Mars Reference

Version 0.8

Draft

November 2007

Contents

Preface

Adobe® Acrobat® Professional, Acrobat Standard, and Adobe Reader® provide support for an XML-based representation of Portable Document Format (PDF) documents known as Mars. This document describes the Mars format in terms of its schema and relationship to PDF.

What's in this guide

This document describes the format of Mars documents by describing portions of the Relax NG schemas that define the Mars format. Where applicable, this document shows the relationship of those schemas to the PDF format (version 1.7).

Who should read this guide

The primary audience of this document are developers who want to leverage their XML tools and knowledge to create, manipulate, and extract information from PDF documents.

This document assumes that you are familiar with the Acrobat product family and that you are an experienced user of Acrobat products. You should understand XML and be familiar with programming on your development platform.

Related documentation

The resources listed in this table can help you learn more about the product.

For Information about	See
Adobe Developer Support	www.adobe.com/go/acrobat_developer
RELAX NG	www.relaxng.org
Developing Applications Using Mars	www.adobe.com/go/mars
RNC Schemas	www.adobe.com/go/mars
PDF Reference, version 1.7	www.adobe.com/go/acrobatsdk_pdf_reference
OEBPS Container Format (OCF)	http://idpf.org/doc_library/informationaldocs/ocf10-20060421.pdf
XML schema data types from W3C	www.w3.org/TR/2004/REC-xml-20040204/
SVG 1.1	*Scalable Vector Graphics (SVG) 1.1 Specification* www.w3.org/TR/SVG11/

For Information about	See
SVG 1.2	*Scalable Vector Graphics (SVG) 1.2 Specification* www.w3.org/TR/SVG12/
Tiny 1.2	*Scalable Vector Graphics (SVG) Tiny 1.2 Specification* www.w3.org/TR/SVGMobile12

1 Introduction

This chapter provides a summary of the overall document structure, packaging, and basis of Mars. It describes the correlation between the Mars and PDF formats and it discusses the schemas that provide the definition of Mars.

Understanding Mars and PDF

Mars provides an XML-based representation of PDF documents. It represents document information by combining standard XML, images, fonts, and color formats within a Zip-based package. For more information on the Mars document structure, see *Developing Applications using Mars*. Page content is represented in SVG, an XML-based W3C standard. In addition, dictionaries in a PDF document that are not defined in the *PDF Reference* can be converted to and from XML.

The information in Mars documents is identical to what would appear in PDF documents, and is organized in a conceptually similar way. The Mars format makes component subassemblies in its PDF counterparts more independent and easier to manipulate by virtue of their XML representation. Thus, Mars addresses the desire in the developer community to comply and interoperate with XML and standards-based formats. Mars provides a representation of PDF that is readily understood and is interchangeable with the PDF format.

There are several kinds of operations you can perform on Mars documents:

- Creation: The XML format makes it possible for you to use XML tools to create documents.
- Manipulation of auxiliary content: Mars makes it easier to create and manipulate the non-content parts of a document, including annotations, bookmarks, JavaScript™, fonts, metadata, external references, specialized processing data, and attachments.
- Document assembly and disassembly: Mars is designed to make page-level assembly of documents easy.
- Page content creation and manipulation: Mars supports manipulation of page content via SVG.

In addition, Acrobat and Adobe Reader can open Mars documents, and Acrobat can be used to save a Mars document as a PDF document or vice versa.

Schemas

The Mars format is described in a series of RELAX NG schemas. The following table lists the schemas that describe the syntax of the Mars format. This table also lists the files that contain each schema. You can obtain these schemas from the Mars Developer web site, at www.adobe.com/go/mars.

Schema	Schema file name
Backbone	backbone.rnc
Bookmarks	bookmarks.rnc
Article threads	articles.rnc

Schema	Schema file name
Application data	appdata.rnc
Default annotation appearances	appearances.rnc
JavaScript	javascripts.rnc
Web IDs	web_ids.rnc
Web URLs	web_urls.rnc
Embedded files	embedded_files.rnc
Page information	pageinfo.rnc
Page structure	pagestruct.rnc
Font descriptors	fontdesc.rnc
Markup annotations	markup_annot.rnc
Content annotations	content_annot.rnc
Named destinations	pagenameddest.rnc
Named destination cache	named_dest_cache.rnc
Structure tree cache	structure_tree_cache.rnc

2 Conventions for Packaging, Organizing, and Encoding Mars Documents

This chapter describes the physical structure of Mars documents, including the recommended directory and file naming conventions.

This chapter contains the following information.

Topic	Description	See
Package format of a Mars document	Introduces the package format of a Mars document.	page 14
Organization of the files that comprise a Mars document	Explains the structure of Mars documents and certain rules that apply across all parts of a Mars document.	page 14
XML conventions used in Mars documents	Explains the conventions used in Mars documents.	page 26

Package format of a Mars document

A *Mars document* (also called a *Mars package*) is a ZIP file that contains a collection of files that represent the document or that represent information about the document. The ZIP file must fulfill the additional requirements described in “Universal Container Format” on page 221 .

Organization of the files that comprise a Mars document

A *Mars document* consists of a number of files that, when interpreted, represent a document that can have the same characteristics as a PDF document. The grammar of the non-SVG files in a Mars document is specified in “Mars Elements and Attributes” on page 54 and is specified in the Mars schemas (“Schemas” on page 12).

The file names within a Mars document must conform to restrictions defined by “Universal Container Format” on page 221.

The structure of the files in a Mars document is a hierarchy that starts with a single root file. The root file has references to other files in the Mars document, which in turn can have references to other files.

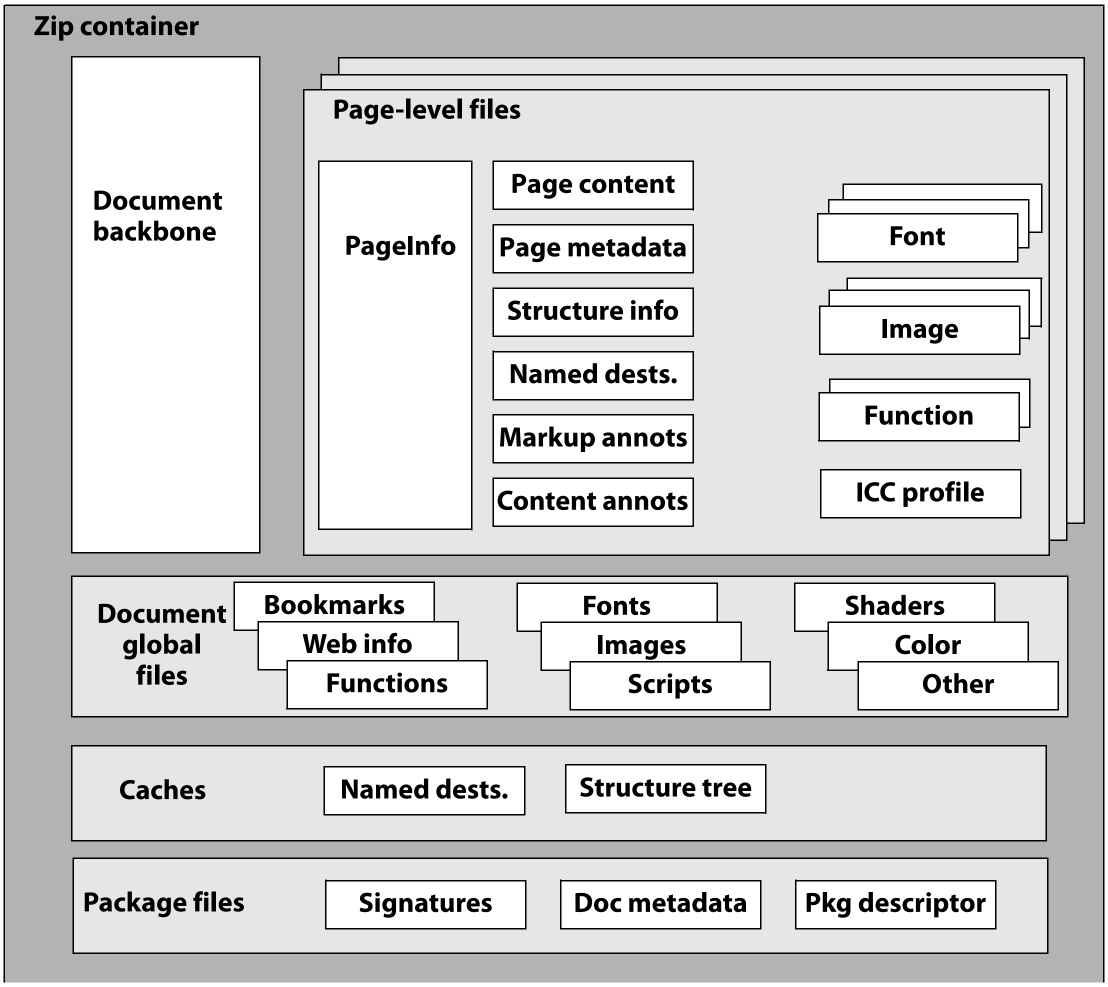
The structure of the files in a Mars document is also reflected in the directory and file naming conventions described here and used throughout this guide. Most of these conventions are optional; however, their use can improve consistency between documents. This guide prefaces non-required directory and file names with the term *conventionally* to indicate their use is suggested by convention rather than by requirement, as reflected in the following sentence:

> Conventionally, the directory containing the files that comprise a page is named /page/*page-number*, where *page-number* is the zero-based page number of the page.

The root file, called a backbone file, describes the overall document and references other files that provide subordinate aspects of the document. Examples of such subordinate files are the page information files (one per page), the bookmarks file, and the threads file. Each page information file describes a single page in the document and references other files that provide the page content and other data unique to that page.

The following figure shows the structure of a Mars document. Actual Mars documents contain only those files needed to represent their content, which is typically a few page-level files per page and a few of the other types of files.

