	"true"
EnableInnerGlow	"true"
EnableBevelEmboss	"true"
EnableSatin	"true"
EnableDirectionalFeather	"true"
EnableGradientFeather	"true"

**Table 55. Object Style Fill Effects Category Settings Properties Represented as Attributes**

Attribute Name	Value
EnableTransparency	"true"
EnableDropShadow	"true"
EnableFeather	"true"
EnableInnerShadow	"true"
EnableOuterGlow	"true"
EnableInnerGlow	"true"
EnableBevelEmboss	"true"
EnableSatin	"true"
EnableDirectionalFeather	"true"
EnableGradientFeather	"true"

**Table 56. Object Style Content Effects Category Settings Properties Represented as Attributes**

Attribute Name	Value
EnableTransparency	"true"
EnableDropShadow	"true"
EnableFeather	"true"
EnableInnerShadow	"true"
EnableOuterGlow	"true"
EnableInnerGlow	"true"
EnableBevelEmboss	"true"
EnableSatin	"true"
EnableDirectionalFeather	"true"
EnableGradientFeather	"true"

**[Normal Grid]**

**Table 57. Object Style Properties Represented as Attributes**

Attribute Name	Value
Name	"\$ID/ [Normal Grid]"
AppliedParagraphStyle	"ParagraphStyle/\$ID/ NormalParagraphStyle"
ApplyNextParagraphStyle	"false"
EnableFill	"true"
EnableStroke	"true"
EnableParagraphStyle	"false"
EnableTextFrameGeneralOptions	"true"
EnableTextFrameBaselineOptions	"true"
EnableStoryOptions	"true"
EnableTextWrapAndOthers	"false"
EnableAnchoredObjectOptions	"false"
FillColor	"Swatch/None"
FillTint	"-1"
StrokeWeight	"0"
MiterLimit	"4"
EndCap	"ButtEndCap"
EndJoin	"MiterEndJoin"
StrokeType	"StrokeStyle/\$ID/Solid"
LeftLineEnd	"None"
RightLineEnd	"None"
StrokeColor	"Swatch/None"
StrokeTint	"-1"
CornerRadius	"12"
GapColor	"Swatch/None"
GapTint	"-1"
StrokeAlignment	"CenterAlignment"
Nonprinting	"false"
GradientFillAngle	"0"
GradientStrokeAngle	"0"
AppliedNamedGrid	"n"
KeyboardShortcut	"0 0"
EnableFrameFittingOptions	"false"
CornerOption	"None"
EnableStrokeAndCornerOptions	"true"

**Table 58. Object Style Properties Represented as Elements**

Element Name	Value
BasedOn	\$ID/ [None]

**Table 59. Text Frame Preferences Properties Represented as Attributes**

Attribute Name	Value
TextColumnCount	"1"
TextColumnGutter	"12"
TextColumnFixedWidth	"144"
UseFixedColumnWidth	"false"
FirstBaselineOffset	"AscentOffset"
MinimumFirstBaselineOffset	"0"
VerticalJustification	"TopAlign"
VerticalThreshold	"0"
IgnoreWrap	"false"

**Table 60. Text Frame Preferences Properties Represented as Elements**

Element Name	Value
InsetSpacing	<List Item type="unit">0</List Item> <List Item type="unit">0</List Item> <List Item type="unit">0</List Item> <List Item type="unit">0</List Item>

**Table 61. Baseline Frame Grid Options Properties Represented as Attributes**

Attribute Name	Value
UseCustomBaselineFrameGrid	"false"
StartingOffsetForBaselineFrameGrid	"0"
BaselineFrameGridRelativeOption	"TopOfInset"
BaselineFrameGridIncrement	"12"

**Table 62. Baseline Frame Grid Options Properties Represented as Elements**

Element Name	Value
BaselineFrameGridColor	LightBlue

**Table 63. Anchored Object Settings Properties Represented as Attributes**

Attribute Name	Value
AnchoredPosition	"InlinePosition"
SpineRelative	"false"

Attribute Name	Value
LockPosition	"false"
PinPosition	"true"
AnchorPoint	"BottomRightAnchor"
HorizontalAlignment	"LeftAlign"
HorizontalReferencePoint	"TextFrame"
VerticalAlignment	"BottomAlign"
VerticalReferencePoint	"LineBaseline"
AnchorXoffset	"0"
AnchorYoffset	"0"
AnchorSpaceAbove	"0"

**Table 64. Text Wrap Preferences Properties Represented as Attributes**

Attribute Name	Value
Inverse	"false"
ApplyToMasterPageOnly	"false"
TextWrapSide	"BothSides"
TextWrapMode	"None"

**Table 65. Text Wrap Preferences Properties Represented as Elements**

Element Name	Value
TextWrapOffset	Top="0" Left="0" Bottom="0" Right="0"

**Table 66. Contour Option Properties Represented as Attributes**

Attribute Name	Value
ContourType	"SameAsClipping"
IncludeInsideEdges	"false"
ContourPathName	"\$ID/"

**Story Preferences Properties Represented as Attributes**

Attribute Name	Value
OpticalMarginAlignment	"false"
OpticalMarginSize	"12"
FrameType	"FrameGridType"
StoryOrientation	"Unknown"
StoryDirection	"UnknownDirection"

**Table 67. Frame Fitting Option Properties Represented as Attributes**

Attribute Name	Value
LeftCrop	"0"
TopCrop	"0"
RightCrop	"0"
BottomCrop	"0"
FittingOnEmptyFrame	"None"
FittingAlignment	"TopLeftAnchor"