Structure of a Mars document



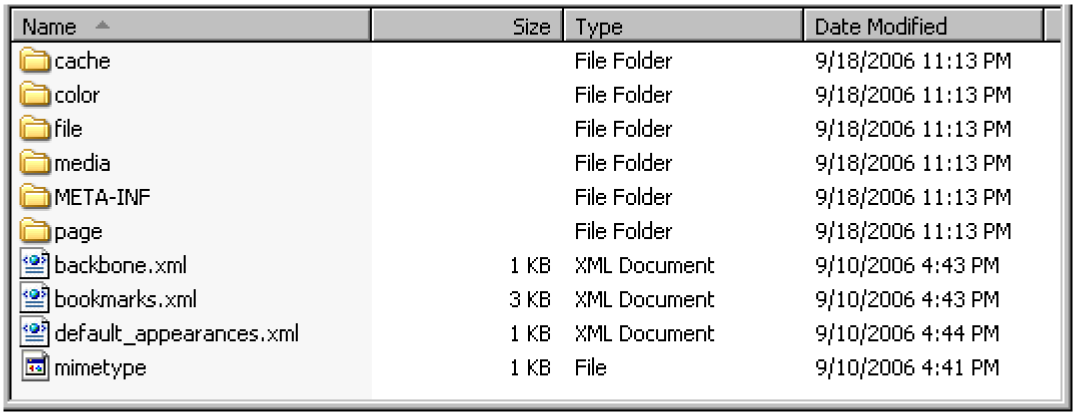
Directory naming conventions

The following table lists the directories that can appear in a Mars document and specifies which directories are required to be present in the document. A “Yes” in the column titled “Required” indicates the directory must be present in the Mars document and must be named as specified. A “No” in this column indicates the directory is optional. For optional directories, the directory name provided in the table is recommended by convention.

Directory name	Required	Contents
/META-INF	Yes	Package-related files. "Meta-information files" on page 25 provides more information. This directory must be created in the root directory of the Mars package.
/cache	No	Files that specify associations between elements in different package files, associations that would otherwise be complex or time-consuming to determine. "Cache-related files" on page 24 provides more information.
/page	No	Subdirectories for each page in the document. The subdirectories are named as the zero-based page number of the page they represent. For example, the /page/0 directory contains the contents of the first page in the document. "Page-related files" on page 20 provides more information.
/form	No	Either the parts from an XML form or the data from an Acrobat form. "Form files" on page 26 provides more information.
/file	No	Embedded or attached files.
/font	No	Fonts and font descriptors that are used on multiple pages. "Fonts and font descriptors" on page 19 provides more information.
/color	No	Color spaces and color look-up tables that are used on multiple pages in the document. "Color profiles" on page 20 provides more information.
/image	No	Image files used on multiple pages or elsewhere in the document. "Images" on page 19 provides more information.
/misc	No	Other kinds of files that are used by objects within the document that do not fall into one of the other categories.
/script	No	Globally-defined JavaScript expressions that can be invoked by JavaScript expressions that appear in JavaScript actions.
/annot	No	Additional files required by annotations.
/media	No	Sound or movie annotation files.
/res	No	Resources used by graphic content, generally functions, patterns, shaders, etc.
/web	No	Data files used by web capture. "Web information" on page 18 provides more information.

The following figure shows the top-level file directory structure for a typical Mars document.

Top-level file directory structure



Name	Size	Type	Date Modified
cache		File Folder	9/18/2006 11:13 PM
color		File Folder	9/18/2006 11:13 PM
file		File Folder	9/18/2006 11:13 PM
media		File Folder	9/18/2006 11:13 PM
META-INF		File Folder	9/18/2006 11:13 PM
page		File Folder	9/18/2006 11:13 PM
backbone.xml	1 KB	XML Document	9/10/2006 4:43 PM
bookmarks.xml	3 KB	XML Document	9/10/2006 4:43 PM
default_appearances.xml	1 KB	XML Document	9/10/2006 4:44 PM
mimetype	1 KB	File	9/10/2006 4:41 PM

Mars root-level files

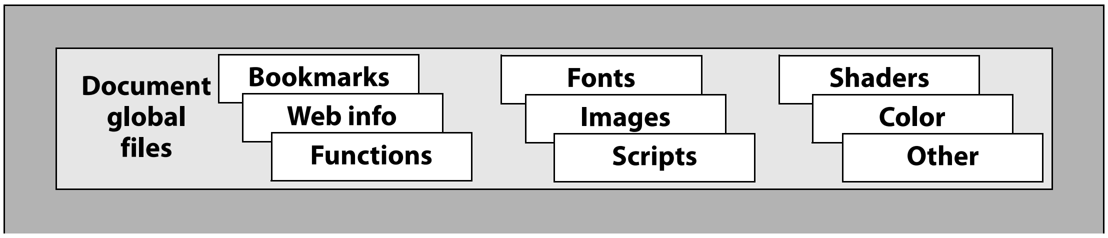
The following table lists the files that can appear in the root directory of a Mars document and specifies which files are required to be present in the document. A "Yes" in the column titled "Required" indicates the file must be present in the Mars document and must be named as specified. A "No" in this column indicates the file is optional. For optional files, the file name provided in the table is recommended by convention.

File	Required	Description
mimetype	Yes	This file must be the first physical file in the Zip package for the Mars document. Its contents must be the Mars MIME type, which is `application/vnd.adobe.x-mars`.
backbone.xml	Yes	This file must be in the root directory of the Mars package. It acts as the root file of the document. It describes these general properties: ● Level of Mars represented by the package ● Viewer preferences ● References to the page-level information files that comprise the document ● References to other files that add features to the Mars document, such as forms, page labels, bookmarks, and articles The name backbone.xml is reserved for this file only.
default_appearances.xml	No	This file contains named appearances that define the default characteristics used to render text. These characteristics include font family, font size, font color and font style. Certain types of annotations can reference these appearances rather than individually specifying them.
Bookmarks.xml	No	This file contains the document bookmarks, which are also called *outline items*. See "Bookmarks" on page 18.

Document global files

There is certain information that is globally available to a Mars document. This includes information about bookmarks, web information, fonts, images, and scripts. For instance, when fonts and images are created globally, they can be shared by various pages within a Mars document.

Structure of document global files



Bookmarks

A bookmarks file (bookmarks.xml) contains the document bookmarks, which are also called *outline items*. The bookmarks file is referenced from the Mars document backbone file.

Bookmarks allow the user to navigate from one part of the document to another or to initiate an action such as bringing up a web site in a browser window. Bookmarks are structured to represent the hierarchy of the document, which serve as a visual table of contents to display the document's structure to the user. For more information about bookmarks, see the *Developing Applications Using Mars*.

Web information

Web Capture is a feature that allows information from Internet-based or locally resident HTML, PDF, GIF, JPEG, and ASCII text files to be imported into a Mars file. The information in the Web Capture data elements enables viewer applications to perform the following operations:

- Save locally and preserve the visual appearance of material from the web
- Retrieve additional material from the web and add it to an existing Mars file
- Update or modify existing material previously captured from the web
- Find source information for material captured from the web, such as the URL (if any) from which it was captured
- Find all material in a Mars file that was generated from a URL
- Find all material in a Mars file that matches a digital identifier

In the Mars file format, Web Capture information is stored in the Mars document backbone file. This information generally provides the context in which the Mars was created from web pages and allows the Mars document to be updated from the same web pages. The URL and ID maps that are in the `Name` dictionary in PDF are stored in separate package files that are referenced from the document backbone.

Functions

The Mars file format provides several types of function objects that represent parameterized classes of functions, including mathematical formulas and sampled representations with arbitrary resolution. Functions are used in various ways in Mars, including device-dependent rasterization information for high-quality printing (halftone spot functions and transfer functions), color transform functions for certain color spaces, and specification of colors as a function of position for smooth shadings. All functions required for a Mars document should be added to the /res directory.

Fonts and font descriptors

Each OpenType font used in the Mars document must be represented by a font descriptor file that uniquely identifies the font and that provides information about the font. If the font is embedded, the font descriptor references the file containing the font. Font descriptors are represented in XML and can be shared across pages in a Mars document.

Note: Currently, Mars supports OpenType and Type 3 fonts, embedded or referenced. It also supports embedded SVG fonts.

Fonts used only on a single page are conventionally located in the directory for that page. Fonts used on multiple pages should be placed in the /font directory. For more information, see *Developing Applications Using Mars*.

Font descriptors (OpenType only) are stored within the directory for the page that uses them or in the /font directory.

Note: The existence of a font or font descriptor in the package does not imply that it is used in any way. Only the references from other package objects establish a using relationship.

Images

In Mars documents, bitmap images reside in separate files that are referenced from the individual pages that use them. Supported image formats include standard representations, such as JPEG and TIF.

Conventionally, images used on a single page in the Mars document are stored in the /page directory for the page, and images that are used on multiple pages are stored in the /image directory. This approach reduces duplicated images. For more information, see *Developing Applications Using Mars*.

Scripts (JavaScript actions)

JavaScript actions that are used across multiple pages in a Mars document are conventionally stored in a separate package-level file called *javascripts.xml*. The JavaScript actions in this file are uniquely named, which enables annotations and other action-recipient properties to reference those JavaScript actions by name. JavaScript actions can also be specified in the annotation or other action-recipient property that uses them. For more information, see *Developing Applications Using Mars*.

Shader objects

When using images in a Mars document, if the area to be painted is a relatively simple shape whose geometry is the same as that of the gradient fill itself, a shader can be used instead of the usual painting operators for the image.

Mars documents that contain functions to be used as parts of graphic objects should use shader objects. All shader data files for a Mars document should be added to the /res directory.

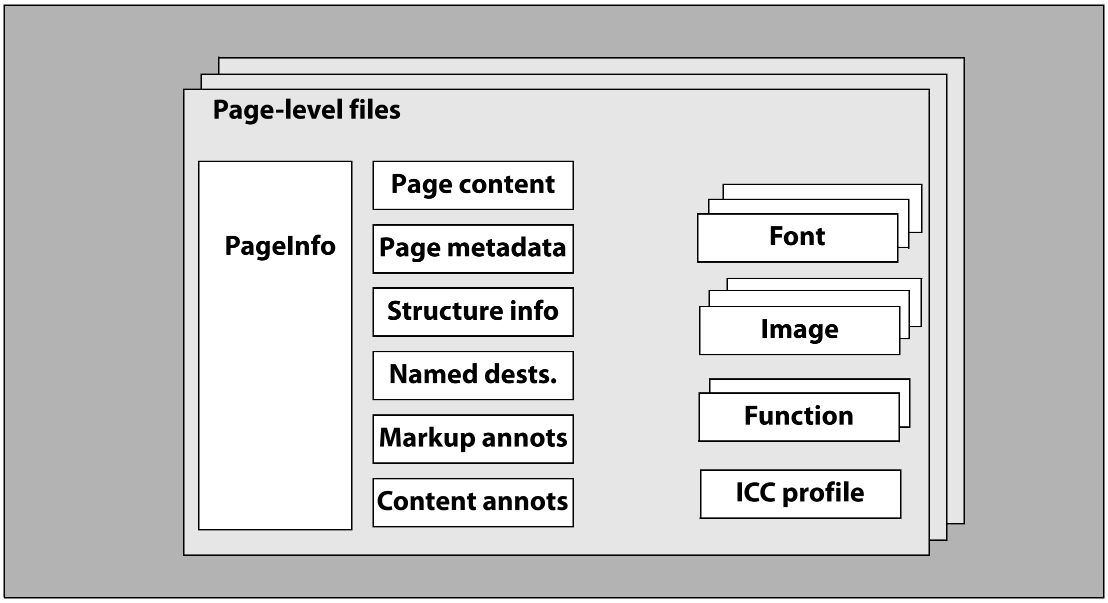
Color profiles

The Mars file format recommends the use of ICC color profiles to represent color spaces. ICC-based color spaces are based on cross-platform color profiles as defined by the International Color Consortium (ICC). There are various places where they can appear, but most are in the page files. The ICC profiles themselves are separate package files referred to by the SVG page content or resource file entries for a page in a Mars document. Color space information includes `Color` definitions and look-up tables. Color space information referenced from multiple pages should be added to the /color directory. Color space information referenced only within a particular page can be added to the page-level color dictionary, as described in Page-related files (below). For more information on color profiles, see "Mars extensions to SVG" on page 32 and *Developing Applications Using Mars*.

Page-related files

Page information for a Mars document is stored within the Mars document backbone. Each page within the document is represented by one or more files, as illustrated below.

Structure of page-related files



The following table describes the page-related files that can appear in a Mars document. The table specifies which files are required to be present in the document. A "Yes" in the Required column indicates that the file must be present in the Mars document. A "No" in this column indicates that the file is optional.

Purpose	Suggested file name	Required	Description of file contents
Page information	info.xml	Yes	Basic information about the page, such as the location of the page's content file, the page's size and orientation, and the location of the page's named-destination file, if any. "Page information" on page 21 provides more information.
Page content (SVG)	pg.svg	No	SVG that specifies the text and graphics to be applied to the page and that references images to be applied to the page. "Page content (SVG)" on page 22 provides more information.
Page metadata	metadata.xml	No	Metadata that applies only to the page described by this directory. "Metadata" on page 22 provides more information.
Logical structure and content tagging	struct.xml	No	Logical structure of the page. "Logical structure and content tagging" on page 23 provides more information.
Named destinations	dests.xml	No	Named destinations for the page described by this directory. "Named destinations" on page 23 provides more information.
Markup annotations	pg.svg.ann	No	Markup annotations for the page described by this directory. "Markup annotations" on page 23 provides more information.
Content annotations	pg.can	No	Page content annotations for the page described by this directory. "Content annotations" on page 24 provides more information.
Fonts, images, functions ICC profiles, and other resources	various	No	Resources used only on the page are stored in the page's directory. These additional files contain images, fonts, font descriptors, functions, shaders, patterns, color profiles, and others. "Fonts, images, functions, color profiles and other resources" on page 24 provides more information.

Page information

The Mars document backbone contains a series of XML `Page` elements, each of which points to a separate page information file. These page information files are conventionally stored in a directory named for the page (/Page/*page_number*) and are conventionally named *info.xml*.

Page information files include the following basic information about the page they represent:

- Page size and orientation
- A reference to the page content file

For more information, see *Developing Applications Using Mars*.

Page content (SVG)

The page content file is referenced from the page information file and is conventionally named *pg.svg*.

The text and graphics page content is represented using SVG. Some structure tree and marked content information is included on the page and interspersed with the SVG XML markup. Additional structure information, named destination information, resources, and annotations are stored in separate files associated with the page to which they apply.

Fonts, images, functions, ICC profiles and other resources used on a page are separate objects directly referenced by SVG page contents. Other resources are represented in the SVG page file. Some of these resources can be shared by multiple pages, and are stored in separate files in this case.

The page content file (conventionally called pg.svg) lists all the named resources that are used directly by the page. Names of resources may be referenced from the SVG page content file or from other page-related files in the Mars document. The names are resolved to specific image, font, color, or other information through the page file.

In SVG, some resources such as images are directly referenced by the SVG. Most other resources are declared in the `svg:defs` element of each page or annotation in which they are used. These definitions may be self-contained or may reference further information stored in a resource file elsewhere within the document package. Thus, multiple page or annotation SVG files can reference common resource definitions.

PDF allows resource definitions to directly include other resource definitions. For example, a shader can directly include a function definition. In Mars, each resource definition must be separate and each resource must be named. All references from one resource to another are by name or URI reference.

Resource names in Mars must follow XML identifier attribute rules and be unique within each XML file in the document package. Thus, different pages can use the same identifiers for their own resources. In the case of resources referenced from multiple pages (shared resources), the same rules apply. The shared-resource identifiers must be unique within the file that contains them. The shared-resource identifiers may be the same as identifiers defined within pages that reference the shared resource.

When creating Mars files, consideration should be given to performance when placing resources in files.

- Rendering of a page cannot begin until all locally-defined resources are read, parsed, and processed. Consequently, pages with large numbers of resources that are used for only isolated parts of the page will benefit from declaring those resources externally from the page.
- If a separate resource file is used, all resources within it must be read and parsed when any reference is made to that resource file. Consequently, a resource file referenced by a single page should contain only resources that are used on the page.
- Creating large numbers of small resource files will result in poor compression and a larger Zip package size, as compression tends to work better on moderate-sized files.

For more information, see "Mars SVG Support" on page 30.

Metadata

Metadata, which can appear at various levels in the Mars document, provides information such as author, modification date, and copyright status.

Level	Description
Document	Document level metadata must be placed in the /META-INF/metadata. xml file. For information on metadata at the document level, see "Document metadata" on page 26.
Page	Metadata associated with the entire page is contained in a metadata file. The implicit associations defined in container.xml associate these metadata files with the files they describe. (See "Implicit resource references" on page 27.) Conventionally, these files are named by appending "xmp" to the path of the file they describe. For example /page/*page_number*/svg.xml.xmp provides metadata for the primary SVG page content.
Page components	Metadata associated with other page-level files such as fonts, or images, appear in separate files related to the files that they describe. The implicit associations defined in container.xml associate these metadata files with the files they describe. (See "Implicit resource references" on page 27.) Conventionally, these files are named by appending "xmp" to the path of the file they describe. For example, the /font/MyFont.otf.xmp file provides metadata for the /font/MyFont.otf file.
SVG objects	Metadata that describes page-level graphic objects is represented using XML markup within the SVG page contents file.

Note: Files containing document-level metadata use the suffix "`.xml`" ("Document metadata" on page 26), while files containing page-level or lower metadata use the suffix "`.xmp`".

Logical structure and content tagging

A page structure file (struct.xml) is a logical description of the page contents. This logical description can include the organization of the document into chapters and sections or the identification of special elements such as figures, tables, and footnotes. The logical structure enables users to more easily navigate a file without knowing the producer's structural conventions. For more information, see *Developing Applications Using Mars*.

Named destinations

A named destination file (dests.xml) associates names with explicit destinations on the page. The names in such named destinations can be associated with actions (such as the GoTo action) associated with a bookmark or a content annotation. When the bookmark or a content annotation is activated, the name in the action is used to find the explicit destination defined in the named destination file.

Using named destinations in bookmarks or content annotations enables a document to be rearranged (pages removed or added) without affecting the behavior of the bookmark or content annotation. For more information on named destinations, see *Developing Applications Using Mars*.

Markup annotations

Markup annotations represent human-reader created marks on the page, often as part of a review and approval workflow. These are implicitly associated with the page and document and can be added without changing any other files in the document. By convention, the file containing markup annotations is named *pg.svg.ann*. For more information, see *Developing Applications Using Mars*.

Content annotations

Mars defines a set of annotations that appear effectively on a plane above the page. These annotations provide additional graphic and interactive content in the document. Content annotations are explicitly referenced by the page information file (info.xml), and represent material that is considered a standard part of page content. This group includes form fields and link annotations. By convention, the file containing content annotations is named *pg.can*. For more information, see *Developing Applications Using Mars*.

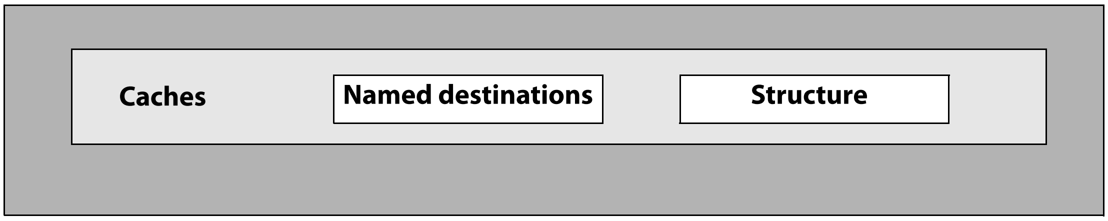
Fonts, images, functions, color profiles and other resources

The directory for a page can include the fonts, images, functions, ICC profiles and other resources used on the page described by the directory. For more information, see the file types described in "Document global files" on page 18.