**Table 68. Object Style Object Effects Category Settings Properties Represented as Attributes**

Attribute Name	Value
EnableTransparency	"true"
EnableDropShadow	"true"
EnableFeather	"true"
EnableInnerShadow	"true"
EnableOuterGlow	"true"
EnableInnerGlow	"true"
EnableBevelEmboss	"true"
EnableSatin	"true"
EnableDirectionalFeather	"true"
EnableGradientFeather	"true"

**Table 69. Object Style Stroke Effects Category Settings Properties Represented as Attributes**

Attribute Name	Value
EnableTransparency	"true"
EnableDropShadow	"true"
EnableFeather	"true"
EnableInnerShadow	"true"
EnableOuterGlow	"true"
EnableInnerGlow	"true"
EnableBevelEmboss	"true"
EnableSatin	"true"
EnableDirectionalFeather	"true"
EnableGradientFeather	"true"

**Table 70. Object Style Fill Effects Category Settings Properties Represented as Attributes**

Attribute Name	Value
EnableTransparency	"true"

Attribute Name	Value
EnableDropShadow	"true"
EnableFeather	"true"
EnableInnerShadow	"true"
EnableOuterGlow	"true"
EnableInnerGlow	"true"
EnableBevelEmboss	"true"
EnableSatin	"true"
EnableDirectionalFeather	"true"
EnableGradientFeather	"true"

**Table 71. Object Style Content Effects Category Settings Properties Represented as Attributes**

Attribute Name	Value
EnableTransparency	"true"
EnableDropShadow	"true"
EnableFeather	"true"
EnableInnerShadow	"true"
EnableOuterGlow	"true"
EnableInnerGlow	"true"
EnableBevelEmboss	"true"
EnableSatin	"true"
EnableDirectionalFeather	"true"
EnableGradientFeather	"true"

### 1.1.16 Trap Presets

The defaults file contains two `<TrapPreset>` elements: [No Trap Preset] and `$ID/kDefaultTrapStyleName`. These are the default trap presets for the document.

#### [No Trap Preset]

**Table 72. Trap Preset Properties Represented as Attributes**

Attribute Name	Value
Name	"\$ID/k[No Trap Preset]"
DefaultTrapWidth	"0.25"
BlackWidth	"0.5"
TrapJoin	"MiterEndJoin"
TrapEnd	"MiterTrapEnds"
ObjectsToImages	"true"

Attribute Name	Value
ImagesToImages	"true"
InternalImages	"false"
OneBitImages	"true"
ImagePlacement	"CenterEdges"
StepThreshold	"10"
BlackColorThreshold	"100"
BlackDensity	"1.6"
SlidingTrapThreshold	"70"
ColorReduction	"100"

### \$ID/kDefaultTrapStyleName

**Table 73. Trap Preset Properties Represented as Attributes**

Attribute Name	Value
Name	"\$ID/kDefaultTrapStyleName"
DefaultTrapWidth	"0.25"
BlackWidth	"0.5"
TrapJoin	"MiterEndJoin"
TrapEnd	"MiterTrapEnds"
ObjectsToImages	"true"
ImagesToImages	"true"
InternalImages	"false"
OneBitImages	"true"
ImagePlacement	"CenterEdges"
StepThreshold	"10"
BlackColorThreshold	"100"
BlackDensity	"1.6"
SlidingTrapThreshold	"70"
ColorReduction	"100"

### Transparency Preferences

The <TransparencyPreference> element defines the default transparency preferences.

**Table 74. Transparency Preferences Properties Represented as Attributes**

Attribute Name	Value
BlendingSpace	"CMYK"
GlobalLightAngle	"120"

Attribute Name	Value
GlobalLightAltitude	"30"

### Story Preferences

The `<StoryPreference>` element defines the default story preferences for the document.

---

**Table 75. Story Preferences Properties Represented as Attributes**

Attribute Name	Value
OpticalMarginAlignment	"false"
OpticalMarginSize	"12"
FrameType	"TextFrameType"
StoryOrientation	"Horizontal"
StoryDirection	"LeftToRightDirection"

### 1.1.17 Text Frame Preferences

The `<TextFramePreference>` element defines the default text frame preferences for the document.

---

**Table 76. Text Frame Preferences Properties Represented as Attributes**

Attribute Name	Value
TextColumnCount	"1"
TextColumnGutter	"12"
TextColumnFixedWidth	"144"
UseFixedColumnWidth	"false"
FirstBaselineOffset	"AscentOffset"
MinimumFirstBaselineOffset	"0"
VerticalJustification	"TopAlign"
VerticalThreshold	"0"
IgnoreWrap	"false"

---

**Table 77. Text Frame Preferences Properties Represented as Elements**

Element Name	Value
InsetSpacing	<pre>&lt;List Item type="unit"&gt;0&lt;/List Item&gt; &lt;List Item type="unit"&gt;0&lt;/List Item&gt; &lt;List Item type="unit"&gt;0&lt;/List Item&gt; &lt;List Item type="unit"&gt;0&lt;/List Item&gt;</pre>

### 1.1.18 Text Preferences

The `<TextPreference>` element defines the default text preferences for the document.