Cache-related files

Most of the information in Mars files is organized for ease of creation and manipulation. To optimize interactive performance, some structures require indexing information to avoid long searches. These indexes are called caches and can be built as a post-processing step of document creation or manipulation. By using this cache approach, the basic organization of the information in the document is not complicated by the need for efficient search.

Structure of cache-related files



Caches are recreated or updated when a document structure is changed. Each cache is self-describing so that document modification tools can update caches without having to understand what is being cached.

Each cache is a file in the /cache directory in the document package. One part of a cache file contains data that represents document-wide associations. Another part contains information that can be used to recreate that data.

The following table describes the cache-related files that can appear in a Mars document. These files conventionally reside in the /cache directory. None of these files are required to be present in a Mars document, and the file names specified are recommended by convention.

Suggested file name	Description
names.xml	This file contains named destination mapping information. "Named destinations cache" on page 25 provides more information.
struct.xml	This file contains structure tree node mapping information. "Logical structure cache" on page 25 provides more information.

Named destinations cache

A named destination cache maps each named destination to the page containing the destination. The cache file lists all the named destination names and, for each, the page on which it is defined. This allows the location of each named destination in the document to be determined without having to read all of the named destination files in order to do so. For more information, see *Developing Applications Using Mars*.

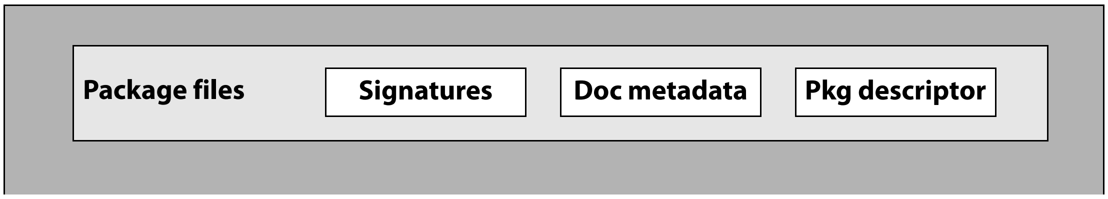
Logical structure cache

A logical structure tree cache (/cache/struct.xml) maps each interior structure tree node to the path for the page that contains information about it. For more information, see *Developing Applications Using Mars*.

Meta-information files

Mars package files are made of up all the files required by the corresponding Mars document. They consist of digital signatures, metadata, and the package descriptor files.

Structure of meta-information files



The following table describes the meta-information files that can appear in a Mars document. These files reside in the /META-INF directory.

The table specifies which files are required to be present in the document. A “Yes” in the column titled “Required” indicates the file must be present in the Mars document and must be named as specified. A “No” in this column indicates the file is optional.

File name	Required	Description
container.xml	Yes	This file specifies implicit associations between similarly named files. For example, this file defines the implicit relationship between files located in the same directory, where one file is named *filename* and the other is named *filename*.xmp.
metadata.xml	No	This file contains document level metadata, if applicable. “Document metadata” on page 26 provides more information.
signature.xml	No	This file contains signatures applied to the document. “Signatures” on page 26 provides more information.
encryption.xml	No	This file contains all encryption information on the contents of the container.
rights.xml	No	This file contains digital rights management information for trusted exchange of publications among rights holders, intermediaries, and users.

Document metadata

Document metadata (metadata.xml) is data that describes the characteristics or properties of a document. In order for multiple applications to be able to work effectively with metadata, there must be a common standard that they understand. The Extensible Metadata Platform (XMP) was designed to provide such a standard.

Note: Files containing document-level metadata use the suffix "`.xml`", while files containing page-level metadata ("Metadata" on page 22) use the suffix "`.xmp`".

In Mars, metadata can be applied to an entire package and to individual resources. Metadata about a resource may be stored separately or directly in the resource. The location of the resource metadata, when stored separately from the resource, is specified by a resource relationship rule in the container.xml file also located in the /META-INF directory. For more information, see *Developing Applications Using Mars*. For more information on the contents to be specified for metadata.xml, see "Universal Container Format" on page 221.

Signatures

A digital signature can be used to authenticate the identity of a user and the document's contents. It stores information about the signer and the state of the document when it was signed.

Form files

Form files contain data for other information associated with forms. These files are conventionally located in the /form directory. None of these files are required to be present in a Mars document, and the file names specified are recommended by convention.

File	Description
form_data.xfdf	This file must be created if the document contains Acrobat form data. This name is reserved for that purpose only, and no other file name may be used.
template.xml	The template.xml file must be created if the document is an XML form. This name is reserved for that purpose only.
config.xml	The config.xml file must be created if the document contains XML form configuration data. This name is reserved for that purpose only.
datasets.xml	The datasets.xml file must be created if the document contains XML form data. This name is reserved for that purpose only. The root element of this file is `ApplicationDatasets`.

XML conventions used in Mars documents

The XML files that comprise a Mars document must conform to specific encoding and namespace conventions.

Encoding

Every XML file in a Mars document must use UTF-8 character encoding.

Namespaces and namespace prefixes

All XML files must follow XML namespace rules. Generally, to promote a compact representation, files containing mostly Mars tags use the Mars namespace as the default and files containing mostly SVG markup will use the SVG namespace as the default.

The Mars namespace is `http://ns.adobe.com/pdf/2006`. All Mars tags are part of the Mars namespace. To reduce file size, Mars files generally declare the Mars namespace as the default namespace; however, some files used in Mars packages may declare other default namespaces. In such situations, the Mars namespace is declared and assigned a prefix, which is conventionally `pdf`.

SVG files are an example of non-Mars namespace files that are included in a Mars package. Such files primarily contain SVG tags but may also contain some Mars tags. An SVG document declares SVG as the default namespace. It also declares the Mars namespace, conventionally assigning it the namespace prefix of `pdf`.

Element order

Except as noted, the order of elements within the XML files is not important; elements can appear in any order.

The order of the attributes in an element is not significant (as defined by the XML standard).

The order in which elements appear in SVG files defines drawing order.

Required elements and attributes

To determine which elements and attributes are required, see “Mars Elements and Attributes” on page 54 in conjunction with the *PDF Reference, version 1.7*.

Explicit resource references

Most references between files in the document are explicit resource references via a URI. At the reference point, an element with a source attribute names the component via a URI. The URI must be either relative or root-based, but cannot specify a different scheme and must refer to an object within the same package as the referenced object. It is highly recommended that relative URIs be used whenever possible.

For information on explicit references to objects within a file, see “Explicit element references” on page 28.

Implicit resource references

Implicit associations are used when there is no specific reference from one Mars resource to a referenced resource. The existence of the resource under a particular name is used to form the association between the two resources. For example, annotations and metadata can be connected to particular Mars document resources by their existence in the Mars package under a specific name. An example of this would be when an `xmp` file is created to contain the corresponding XMP metadata associated with a particular image file. For the example files below, the file named `im_123.jpeg.xmp` contains XMP metadata associated with the file named `im_123.jpeg`:

```
/page/3/im_123.jpeg
/page/3/im_123.jpeg.xmp
```

Classes of implicit associations based on the file name are defined in the container.xml file in the `META-INF` directory. These relationships provide implied references between pairs of package-level files. Implicit relationships are used, because they allow addition or removal of information without modifying either the referencer, or the referenced component. This enables changes such as adding annotations, or metadata that do not affect document content.

For more information on implicit references, see "Universal Container Format" on page 221.

Unused files in a document

Files that are deleted from the package or that are no longer referenced, do not need to be physically deleted, because they are ignored. However, it is recommended to remove these files (and compact the ZIP archive) to avoid unnecessarily large document files.

Explicit element references

A URI can include a file reference and an id property that references another object. The file reference is similar to an explicit resource reference ("Explicit resource references" on page 27). If the file reference is omitted, the current resource contains the referenced resource. The `id` property identifies a specific element via its `id` attribute.

As shown in the following example, the `ref` attribute references the element in the current file that has an `id` attribute with the value `view23`.

```
<ExData>
  <View ref="#view23"/>
  ...
</ExData>

  ...
<View id="view23" ....>
... </View>
```

Here is an example of a reference to an element within a different file, where the file reference is relative. The `ref` attribute references the element that has an `id` attribute with the value `W3452` located in the pg.can file.

```
<Widget ref="pg.can#W3452"/>
```

Here is an example of a reference to a file, where the file reference is absolute.

```
<Widget ref="/page/23/pg.can#W3453"/>
```

In some cases, references are made within a single file using a simple name. There are specific rules for the given context stating how the reference is to be resolved. For example, in an SVG file, a color reference (below) references a color space definition identified as `cs-1`.

```
fill="rgb(246,177,153) icc-color(cs-1,80,20,20)"
```

Where the color profile definition appears elsewhere in the SVG file as follows:

```
<svg:color-profile name="cs-1" ... />
```

External object references

PDF supports the concept of referencing objects outside the document itself. The main cases of this are to external web pages (from URI actions, form submit, and web capture), and references to external files (usually for large images or fonts). For security reasons, the latter references are disabled by default in viewing applications.

External web links are supported only in elements and attributes that correspond to PDF properties that support such links. Only those element/attribute combinations will be interpreted as web links.

External files have a more general representation in PDF and therefore Mars. If a stream includes a `file_spec_dictionary` and no stream data (specified with the `src` attribute), then the stream contents is considered an external reference and the `file_spec_dictionary` is used to determine the external file name.

3 Mars SVG Support

This chapter describes how SVG is supported in Mars, and provides detail on the limitations of that support as well as SVG extensions provided in the Mars format. For more information, see *Developing Applications using Mars*.

SVG representation of page contents

Page content is represented in SVG format, which is a W3C standard for representing text and graphics. To meet the needs of representing and rendering documents in a compact and efficient way, Mars supports a subset of SVG and it adds SVG extensions to support additional capabilities defined by PDF. This subset is called the *SVG Fast Static Subset* (SVG/FSS). It corresponds to a subset of SVG Tiny 1.2 and a few features from SVG 1.1 and 1.2. To support the representation and display of graphic constructs supported by PDF documents, SVG/FSS also includes some private namespace extensions to SVG. In addition, Mars does not support some infrequently used PDF features.

Page contents characterize all of the text and graphic content and supplementary structure of the page. The basic SVG constructs are in the SVG namespace, and Mars-defined extensions are in the PDF namespace. It is possible to convert between the Mars SVG representation and the PDF page content representation of a page.

For information on the SVG features that Mars supports, see “SVG properties supported in SVG/FSS” on page 42. For information on the private namespace extensions to SVG, see “Mars extensions to SVG” on page 32.

Unsupported SVG content

Only supported SVG content should appear in Mars documents. In the event that unsupported SVG content does appear, user agents should ignore that content and continue processing the valid SVG/FSS content. Optionally, a warning could be displayed informing the user that unsupported content was encountered and ignored.

SVG graphic content and PDF resources

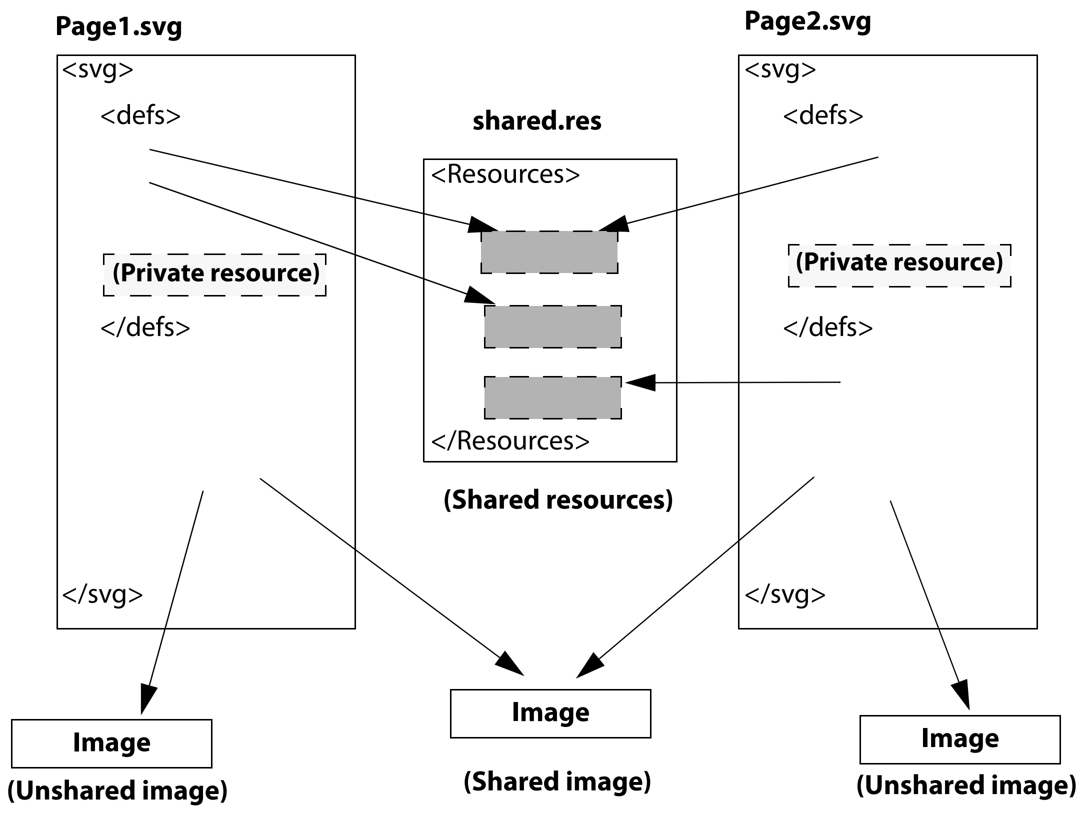
PDF graphic content references zero or more named resources. These resources are graphic states, fonts, color spaces, patterns, shaders, procedure sets, images, and graphic subroutines called *form XObjects*. PDF resources and their names are defined in a resources dictionary for the page or annotation in which they appear.

In SVG/FSS, some resources (for example, images) are directly referenced by the SVG. Most other resources are declared in the `<svg:defs>` element of each page or annotation in which they are used. These definitions may be self-contained or may reference further information stored in a resource file elsewhere within the document package. Thus, multiple page or annotation SVG files can reference common resource definitions. Finally, some resources in PDF, such as graphic states and procedure sets, are not required in Mars and do not appear.

PDF allows resource definitions to directly include other resource definitions. For example, a shader can directly include a function definition. In Mars, each resource definition must be separate and each resource must be named. All references from one resource to another are by name or URI reference.

Resource names in Mars must follow XML ID attribute rules and be unique within each XML file in the document package. Thus, different pages (in different SVG files) can use the same identifiers for their own resources. When resources are shared between pages in a common resource file, the same rules apply. The resource identifiers must be unique in the shared resource file; identifiers in the shared resource file can conflict with identifiers on referencing pages because they are separate XML files.

Mars resources



When placing resources in files, consideration should be given to performance.

Rendering of a page cannot begin until all locally defined resources are read, parsed, and processed. Therefore pages with large numbers of resources used only for isolated parts of a given page should declare those resources externally (from the page).

If a separate resource file is used, all resources within it must be read and parsed when any reference is made to that resource file. Consequently, when a page references a file, avoid having resources in the file that are not used on that page.

Creating large numbers of small resource files will result in poor compression and larger file size, because compression tends to work better on files of moderate size.

Mars extensions to SVG

Several PDF namespace extensions to SVG are defined to enable full-fidelity representation of PDF documents. These are shown in the following table.

Name	Description
Top level declarations	The root <svg:svg> element must contain additional attributes that indicate usage of specific imaging features on the page. These are required so that tools can optimize processing of the document without having to fully analyze the page contents. For more information, see Mars rendering information extensions. ● `pdf:UsesTransparency` ● `pdf:UsesSpotColors` ● `pdf:UsesOverPrint` ● `pdf:UsesKnockoutTransparencyGroups` ● `pdf:UsesIsolatedTransparencyGroupsNonRGBBlends` ● `pdf:UsesDefaultColorSpaces` ● `pdf:PageTransparencyGroupIsKnockout` ● `pdf:PageTransparencyGroupIsIsolated` ● `pdf:PageTransparencyGroupBlendColorspace` ● `svg:enable-background="new"` (required for transparency)
Transparency	The following extensions enable expression of the full PDF 1.4 transparency model. These attributes can appear anywhere the `%SVG.Opacity.attrib` set can appear. ● attribute `pdf:BlendMode {blend_mode}?` The `blend_mode` pattern specifies a blend mode name or array of names; see the schema for details. ● `attribute pdf:BlendingColorSpace {color_space}` The color_space pattern specifies this attribute type (see the schema, as well as the `CS` key from Table 7.13 and section 7.2.3 in the *PDF Reference, version 1.6*. This must be able to be represented as an attribute value). ● `attribute pdf:AlphaIsShape {boolean}?` ● `attribute pdf:Isolated {boolean}?` ● `attribute pdf:KnockOut {boolean}?`

Name	Description
Soft mask extensions	The following extensions specify soft masks, which provide the mask shape and mask opacity used for compositing an object. A soft mask defines values that can vary across different points on the page. The `pdf:softMask` element is a special variant of the SVG `mask` element. It contains a transparency group (a `<g>` element that has the special pdf transparency attributes defined, see above). The `pdf:softMask` element may have the following attributes: ● `attribute pdf:Subtype ( 'Alpha' | 'Luminosity' )` ● `attribute pdf:BackdropColorSpace {color_space}?` ● `attribute pdf:BackdropColor {array_of_number}?` ● `attribute pdf:TransferFunction {function}?` `BackdropColorSpace` and `BackdropColor` only appear if `Subtype` is `Luminosity`. The array size of `BackdropColor` is the number of components in the `BackdropColorSpace`. For images used as soft masks, the `image` element can be enclosed in a regular SVG `mask` element. The `image` element may carry an additional optional attribute: ● `attribute pdf:Matte {array_of_number}?` `pdf:Matte` specifies a color value. The size of the array is the number of components in the color space of the 'host' image to which the soft mask is applied

Name	Description
High end print support	Various `"pdf:"` elements and attributes for high end print workflows are supported (see Table 4.8 in the *PDF Reference, version 1.6*). These attributes can appear anywhere the `%SVG.Opacity.attrib` set can appear. • `attribute pdf:StrokeAdjustment {boolean}?` • `attribute pdf:Overprint {boolean}?` • `attribute pdf:OverprintPaint {boolean}?` • `attribute pdf:OverprintMode {integer}?` • `attribute pdf:UndercoverRemoval {function_name}?` • `function_name` (`"Default"` or a function reference in the `<svg:defs>` element or referenced file and id; see the schema and the `UCR` and `UCR2` keys in Table 4.8 of the *PDF Reference, version 1.5*). • `attribute pdf:FlatnessTolerance {number}?` • `attribute pdf:SmoothnessTolerance {number}?` • `attribute pdf:BlackGeneration { function_name }?` • `attribute pdf:HalfTone { resource_name }?` • `resource_name` (`"Default"`, the name of a halftone resource defined in the `<svg:defs>` element, or the referenced file and identifier). • `attribute pdf:TransferFunction { function_names }?` • `function_names` = (`"Identity"` or 1 of 4 function names that reference function resources in the `<svg:defs>` element or elsewhere).
Image interpolation	• `<svg:image pdf:Interpolate="true"` …> attribute is required on the `<svg:image>` element to express the `/Interpolate` property on a PDF image. • `attribute pdf:Interpolate {boolean}?` • `attribute pdf:Decode{color_space_map}`
Text extensions	• `attribute pdf:Transform` on `<svg:tspan>` • `element <svg:altGlyph>` on `<svg:text>` and `<svg:tspan>` • `attribute pdf:Matrix` on `<svg:font>` • `attribute pdf:BBox` on `<svg:glyph>` • `attribute pdf:HWidth` on `<svg:glyph>` • attribute `pdf:IsShapeOnly` on `<svg:glyph>` For more information, see PDF text extensions.