**Table 78. Text Preferences Properties Represented as Attributes**

Attribute Name	Value
TypographersQuotes	"true"
HighlightHjViolations	"false"
HighlightKeeps	"false"
HighlightSubstitutedGlyphs	"false"
HighlightCustomSpacing	"false"
HighlightSubstitutedFonts	"true"
UseOpticalSize	"true"
UseParagraphLeading	"false"
SuperscriptSize	"58.3"
SuperscriptPosition	"33.3"
SubscriptSize	"58.3"
SubscriptPosition	"33.3"
SmallCap	"70"
LeadingKeyIncrement	"2"
BaselineShiftKeyIncrement	"2"
KerningKeyIncrement	"20"
ShowInvisibles	"false"
JustifyTextWraps	"false"
AbutTextToTextWrap	"true"
ZOrderTextWrap	"false"
LinkTextFilesWhenImporting	"false"
HighlightKinsoku	"false"
UseNewVerticalScaling	"false"
UseCidMojikumi	"false"
EnableStylePreviewMode	"false"
SmartTextReflow	"true"
AddPages	"EndOfStory"
LimitToMasterTextFrames	"true"
PreserveFacingPageSpreads	"false"
DeleteEmptyPages	"false"

### 1.1.19 Text Defaults

The <TextDefault> element defines the default text formatting for the document.

**Table 79. Text Defaults Properties Represented as Attributes**

Attribute Name	Value
FirstLineIndent	"0"

Attribute Name	Value
LeftIndent	"0"
RightIndent	"0"
SpaceBefore	"0"
SpaceAfter	"0"
Justification	"LeftAlign"
SingleWordJustification	"FullyJustified"
AutoLeading	"120"
DropCapLines	"0"
DropCapCharacters	"0"
KeepLinesTogether	"false"
KeepAllLinesTogether	"false"
KeepWithNext	"0"
KeepFirstLines	"2"
KeepLastLines	"2"
StartParagraph	"Anywhere"
Composer	"HL Composer"
MinimumWordSpacing	"80"
MaximumWordSpacing	"133"
DesiredWordSpacing	"100"
MinimumLetterSpacing	"0"
MaximumLetterSpacing	"0"
DesiredLetterSpacing	"0"
MinimumGlyphScaling	"100"
MaximumGlyphScaling	"100"
DesiredGlyphScaling	"100"
RuleAbove	"false"
RuleAboveOverprint	"false"
RuleAboveLineWeight	"1"
RuleAboveTint	"-1"
RuleAboveOffset	"0"
RuleAboveLeftIndent	"0"
RuleAboveRightIndent	"0"
RuleAboveWidth	"ColumnWidth"
RuleAboveGapTint	"-1"
RuleAboveGapOverprint	"false"
RuleBelow	"false"
RuleBelowLineWeight	"1"
RuleBelowTint	"-1"
RuleBelowOffset	"0"

Attribute Name	Value
RuleBelowLeftIndent	"0"
RuleBelowRightIndent	"0"
RuleBelowWidth	"ColumnWidth"
RuleBelowGapTint	"-1"
HyphenateCapitalizedWords	"true"
Hyphenation	"true"
HyphenateBeforeLast	"2"
HyphenateAfterFirst	"2"
HyphenateWordsLongerThan	"5"
HyphenateLadderLimit	"3"
HyphenationZone	"36"
HyphenWeight	"5"
AppliedParagraphStyle	"ParagraphStyle/\$ID/ NormalParagraphStyle"
AppliedCharacterStyle	"CharacterStyle/\$ID/ [No character style]"
FontStyle	"Regular"
PointSize	"12"
KerningMethod	"\$ID/Metrics"
Tracking	"0"
Capitalization	"Normal"
Position	"Normal"
Underline	"false"
StrikeThru	"false"
Ligatures	"true"
NoBreak	"false"
HorizontalScale	"100"
VerticalScale	"100"
BaselineShift	"0"
Skew	"0"
FillTint	"-1"
StrokeTint	"-1"
StrokeWeight	"1"
OverprintStroke	"false"
OverprintFill	"false"
OTFFigureStyle	"Default"
OTFOrdinal	"false"
OTFFraction	"false"
OTFDiscretionaryLigature	"false"

Attribute Name	Value
OTFTitling	"false"
OTFContextualAlternate	"true"
OTFSwash	"false"
UnderlineTint	"-1"
UnderlineGapTint	"-1"
UnderlineOverprint	"false"
UnderlineGapOverprint	"false"
UnderlineOffset	"-9999"
UnderlineWeight	"-9999"
StrikeThroughTint	"-1"
StrikeThroughGapTint	"-1"
StrikeThroughOverprint	"false"
StrikeThroughGapOverprint	"false"
StrikeThroughOffset	"-9999"
StrikeThroughWeight	"-9999"
FillColor	"Color/Black"
StrokeColor	"Swatch/None"
AppliedLanguage	"\$ID/English: USA"
LastLineIndent	"0"
HyphenateLastWord	"true"
OTFSlashedZero	"false"
OTFHistorical	"false"
OTFStylisticSets	"0"
GradientFillLength	"-1"
GradientFillAngle	"0"
GradientStrokeLength	"-1"
GradientStrokeAngle	"0"
GradientFillStart	"0 0"
GradientStrokeStart	"0 0"
RuleBelowOverprint	"false"
RuleBelowGapOverprint	"false"
DropcapDetail	"0"
HyphenateAcrossColumns	"true"
KeepRuleAboveInFrame	"false"
IgnoreEdgeAlignment	"false"
OTFMark	"true"
OTFLocale	"true"
PositionalForm	"None"
ParagraphDirection	"LeftToRightDirection"

Attribute Name	Value
ParagraphJustification	"DefaultJustification"
MiterLimit	"4"
StrokeAlignment	"OutsideAlignment"
EndJoin	"MiterEndJoin"
OTFOverlapSwash	"false"
OTFStylisticAlternate	"false"
OTFJustificationAlternate	"false"
OTFStretchedAlternate	"false"
CharacterDirection	"DefaultDirection"
KeyboardDirection	"DefaultDirection"
DigitsType	"DefaultDigits"
Kashidas	"DefaultKashidas"
DiacriticPosition	"OpentypePosition"
XOffsetDiacritic	"0"
YOffsetDiacritic	"0"
ParagraphBreakType	"Anywhere"
PageNumberType	"AutoPageNumber"
AppliedNamedGrid	"n"
CharacterAlignment	"AlignEmCenter"
Tsume	"0"
LeadingAki	"-1"
TrailingAki	"-1"
CharacterRotation	"0"
Jidori	"0"
ShataiMagnification	"0"
ShataiDegreeAngle	"4500"
ShataiAdjustRotation	"false"
ShataiAdjustTsume	"true"
Tatechuyoko	"false"
TatechuyokoXOffset	"0"
TatechuyokoYOffset	"0"
KntenTint	"-1"
KntenStrokeTint	"-1"
KntenWeight	"-1"
KntenOverprintFill	"Auto"
KntenOverprintStroke	"Auto"
KntenKind	"None"
KntenPlacement	"0"
KntenAlignment	"AlignKntenCenter"

Attribute Name	Value
KntenPosition	"AboveRight"
KntenFontSize	"-1"
KntenXScale	"100"
KntenYScale	"100"
KntenCustomCharacter	""
KntenCharacterSet	"CharacterInput"
RubyTint	"-1"
RubyWeight	"-1"
RubyOverprintFill	"Auto"
RubyOverprintStroke	"Auto"
RubyStrokeTint	"-1"
RubyFontSize	"-1"
RubyOpenTypePro	"true"
RubyXScale	"100"
RubyYScale	"100"
RubyType	"PerCharacterRuby"
RubyAlignment	"RubyJIS"
RubyPosition	"AboveRight"
RubyXOffset	"0"
RubyYOffset	"0"
RubyParentSpacing	"RubyParent121Aki"
RubyAutoAlign	"true"
RubyOverhang	"false"
RubyAutoScaling	"false"
RubyParentScalingPercent	"66"
RubyParentOverhangAmount	"RubyOverhangOneRuby"
Warichu	"false"
WarichuSize	"50"
WarichuLines	"2"
WarichuLineSpacing	"0"
WarichuAlignment	"Auto"
WarichuCharsAfterBreak	"2"
WarichuCharsBeforeBreak	"2"
OTFProportionalMetrics	"false"
OTFHVKana	"false"
OTFRomanItalics	"false"
ScaleAffectsLineHeight	"false"
CjkGridTracking	"false"
GlyphForm	"None"

Attribute Name	Value
GridAlignFirstLineOnly	"false"
GridAlignment	"None"
GridGyoudori	"0"
AutoTcy	"0"
AutoTcyIncludeRoman	"false"
KinsokuType	"KinsokuPushInFirst"
KinsokuHangType	"None"
BunriKinshi	"true"
Rensuuji	"true"
RotateSingleByteCharacters	"false"
LeadingModel	"LeadingModelAkiBelow"
RubyAutoTcyDigits	"0"
RubyAutoTcyIncludeRoman	"false"
RubyAutoTcyAutoSize	"true"
TreatIdeographicSpaceAsSpace	"false"
AllowArbitraryHyphenation	"false"
ParagraphGyoudori	"false"
BulletsAndNumberingListType	"NoList"
NumberingExpression	"^#.^t"
BulletsTextAfter	"^t"
NumberingLevel	"1"
NumberingContinue	"true"
NumberingStartAt	"1"
NumberingApplyRestartPolicy	"true"
BulletsAlignment	"LeftAlign"
NumberingAlignment	"LeftAlign"

**Table 80. Text Defaults Properties Represented as Elements**

Element Name	Value
BalanceRaggedLines	NoBalancing
RuleAboveColor	Text Color
RuleAboveGapColor	Swatch/None
RuleAboveType	StrokeStyle/\$ID/Solid
RuleBelowColor	Text Color
RuleBelowGapColor	Swatch/None
RuleBelowType	StrokeStyle/\$ID/Solid
AppliedFont	Times New Roman
Leading	Auto

Element Name	Value
UnderlineColor	Text Color
UnderlineGapColor	Swatch/None
UnderlineType	StrokeStyle/\$ID/Solid
StrikeThroughColor	Text Color
StrikeThroughGapColor	Swatch/None
StrikeThroughType	StrokeStyle/\$ID/Solid
KentenFillColor	Text Color
KentenStrokeColor	Text Color
KentenFont	\$ID/
KentenFontStyle	Nothing
RubyFill	Text Color
RubyStroke	Text Color
RubyFont	\$ID/
RubyFontStyle	Nothing
KinsokuSet	Nothing
Mojikumi	Nothing
BulletChar	BulletCharacterType="UnicodeOnly" BulletCharacterValue="8226"
BulletsFont	\$ID/
BulletsFontStyle	Nothing
BulletsCharacterStyle	CharacterStyle/\$ID/ [No character style]
NumberingCharacterStyle	CharacterStyle/\$ID/ [No character style]
AppliedNumberingList	/\$ID/ [Default]
NumberingFormat	1, 2, 3, 4...
NumberingRestartPolicies	RestartPolicy="AnyPreviousLevel" LowerLevel="0" UpperLevel="0"

### 1.1.20 Anchored Object Defaults

The `<AnchoredObjectDefault>` element defines the default formatting and behavior of anchored objects in the document.