Name	**Description**

Color space extensions	SVG 1.2 style colors are partially supported and include extensions that can specify a color space.
	● Colors that have n colorants and are part of a color space with an ICC profile have this format:
	`rgb(r,g,b) icc-color`
	`   (<colorspace-name>, n1, n2, n3 … nn)`
	The first r,g,b triplet is a device-color approximation of the ICC color. `colorspace-name` refers to a color-profile element that is defined in the document's `defs` section.
	● Colors that are part of a color space without an ICC profile have this format:
	`rgb(r,g,b) device-color`
	`   (<colorspace-name>, n1, n2, n3 … nn)`
	`colorspace-name` refers to a color space definition element that is defined in the `defs` section, or it can be one of two special names: `DeviceCMYK` or `DeviceGray`
	For a description of this extension, see *Developing Applications using Mars*. For limitations on color expressions, see "Color values" on page 43.

Smooth shading extensions	For more information, see PDF smooth shading extensions.
Function extensions	For more information, see PDF function extensions.
Blending mode marker	SVG/FSS page content that uses blending modes other than `Normal` must specify `enable-background="new"` on the outermost `<svg>` element; otherwise, all blending modes will be converted to `Normal`. This is necessary for SVG compatibility. The `enable-background="new"` setting tells the SVG user agent to be prepared for the possibility of upcoming transparency groups.
Font descriptor for non-embedded fonts	An additional `pdf` namespace element is defined that is part of the `<svg:font-face>` element. This additional element, `<pdf:font-information>`, has an `xlink:href` attribute referring to a font descriptor file in the document package. When a font substitution is required, the information in this file is used to select a substitution font and process it consistently with the original font. For more information, see *Developing Applications using Mars*.
Structure information	For information on SVG extensions that represent structure information, see "Mars marked content extensions" on page 35.

Mars marked content extensions

Mars defines extension attributes and elements that associate non-rendered information with groups of SVG content. This marking mechanism allows applications or plug-ins to later identify or access the content and to associate metadata with it. Mars introduces several extensions to SVG for structure information

The attributes are described in the following table.

Name	Description
`pdf:Mark`	Specifies a tag that identifies a marked content group. The value of the attribute is not unique and can be shared within the same page and across multiple pages. The value can be any string that is meaningful to the applications consuming the marked content. This attribute can appear in the SVG `<g>` element.
`pdf:Props`	Specifies an XPath expression that specifies the location and name of a user-properties attribute class. For example: `pdf:Props="resources/properties.xml#AProp"` User-property attribute classes include the attribute definition `Owner="UserProperties"`. This attribute can appear in the SVG `<g>` element or in the `pdf:Point` element.

The elements are described in the following table.

Name	Description
`pdf:Props/Custom`	Specifies private data that describes the non-rendered properties (see Example 3.1). The `pdf:Props` element contains a child `Custom` element that uses the default dictionaries to describe private data. This element can appear in the SVG `<g>` element and the `pdf:Point` element.
`pdf:Props/Membership`	Specifies the optional content groups to which the `<g>` element belongs (see Example 3.2). The `pdf:Props` element optionally contains an `id` attribute that uniquely identifies the properties it defines. Subsequent marks in the SVG file can then reference the object rather than re-creating it. Specifies an array of membership dictionaries with the characteristics described in the section “Optional content membership dictionary” in “Mars Elements and Attributes” on page 54.
`pdf:Point`	Specifies a single marked-content point in the content stream. This element can appear in the SVG `<g>` element.

The following example specifies several private non-rendered properties associated with the content in the SVG `<g>` element. Inline properties and application datasets use the same conventions to describe private data, except the inline properties data resides in the `Custom` element while the application datasets data resides in the `Private` element. See “Adding private data to application datasets” in the guide *Developing Applications Using Mars*.

Example 3.1 ***Inline pdf:Props***

```
<SVG>
```

```
      <g pdf:Mark="Artifact">
         <Props inline="true" xmlns="http://ns.adobe.com/pdf/2006">
            <Custom>
               <Type Value="Layout" type="name"/>
               <BBox type="array">
                  <Int Value="89"/>
                  <Int Value="78"/>
                  <Int Value="515"/>
                  <Int Value="580"/>
               </BBox>
            </Custom>
         </Props>
         ...
      </g>
   </SVG>
```

Note: Future versions of Mars may define a new namespace for defining private properties.

The `Props` element can also provide information that associates content in a group with an optional content group, as shown in Example 3.2.

Example 3.2 ***Optional content membership***

```
<g pdf:Mark="OC">
   <Props xml:id="prop_1" xmlns="http://ns.adobe.com/pdf/2006">
      <Membership>
         <Groups>
            <Group ref="/backbone.xml#3"/>
         </Groups>
      </Membership>
   </Props>
</g>
```

Mars rendering information extensions

Several extension attributes can appear on the root `<svg>` element of a page or annotation appearance. These attributes make it possible to accelerate the rendering of the page by enabling the proper configuration of rendering code prior to reading and parsing the `<svg>` content for the page. If one of these attributes is missing or incorrectly set, the page must still render correctly, but may take significantly longer to appear. The attributes are described in the following table.

Name	Description
`pdf:UsesTransparency`	Determines whether there is transparency on the page and, optionally, in the annotations. Transparency is deemed to be present if these conditions are met: ● An `SMask` is found ● A non-`Normal` blend mode is found ● `/CA` or `/ca` has a value of less than `1` You can also examine image resources and see if they have a `/SMaskInData` with a value greater than `1`, which can only occur with JPEG20000 encoded images.
`pdf:UsesSpotColors`	Determines whether spot colors are used on the page and, optionally, in annotations on the page.
`pdf:UsesOverPrint`	Determines whether overprint is used on the page and, optionally, in annotations. Overprint is detected if these conditions are met: ● There are separation or DeviceN color spaces, and `/OP` or `/op` is `true` in some extended `gstate` ● `OPM` is seen with a value of `1`, and `/OP` or `/op` is `true`
`pdf:UsesKnockoutTransparencyGroups`	Determines whether knockout transparency groups are used.
`pdf:UsesIsolatedTransparencyGroupsNonRGBBlends`	Determines whether isolated transparency groups with or without non-RGB blending spaces are used. (If no blending space is specified a CMYK one is assumed).
`pdf:UsesDefaultColorSpaces`	Determines whether the page has default color spaces. These would be `/DefaultRGB`, `/DefaultCMYK` and `/DefaultGray` in the `/Colorspace` part of the `/Resources` dictionary.

`pdf:PageTransparencyGroupIsKnockout`
`pdf:PageTransparencyGroupIsIsolated`
`pdf:PageTransparencyGroupBlendColorspace`
	If there is a page level transparency group (this would be a `/Group` entry in the page dictionary with subtype `/Transparency`), these attributes indicate if it is a knockout group, if it is isolated, and if it is a blending color space.

PDF smooth shading extensions

PDF smooth shading extends SVG's gradient capabilities to draw more complex shading of objects. A shading object defines the properties of the gradient fill. This is a complex "paint" that determines the type of color transition the shading pattern produces when painted across an area.

Named shading objects must appear in the `<svg:defs>` element or in a separate resource file. References to locally-defined shading objects must appear following their declaration in the SVG file.

There are seven PDF shading types represented by seven PDF namespace elements. Their RELAX NG schema patterns are named as follows:

- `type_1_shading_dictionary`
- `type_2_shading_dictionary`
- `type_3_shading_dictionary`
- `type_4_shading_dictionary`
- `type_5_shading_dictionary`
- `type_6_shading_dictionary`
- `type_7_shading_dictionary`

The following are examples of shaders in SVG content. The first example corresponds to the axial shader. The second example corresponds to the radial shader. Both shaders are described in section 4.6.3 in the *PDF Reference.*

Example 3.3 ***Type 2 (axial) shading***

```
<pdf:AxialShader id="sh-52734" ColorSpace="DeviceCMYK" AntiAlias="false"
  Matrix="1 0 0 1 0 0">
  <Axis x0="0" y0="0" x1="1" y1="0"/>
  <Domain t0="0" t1="1"/>
  <Extend BeyondStart="true" BeyondEnd="true"/>
  <pdf:ColorFunctions>
    <pdf:SampledFunction Domain="0 1" Range="0 1 0 1 0 1 0 1" Size="256"
      BitsPerSample="8" Order="linear" Encode="0 255"
      Decode="0 1 0 1 0 1 0 1" src="/res/func-60.f0"/>
  </pdf:ColorFunctions>
</pdf:AxialShader>
```

Example 3.4 ***Type 3 (radial) shading***

```
<pdf:RadialShader Coords="25 25 0 50 50 50" id="sh-52073"
  ColorSpace="DeviceCMYK" AntiAlias="false" mtx="1 0 0 1 0 0">
  <Domain t0="0" t1="1"/>
  <Extend BeyondStart="false" BeyondEnd="false"/>
  <pdf:ColorFunctions>
    <pdf:SampledFunction Domain="0 1" Range="0 1 0 1 0 1" Size="255"
      BitsPerSample="8" Order="linear" Encode="0 254" Decode="0 1 0 1 0 1">
      <pdf:File Name="/res/func-0.f0"/>
    </pdf:SampledFunction>
  </pdf:ColorFunctions>
</pdf:RadialShader>
```

For more information, see the schemas and Mars Elements and Attributes.