**Table 81. Anchored Object Defaults Properties Represented as Attributes**

Attribute Name	Value
AnchorContent	"Unassigned"
InitialAnchorHeight	"72"
InitialAnchorWidth	"72"
AnchoredParagraphStyle	"ParagraphStyle/\$ID/ [No paragraph style]"
AnchoredObjectStyle	"ObjectStyle/\$ID/ [None]"

### 1.1.21 Anchored Object Settings

The `<AnchoredObjectSetting>` element defines the default anchored objects settings used in the document.

**Table 82. Anchored Object Settings Properties Represented as Attributes**

Attribute Name	Value
AnchoredPosition	"InlinePosition"
SpineRelative	"false"
LockPosition	"false"
PinPosition	"true"
AnchorPoint	"BottomRightAnchor"
HorizontalAlignment	"LeftAlign"
HorizontalReferencePoint	"TextFrame"
VerticalAlignment	"TopAlign"
VerticalReferencePoint	"LineBaseline"
AnchorXoffset	"0"
AnchorYoffset	"0"
AnchorSpaceAbove	"0"

### 1.1.22 Baseline Frame Grid Options

The `<BaselineFrameGridOption>` element defines the default anchored objects settings used in the document.

**Table 83. Basic Frame Grid Options Properties Represented as Attributes**

Attribute Name	Value
UseCustomBaselineFrameGrid	"false"
StartingOffsetForBaselineFrameGrid	"0"
BaselineFrameGridRelativeOption	"TopOfInset"
BaselineFrameGridIncrement	"12"

**Table 84. Baseline Frame Grid Options Represented as Elements**

Element Name	Value
BaselineFrameGridColor	LightBlue

### 1.1.23 Footnote Options

The `<FootnoteOption>` element defines the default footnote options used in the document.

**Table 85. Footnote Options Properties Represented as Attributes**

Attribute Name	Value
StartAt	"1"
Prefix	""
Suffix	""
FootnoteTextStyle	"ParagraphStyle/\$ID/ NormalParagraphStyle"
FootnoteMarkerStyle	"CharacterStyle/\$ID/ [No character style]"
SeparatorText	"&#x9;"
SpaceBetween	"0"
Spacer	"0"
FootnoteFirstBaselineOffset	"LeadingOffset"
FootnoteMinimumFirstBaselineOffset	"0"
EosPlacement	"false"
NoSplitting	"false"
RuleOn	"true"
RuleLineWeight	"1"
RuleTint	"100"
RuleGapTint	"100"
RuleGapOverprint	"false"
RuleOverprint	"false"
RuleLeftIndent	"0"
RuleWidth	"72"
RuleOffset	"0"
ContinuingRuleOn	"true"
ContinuingRuleLineWeight	"1"
ContinuingRuleTint	"100"
ContinuingRuleGapTint	"100"
ContinuingRuleOverprint	"false"
ContinuingRuleGapOverprint	"false"
ContinuingRuleLeftIndent	"0"
ContinuingRuleWidth	"288"
ContinuingRuleOffset	"0"

**Table 86. Footnote Options Properties Represented as Elements**

Element Name	Value
FootnoteNumberingStyle	Arabic
RestartNumbering	DontRestart
ShowPrefixSuffix	NoPrefixSuffix

Element Name	Value
MarkerPositioning	SuperscriptMarker
RuleType	StrokeStyle/\$ID/Solid
RuleColor	Color/Black
RuleGapColor	Swatch/None
ContinuingRuleType	StrokeStyle/\$ID/Solid
ContinuingRuleColor	Color/Black
ContinuingRuleGapColor	Swatch/None

### 1.1.24 Text Wrap Preferences

The `<TextWrapPreference>` element defines the default text wrap preferences used in the document. The `<TextWrapPreference>` element contains a `<ContourOption>` element.

**Table 87. Text Wrap Preferences Properties Represented as Attributes**

Attribute Name	Value
Inverse	"false"
ApplyToMasterPageOnly	"false"
TextWrapSide	"BothSides"
TextWrapMode	"None"

**Table 88. Text Wrap Preferences Properties Represented as Elements**

Element Name	Value
TextWrapOffset	Top="0" Left="0" Bottom="0" Right="0"

**Table 89. Contour Option Properties Represented as Attributes**

Attribute Name	Value
ContourType	"SameAsClipping"
IncludeInsideEdges	"false"
ContourPathName	"\$ID/"

### 1.1.25 Document Preferences

The `<DocumentPreference>` element defines the default document preferences.

**Table 90. DocumentPreference Properties Represented as Attributes**

Attribute Name	Value
PageHeight	"792"

Attribute Name	Value
PageWidth	"612"
PagesPerDocument	"1"
FacingPages	"true"
DocumentBleedTopOffset	"0"
DocumentBleedBottomOffset	"0"
DocumentBleedInsideOrLeftOffset	"0"
DocumentBleedOutsideOrRightOffset	"0"
DocumentBleedUniformSize	"true"
SlugTopOffset	"0"
SlugBottomOffset	"0"
SlugInsideOrLeftOffset	"0"
SlugRightOrOutsideOffset	"0"
DocumentSlugUniformSize	"false"
PreserveLayoutWhenShuffling	"true"
AllowPageShuffle	"true"
OverprintBlack	"true"
PageBinding	"LeftToRight"
ColumnDirection	"Horizontal"
ColumnGuideLocked	"true"
MasterTextFrame	"false"
SnippetImportUsesOriginalLocation	"false"

**Table 91. Document Preferences Properties Represented as Elements**

Element Name	Value
ColumnGuideColor	Violet
MarginGuideColor	Magenta

### 1.1.26 Margin Preferences

The <MarginPreference> element defines the default margin preferences used in the document.

**Table 92. Margin Preferences Properties Represented as Attributes**

Attribute Name	Value
ColumnCount	"1"
ColumnGutter	"12"
Top	"36"
Bottom	"36"
Left	"36"

Attribute Name	Value
Right	"36"
ColumnDirection	"Horizontal"

### 1.1.27 Page Item Defaults

The `<PageItemDefault>` element defines the default formatting used for page items.

**Table 93. Page Item Defaults Properties Represented as Attributes**

Attribute Name	Value
AppliedGraphicObjectStyle	"ObjectStyle/\$ID/[Normal Graphics Frame]"
AppliedTextObjectStyle	"ObjectStyle/\$ID/[Normal Text Frame]"
AppliedGridObjectStyle	"ObjectStyle/\$ID/[Normal Grid]"
FillColor	"Swatch/None"
FillTint	"-1"
StrokeWeight	"1"
MiterLimit	"4"
EndCap	"ButtEndCap"
EndJoin	"MiterEndJoin"
StrokeType	"StrokeStyle/\$ID/Solid"
LeftLineEnd	"None"
RightLineEnd	"None"
StrokeColor	"Swatch/None"
StrokeTint	"-1"
CornerOption	"None"
CornerRadius	"12"
GradientFillAngle	"0"
GradientStrokeAngle	"0"
GapColor	"Swatch/None"
GapTint	"-1"
StrokeAlignment	"CenterAlignment"
Nonprinting	"false"

### 1.1.28 Frame Fitting Options

The `<FrameFittingOption>` element defines the default frame fitting options used in the document.

**Table 94. Frame Fitting Properties Represented as Attributes**

Attribute Name	Value
LeftCrop	"0"
TopCrop	"0"
RightCrop	"0"
BottomCrop	"0"
FittingOnEmptyFrame	"None"
FittingAlignment	"TopLeftAnchor"

### 1.1.29 Button Preferences

The `<ButtonPreference>` element defines the default button name used in the document.

**Table 95. Button Preferences Properties Represented as Attributes**

Attribute Name	Value
Name	"\$ID/"

### 1.1.30 Conditional Text Preferences

The `<ConditionalTextPreference>` element defines the defaults for conditional text used in the document.

**Table 96. Conditional Text Preferences Properties Represented as Attributes**

Attribute Name	Value
ShowConditionIndicators	"ShowIndicators"
ActiveConditionSet	"n"

### 1.1.31 XML Tag

The `<XMLTag>` element defines the default XML tag used in the document.

**Table 97. XML Tag Properties Represented as Attributes**

Attribute Name	Value
Name	"Root"

**Table 98. XMLTag Properties Represented as Elements**

Attribute Name	Value
TagColor type	"enumeration">LightBlue

### 1.1.32 Layer

The <Layer> element defines the default layer used in the document.

**Table 99. Layer Properties Represented as Attributes**

Attribute Name	Value
Name	"Layer 1"
Visible	"true"
Locked	"false"
IgnoreWrap	"false"
ShowGuides	"true"
LockGuides	"false"
UI	"true"
Expendable	"true"
Printable	"true"

**Table 100. Layer Properties Represented as Elements**

Attribute Name	Value
LayerColor type	"enumeration">LightBlue

### 1.1.33 Master Spread

The <MasterSpread> element defines the default master spread of the document.

The <MasterSpread> element contains two <Page> elements, which, in turn, contain <MarginPreference> and <GridDataInformation> elements.

**Table 101. Master Spread Properties Represented as Attributes**

Attribute Name	Value
ItemTransform	"1 0 0 1 0 0"
Name	"A-Master"
NamePrefix	"A"
BaseName	"Master"
ShowMasterItems	"true"
PageCount	"2"
OverriddenPageItemProps	""

The two <Page> elements are identical.

**Table 102. Page Properties Represented as Attributes**

Attribute Name	Value
Name	"A"
AppliedTrapPreset	"TrapPreset/\$ID/kDefaultTrapStyleName"
AppliedMaster	"n"
OverrideList	""
TabOrder	""
GridStartingPoint	"TopOutside"
UseMasterGrid	"true"

Each <Page> element contains a <MarginPreference> element.

**Table 103. Margin Preferences Properties Represented as Attributes**

Attribute Name	Value
ColumnCount	"1"
ColumnGutter	"12"
Top	"36"
Bottom	"36"
Left	"36"
Right	"36"
ColumnDirection	"Horizontal"
ColumnsPositions	"0 540"

Each <Page> element contains a <GridDataInformation> element.