PDF function extensions

PDF functions are represented by four `pdf` namespace elements. The `pdf:Function` element can contain other functions, as allowed. Their RELAX NG schema elements can be traced through the `function_dictionary` rule.

Here is an example of a Type 1 (sampled) function, which is described in section 3.9.1 of the *PDF Reference, version 1.7.*

```
<pdf:SampledFunction Domain="0 1" Range="0 1 0 1 0 1 0 1" Size="255"
  BitsPerSample="8" Order="linear" Encode="0 254" Decode="0 1 0 1 0 1 0 1"
  src="/res/func-0.f0"/>
```

Here is an example of a Type 2 (interpolation) function, which is described in section 3.9.2 of the *PDF Reference, version 1.7*.

```
<InterpolatedFunction C0="0.448 0.174 0.51 0.303" C1="0.184 0 0.016 0"
  Exponent="0.5" Domain="0 1"/>
```

Here is the general form of a Type 3 (stitching function), which is described in section 3.9.3 of the *PDF Reference, version 1.7*.

```
<pdf:StitchingFunction Domain="x1 x2 ... x(2 * m)" Range="x1 x2 ... x(2 * n)"
  Bounds="x1 x2 ... x(k - 1)" Encode="x1 x2 ... x(2 * k)">
  <pdf:StitchingFunctions>
    … (k) functions ...
  </pdf:StitchingFunctions>
</pdf:StitchingFunction>
```

Here is the general form of a Type 4 (PostScript calculator) function, which is described in section 3.9.4 of the *PDF Reference, version 1.7*.

```
<pdf:PostscriptFunction Domain="x1 x2 ... x(2 * m)" Range="x1 x2 ... x(2 * n)"
  xlink:href=”/function/ps_func12.ps” />
```

For more information, see the schemas and Mars Elements and Attributes.

PDF halftone extensions

Halftones are references from a `pdf:HalfTone` attribute whose value references a halftone resource. The resource may be in a child element of the `<svg:defs>` element or may be a resource in a separate resource file. Their RELAX NG schema elements can be traced through the `halftone_resource` rule.

For more information, see the schemas and Mars Elements and Attributes.

PDF text extensions

Text transformation

To match the expressiveness of PDF, a `pdf:Transform` attribute is allowed on `<svg:tspan>` elements. The attribute value is the same as in other uses of the `<svg:transform>` attribute defined in SVG:

```
attribute pdf:Transform {…} on <svg:tspan>
```

More compact alternate-glyphs expression

SVG already includes a mechanism for specifying alternate glyphs for Unicode text. Such glyph specifications are more common in PDF and require a textually shorter expression. In `<svg:text>` and `<svg:tspan>` elements, it is defined as follows:

```
attribute altGlyphs { text }
```

The value of `altGlyphs` is a space-separated list of strings (URI references or glyph identifiers) with an optional parenthetical pair of numbers: `(#chars[, #glyphs])` that can precede the string. For example, `"(3,2) 125 126"` means that three characters are to be consumed from the Unicode string and replaced with the next two glyphs (glyph identifiers `125` and `126`).

If a given glyph is not preceded by an `(n,m)` expression that includes it, the default is `(1,1)`.

If only `"#chars"` is provided, the default for `#glyphs` is `1`.

If `n=0`, no Unicode characters are consumed, but the following glyph(s) is/are rendered.

If `n>0` and `m=0`, then the next *n* Unicode characters are rendered without glyph substitution.

In situations where the glyph attribute is incorrectly formed (for example, you run out of glyphs or characters), the glyph attribute is treated as an unsupported attribute and the viewer must render the Unicode characters as if the glyph attribute were not present.

The following example shows SVG text with an `altGlyphs` attribute and no Unicode characters:

```
<text transform="…" font-size="12" font-family="F1" fill="…"
fill-rule="evenodd">
  <tspan x="0 12.0769 16.4434 27.5957" altGlyphs="(0,4) 4 14 16 12"/>
</text>
```

The following example shows SVG text with an `altGlyphs` attribute with Unicode characters (assume the same parent `<text>` element as above):

```
<tspan x="0 12.0769 16.4434 27.5957" altGlyphs="(3,4) 4 14 16 12"> abcdefg
</tspan>
```

In this example, `a`, `b`, and `c` are replaced with glyphs `4`, `14`, `16`, and `12`. Then `d`, `e`, `f`, and `g` are rendered based on the Unicode map for font `F1`.

Representation of Type 3 fonts as Mars SVG

In Mars, Type 3 fonts are expressed as SVG fonts, with the addition of some PDF-specific extensions to the SVG <font> and <glyph> elements.

Attribute pdf:Matrix on svg:font element

The `pdf:Matrix` attribute specifies a font matrix (a list of six numbers) that maps glyph space to text space. The format of this attribute is as follows:

```
  Attribute pdf:Matrix { coordinate_map_array }
```

For more information, see the `FontMatrix` entry in Table 5.9 of the *PDF Reference, version 1.7*.

Attribute pdf:BBox on the svg:glyph element

The `pdf:BBox` attribute is a list of four numbers that specify the glyph bounding box, the smallest rectangle needed to enclose the shape of the glyph. The format of this attribute is as follows:

```
  pdf:Matrix { rectangle_blank_sep }
```

For more information, see the `FontBBox` entry in Table 5.9 of the *PDF Reference, version 1.7*.

Attribute pdf:HWidth on the svg:glyph element

The `pdf:HWidth` attribute is a number that specifies the horizontal width of the glyph. The format of this attribute is as follows:

```
pdf:HWidth { number}
```

The width is the same as the glyph's width in the font's `Widths` table or the horizontal width specified using the d0 or d1 operators (see Table 5.10 of the *PDF Reference, version 1.7*).

Attribute pdf:IsShapeOnly on the svg:glyph element

If the `pdf:IsShapeOnly` attribute has a value of `true`, this glyph description specifies only the glyph's shape, not the glyph's color. The format of this attribute is as follows:

```
pdf:IsShapeOnly { Boolean }
```

Setting the `pdf:IsShapeOnly` property to true has the same effect as declaring a glyph using the `d1` operator (see Table 5.10 of the *PDF Reference, version 1.7*).

Other requirements

For a Type 3 font expressed as Mars SVG, the first glyph in the font must be named "`.notdef`". The `.notdef` glyph is used as a substitute in the event that a text element using the font calls for a glyph that is not defined for the font.

Requirements for defining glyph shapes is more restrictive for Type 3 fonts expressed as Mars SVG than for normal SVG fonts. For a Type 3 font, glyph shapes can be defined only as content enclosed by a `glyph` element; the `d` attribute cannot be used. For normal SVG fonts, glyph shapes can also be defined by using the `glyph` element's `d` attribute.

Here is a segment of an example Type 3 Font expressed as Mars SVG:

```
<font pdf:Matrix="0.00999 0 0 0.00999 0 0">
  <glyph glyph-name=".notdef" pdf:BBox="0 250 0 0" pdf:HWidth="250"
    pdf:IsShapeOnly="true"/>
  <glyph glyph-name="70" pdf:BBox="7.324 50.586 0 64.16" pdf:HWidth="53.9063"
    pdf:IsShapeOnly="true">
    <path fill="rgb(0,0,0) device-color(DeviceGray,0)"
      d="M7.324,0v64.16h43.262v-7.519H15.82v-20.02h30.078v-7.519H15.82V0"/>
  </glyph>
  ( … other glyph descriptions go here …)
</font>
```

PDF image extensions

For image interpolation, the `pdf:Interpolate` attribute is required on the `svg:image` element to express the PDF `/Interpolate` property on a PDF image. The format of this attribute is as follows:

```
attribute pdf:Interpolate {boolean}?
```

For example:

```
<svg:image pdf:Interpolate=”true” …>
```

SVG properties supported in SVG/FSS

Mars supports a set of SVG expressions, called *Mars SVG Fast Static Subset* (FSS), that include most of the properties from the *Scalable Vector Graphics (SVG) Tiny 1.2 Specification* and some of the properties from *Scalable Vector Graphics (SVG) 1.1 Specification* and *Scalable Vector Graphics (SVG) 1.2 Specification* that are not already part of SVG Tiny v1.2. The following table is a generalized list of SVG properties that Mars does not fully support. For a definitive list of SVG specifications that Mars does not fully support, see "Tiny 1.2 expressions excluded from SVG/FSS" on page 44, "SVG 1.1 expressions excluded from SVG/FSS" on page 50, "SVG 1.2 expressions excluded from SVG/FSS" on page 51 .