**Table 104. Grid Data Information Properties Represented as Attributes**

Attribute Name	Value
FontStyle	"Regular"
PointSize	"12"
CharacterAki	"0"
LineAki	"9"
HorizontalScale	"100"
VerticalScale	"100"
LineAlignment	"LeftOrTopLineJustify"
GridAlignment	"AlignEmCenter"
CharacterAlignment	"AlignEmCenter"

**Table 105. Grid Data Information Properties Represented as Elements**

Attribute Name	Value
AppliedFont	Times New Roman

**1.1.34 Page**

The <Page> element defines the default page used in the document.

**Table 106. Page Properties Represented as Attributes**

Attribute Name	Value
Name	"A"
AppliedTrapPreset	"TrapPreset/\$ID/kDefaultTrapStyleName"
AppliedMaster	"n"
OverrideList	" "
TabOrder	" "
GridStartingPoint	"TopOutside"
UseMasterGrid	"true"

Each <Page> element contains a <MarginPreference> element.

**Table 107. Margin Preferences Properties Represented as Attributes**

Attribute Name	Value
ColumnCount	"1"
ColumnGutter	"12"
Top	"36"
Bottom	"36"
Left	"36"
Right	"36"
ColumnDirection	"Horizontal"
ColumnsPositions	"0 540"

Each <Page> element contains a <GridDataInformation> element.

**Table 108. Grid Data Information Properties Represented as Attributes**

Attribute Name	Value
FontStyle	"Regular"
PointSize	"12"
CharacterAki	"0"
LineAki	"9"
HorizontalScale	"100"

Attribute Name	Value
VerticalScale	"100"
LineAlignment	"LeftOrTopLineJustify"
GridAlignment	"AlignEmCenter"
CharacterAlignment	"AlignEmCenter"

**Table 109. Grid Data Information Properties Represented as Elements**

Element Name	Value
AppliedFont	Times New Roman

### 1.1.35 Spread

The `<Spread>` element defines the default spread used in the document. The `<Spread>` element contains a `<FlattenerPreference>` element. The `<Spread>` element contains a `<Page>` element.

**Table 110. Spread Properties Represented as Attributes**

Attribute Name	Value
PageTransitionType	"None"
PageTransitionDirection	"NotApplicable"
PageTransitionDuration	"Medium"
FlattenerOverride	"Default"
ShowMasterItems	"true"
PageCount	"1"
BindingLocation	"0"
AllowPageShuffle	"true"
ItemTransform	"1 0 0 1 0 0"

**Table 111. Flattener Preference Properties Represented as Attributes**

Attribute Name	Value
LineArtAndTextResolution	"300"
GradientAndMeshResolution	"150"
ClipComplexRegions	"false"
ConvertAllStrokesToOutlines	"false"
ConvertAllTextToOutlines	"false"

**Table 112. Flattener Preference Properties Represented as Elements**

Element Name	Value
RasterVectorBalance	50

**Table 113. Page Properties Represented as Attributes**

Attribute Name	Value
Name	"A"
AppliedTrapPreset	"TrapPreset/\$ID/kDefaultTrapStyleName"
AppliedMaster	"n"
OverrideList	""
TabOrder	""
GridStartingPoint	"TopOutside"
UseMasterGrid	"true"

Each <Page> element contains a <MarginPreference> element.

**Table 114. Margin Preferences Properties Represented as Attributes**

Attribute Name	Value
ColumnCount	"1"
ColumnGutter	"12"
Top	"36"
Bottom	"36"
Left	"36"
Right	"36"
ColumnDirection	"Horizontal"
ColumnsPositions	"0 540"

Each <Page> element contains a <GridDataInformation> element.

**Table 115. Grid Data Information Properties Represented as Attributes**

Attribute Name	Value
FontStyle	"Regular"
PointSize	"12"
CharacterAki	"0"
LineAki	"9"
HorizontalScale	"100"
VerticalScale	"100"
LineAlignment	"LeftOrTopLineJustify"
GridAlignment	"AlignEmCenter"
CharacterAlignment	"AlignEmCenter"

**Table 116. Grid Data Information Properties Represented as Elements**

Attribute Name	Value
AppliedFont	Times New Roman

### 1.1.36 Section

The <Section> element defines the default section used in the document.

**Table 117. Section Properties Represented as Attributes**

Attribute Name	Value
Length	"1"
Name	""
PageNumberStyle	"Arabic"
ContinueNumbering	"true"
IncludeSectionPrefix	"false"
Marker	""
PageStart	"ube"
SectionPrefix	""

### 1.1.37 XmlStory

The <xmlStory> element defines the default XML story (unplaced XML text elements) of the document. The <XmlStory> element contains a <StoryPreference> element and a <ParagraphStyleRange> element.

**Table 118. Xml Story Properties Represented as Attributes**

Attribute Name	Value
AppliedTOCStyle	"n"
TrackChanges	"false"
StoryTitle	"\$ID/ "
AppliedNamedGrid	"n"

**Table 119. Story Preferences Properties Represented as Attributes**

Attribute Name	Value
OpticalMarginAlignment	"false"
OpticalMarginSize	"12"
FrameType	"TextFrameType"
StoryOrientation	"Horizontal"
StoryDirection	"LeftToRightDirection"

**Table 120. Paragraph Style Range Properties Represented as Attributes**

Attribute Name	Value
AppliedParagraphStyle	"ParagraphStyle/\$ID/ NormalParagraphStyle"

The <ParagraphStyleRange> element contains a <CharacterStyleRange> element.

**Table 121. Character Style Range Properties Represented as Attributes**

Attribute Name	Value
AppliedCharacterStyle	"CharacterStyle/\$ID/[No character style]"

**Table 122. Character Style Range Properties Represented as Elements**

Element Name	Value
XMLElement	MarkupTag="XMLTag/Root"
Content	ï"¿

### 1.1.38 IndexingSortOptions

The <IndexingSortOptions> elements defines the default indexing sort options for the document. The default document contains eight <IndexSortOptions> elements with the following names: \$ID/kIndexGroup\_Symbol, \$ID/kIndexGroup\_Alphabet, \$ID/kIndexGroup\_Numeric, \$ID/kWRIndexGroup\_GreekAlphabet, \$ID/kWRIndexGroup\_CyrillicAlphabet, \$ID/kIndexGroup\_Kana, \$ID/kIndexGroup\_Chinese, \$ID/kIndexGroup\_Korean.

#### \$ID/kIndexGroup\_Symbol

**Table 123. Indexing Sort Options Properties Represented as Attributes**

Attribute Name	Value
Name	"\$ID/kIndexGroup_Symbol"
Include	"true"
Priority	"0"
HeaderType	"Nothing"

#### \$ID/kIndexGroup\_Alphabet

**Table 124. Indexing Sort Options Properties Represented as Attributes**

Attribute Name	Value
Name	"\$ID/kIndexGroup_Alphabet"
Include	"true"

Attribute Name	Value
Priority	"1"
HeaderType	"BasicLatin"

### \$ID/kIndexGroup\_Numeric

**Table 125. Indexing Sort Options Properties Represented as Attributes**

Attribute Name	Value
Name	"\$ID/kIndexGroup_Numeric"
Include	"false"
Priority	"2"
HeaderType	"Nothing"

### \$ID/kWRIndexGroup\_GreekAlphabet

**Table 126. Indexing Sort Options Properties Represented as Attributes**

Attribute Name	Value
Name	"\$ID/kWRIndexGroup_GreekAlphabet"
Include	"false"
Priority	"3"
HeaderType	"Nothing"

### \$ID/kWRIndexGroup\_CyrillicAlphabet

**Table 127. Indexing Sort Options Properties Represented as Attributes**

Attribute Name	Value
Name	"\$ID/kWRIndexGroup_CyrillicAlphabet"
Include	"false"
Priority	"4"
HeaderType	"Russian"

### \$ID/kIndexGroup\_Kana

**Table 128. Indexing Sort Options Properties Represented as Attributes**

Attribute Name	Value
Name	"\$ID/kIndexGroup_Kana"
Include	"false"

Attribute Name	Value
Priority	"5"
HeaderType	"HiraganaAll"

### \$ID/kIndexGroup\_Chinese

**Table 129. Indexing Sort Options Properties Represented as Attributes**

Attribute Name	Value
Name	"\$ID/kIndexGroup_Chinese"
Include	"false"
Priority	"6"
HeaderType	"ChinesePinyin"

### \$ID/kIndexGroup\_Korean

**Table 130. Indexing Sort Options Properties Represented as Attributes**

Attribute Name	Value
Name	"\$ID/kIndexGroup_Korean"
Include	"false"
Priority	"7"
HeaderType	"KoreanConsonant"

### 1.1.39 Bullets

A series of <ABullet> elements define the default bullets for the document. These elements are named dABullet0, dABullet1, dABullet2, dABullet3, and dABullet4.

#### dABullet0

**Table 131. Bullet Properties Represented as Attributes**

Attribute Name	Value
CharacterType	"UnicodeOnly"
CharacterValue	"8226"

#### Bullet Properties Represented as Elements

Element Name	Value
BulletsFont	\$ID/
BulletsFontStyle	\$ID/

**dABullet1****Table 132. Bullet Properties Represented as Attributes**

Attribute Name	Value
CharacterType	"UnicodeOnly"
CharacterValue	"42"

**Table 133. Bullet Properties Represented as Elements**

Element Name	Value
BulletsFont	\$ID/
BulletsFontStyle	\$ID/

**dABullet2****Table 134. Bullet Properties Represented as Attributes**

Attribute Name	Value
CharacterType	"UnicodeOnly"
CharacterValue	"9674"

**Table 135. Bullet Properties Represented as Elements**

Element Name	Value
BulletsFont	\$ID/
BulletsFontStyle	\$ID/

**dABullet3****Table 136. Bullet Properties Represented as Attributes**

Attribute Name	Value
CharacterType	"UnicodeWithFont"
CharacterValue	"187"

**Table 137. Bullet Properties Represented as Elements**

Element Name	Value
BulletsFont	Myriad_Pro
BulletsFontStyle	\$ID/Regular

**dABullet4****Table 138. Bullet Properties Represented as Attributes**

Attribute Name	Value
CharacterType	"GlyphWithFont"
CharacterValue	"503"

**Table 139. Bullet Properties Represented as Elements**

Element Name	Value
BulletsFont	Minion Pro
BulletsFontStyle	\$ID/Regular

**1.1.40 Assignment**

The <Assignment> element defines the default assignment used in the document.

**Table 140. Assignment Properties Represented as Attributes**

Attribute Name	Value
Name	"\$ID/UnassignedInCopy"
UserName	"\$ID/"
ExportOptions	"AssignedSpreads"
IncludeLinksWhenPackage	"true"
FilePath	"\$ID/"

**Table 141. Assignment Properties Represented as Elements**

Element Name	Value
FrameColor	Nothing