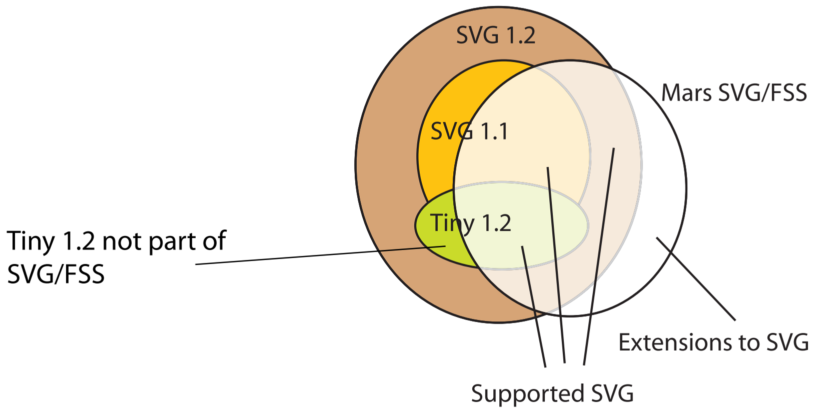
Topic	**Exclusions and deviations from SVG Tiny 1.2, SVG 1.2, and SVG 1.1 specifications**
Animation and multimedia	All markup related to animation and multimedia is not supported.
Arbitrary order of definitions and use	All non-URI references for symbols, fonts, glyphs, gradients, patterns, clipping paths and masks must be local in the same file and must resolve to an element earlier in the document tree, so that the file can be processed in a single pass.
Bidirectional text properties	Properties for controlling bidirectional text: `direction` and `unicode-bidi`.
Color interpolation	`color-interpolation` attribute is supported only for SVG-style gradients. In other situations, use PDF extension properties for blending.
Color values	Float functional percentages in the form `rgb(R%, G%, B%)` are not supported. For information on extensions to color value expressions, see "Color space extensions" on page 35.
CSS	All CSS-related markup (`stylesheet PI, <svg:style>` element, `style` attribute) is not supported and must be treated as unsupported markup (in other words, ignored) by Mars user agents.
CSS units	CSS units or percentages in length values are not allowed in conforming content and are treated as unsupported markup (ignored) by Mars user agents. However, the `width` and `height` attributes in the outermost `<svg:svg>` element can use CSS units or percentages. In the context of Mars, other features in Mars will override the `width` and `height` attributes in the outermost `<svg:svg>` element, with the result that these two attributes are ignored when imported into a Mars user agent).
Define-before-use	`<svg:use>` elements can only refer to `<svg:symbol>` elements within the same file.
Filter effects	None of the markup related to filter effects is supported. Examples of these properties are the `svg:filter` element and its properties `color-interpolation-filters`, `flood-color`, and `lighting-color`.
Font selection properties	Properties that have to do with font selection: `font`, `font-size-adjust`, `font-stretch`, `font-style`, `font-variant`, `font-weight`.

Topic	Exclusions and deviations from SVG Tiny 1.2, SVG 1.2, and SVG 1.1 specifications
`foreignObject` restrictions	In conformant content, all use of `<svg:foreignObject>` must be enclosed within an `<svg:switch>` with fallback rendering for SVG and Mars user agents that do not support the extension required by the `<svg:foreignObject>`. Mars user agents are required to support a small number of rendering extensions. (Smooth shading is one of them. There might be additional extensions, such as ones used for tiling patterns).
Interactivity and scripts	All markup related to interactivity and scripting (for example, `<svg:script>`, event attributes, the `pointer-events` property, `<svg:view>` element) is not supported.
Markers	Markers are not allowed in conforming content and must be treated as unsupported markup (ignored) by Mars user agents.
`tref`, `textPath`	The `<svg:tref>` and `<svg:textPath>` elements are not supported.
Text layout: • Ligatures • Line layout • Glyph substitutions • Text alignment • Kerning • BIDI groups • Text area	SVG properties related to spacing and layout are not supported. Text nodes which combine BIDI groups are not supported. Mars user agents must not perform ligature formation beyond those specified within the content via `<svg:altGlyph>` and `<svg:altGlyphs>`. SVG font definitions must not define any glyphs with more than one Unicode character within the `unicode` attribute. No glyph substitutions are to be performed.
Rendering hints	It is acceptable for Mars user agents to ignore SVG's rendering hints (`colorrendering`, `image-rendering`, `shape-rendering`, `text-rendering`).
Symbol resolution	All URI references for symbols, fonts, glyphs, gradients, patterns, clipping paths and masks must be local and must resolve to an element earlier in the document tree.
Text alignment	Properties that have to do with text: `alignment-baseline`, `baseline-shift`, `dominant-baseline`, `glyph-orientation-horizontal`, `glyph-orientation-vertical`, `text-anchor` (all content must conform to `text-anchor="start"`), `text-decoration`, `writing-mode`.
Other	`solid-color`

Tiny 1.2 expressions excluded from SVG/FSS

This section describes the parts of *Scalable Vector Graphics (SVG) Tiny 1.2 Specification* (21 July 2006) (www.w3.org/TR/SVGMobile12) that SVG/FSS does not support. The following figure shows the constituents of SVG/FSS, which include extensions in the pdf namespace ("Mars extensions to SVG" on page 32) and parts of Tiny 1.2, SVG 1.1, and SVG 1.2. Rather than list the supported Tiny 1.2 expressions, this section describes the unsupported expressions, which correspond to the area in unobscured green.

Components of the Mars SVG/FSS



The following table identifies sections from the *Scalable Vector Graphics (SVG) Tiny 1.2 Specification* (www.w3.org/TR/SVGMobile12) that describe expressions that Mars does not fully support. If a section is not listed, Mars supports the expressions it contains. Notice that only the main instances of descriptions of SVG features are called out. Many times, subsequent sections make short mentions of earlier features; these are not listed as variances.

Section in SVG Tiny 1.2 Specification	Exclusion or limitation
1.1 About SVG	Interaction, dynamic, scripting, and animation are not included.
2.1.6 Scriptable	Scripting is not supported.
2.2.4 Animation	Animation is not supported.
2.3 Options for using SVG in Web pages	Use in Mars is not in the context of web pages; therefore, this section is not relevant.
4.1 Basic Data Types	Numbers are not limited to 4 decimal places; range is not limited to +/-32767. `scientific-number` is not supported.

Section in SVG Tiny 1.2 Specification	**Exclusion or limitation**
5.1.2 The 'svg' element	SVG content that conforms to the Fast Static Subset that uses blending modes other than Normal must specify enable-background=" new" on the outermost `<svg>` element; otherwise, all blending modes will be converted to Normal. (This is necessary for SVG compatibility. The enable-background="new" setting tells the SVG user agent to be prepared for the possibility of upcoming transparency groups.) Nested `svg` elements are not supported. Unsupported attributes: `snapshotTime`, `playbackOrder`, `timelineBegin`, `contentScriptType`, `focusable`, `preserveAspectRatio`, `zoomAndPan`, `viewBox`. (The PDF media box and crop box subsume the function of `viewBox`.) Navigation attributes are not supported.
5.2 Grouping: the 'g' element	The `g` element may contain the `pdf:Mark` attribute. See "Mars marked content extensions" on page 35.
5.3 The 'defs' element	Definitions must be inside the `defs` element and definitions must appear before they are referenced.
5.4 The 'discard' element	The `discard` element is not supported.
5.5 The 'desc' and 'title' elements	The `desc` and `title` elements are not supported.
5.7 The 'image' element	For image interpolation, the `pdf:Interpolate` attribute is required on the `svg:image` element to express the `/Interpolate` property on a PDF image, as shown in this example: `<svg:image pdf:Interpolate="true" ...>` For information on supported image file types, see "SVG representation of page contents" on page 30 and "PDF image extensions" on page 42
5.8 Conditional processing	The `switch` element and its attributes are not supported.
5.9 External Resources	External resources (external `ResourcesRequired` attributes) are not supported.
5.9.3 The 'prefetch' element	The `prefetch` element is not supported.
6 Styling	All CSS-related markup (stylesheet PI, the `svg:style` element, the `style` attribute) is not supported. CSS Units: CSS units or percentages on length values is not supported.
7.14 Geographic Coordinate Systems	Coordinate system metadata is not supported.
7.15 The svg:transform attribute	Coordinate system metadata is not supported.

Section in SVG Tiny 1.2 Specification	**Exclusion or limitation**
10.2 Characters and their corresponding glyphs	These properties are replaced by the Mars `altGlyph` extension. See “PDF text extensions” on page 40.
10.5 The 'tspan' element	This element is replaced by the pdf namespace extension: `attribute pdf:transform {…}` on the `svg:tspan` element. See “PDF text extensions” on page 40.
10.6 Text layout	The PDF model is a precision text layout model. Consequently, all SVG features related to automatic text layout are not supported. This includes most of the features in this section.
10.8 Alignment properties	The text-anchor property is not supported (all content must conform to the SVG expression `text-anchor="start"`)
10.9 Font selection properties	The `font-size-adjust` and `font` properties are not supported. The `font-weight`, `font-style`, `font-variant`, `font-size`, and `font-stretch` properties are supported. Only a single font-family name is allowed in a font-family property value. If a list is provided, the content is not conforming and the processor must use only the first value in the list.
10.11 Text in an area	The `textArea` and `tbreak` elements are not supported. The `line-increment`, `text-align`, and `display-align` properties are not supported.
10.12 Editable Text Fields	Editable text fields are not supported.
11.2 Specifying paint	Paint servers URIs can refer to pdf-namespace shaders and pattern definitions.
11.5 Non-Scaling Stroke	Vector-effect is not supported.
11.7 The 'viewport-fill' Property	Viewport-fill is not supported.
11.8 The 'viewport-fill-opacity' Property	Viewport-fill-opacity is not supported.
11.9 Controlling visibility and rendering	Visibility control and the `display` property are not supported.

11.12 Object and group opacity	Here are extensions that can appear in the same places that the opacity attribute can appear:
	`attribute pdf:BlendMode {blend_mode}?`
	`attribute pdf:blendingColorSpace {color_space}`
	`attribute pdf:alphaIsShape {boolean}?`
	`attribute pdf:isolated {boolean}?`
	`attribute pdf:knockOut {boolean}?`

Section in SVG Tiny 1.2 Specification	**Exclusion or limitation**

11.13.1 Syntax for color values	Float functional percentage `rgb()` is not supported.
	SVG 1.2 style colors are partially supported and include some extensions.
	Colors are represented in fill and stroke attributes as follows:
	● Colors that have n colorants and are part of a color space with an ICC profile have this format:
	`rgb(r, g, b)`
	`icc-color`
	`(<colorspace-name>, n1, n2, n3 … nn)`
	The first r,g,b triplet is a device-color approximation of the ICC color.
	`colorspace-name` refers to a color-profile element that is defined in the document's `defs` section.
	● Colors that are part of a color space without an ICC profile have this format:
	`rgb(r, g, b)`
	`device-color`
	`(<colorspace-name>, n1, n2, n3 … nn)`
	`colorspace-name` refers to a color space definition element that is defined in the `defs` section, or it can be one of two special names: `DeviceCMYK` or `DeviceGray`.

11.14.1 System Paint Servers	System Paint Servers are not supported.
11.14.3 The SVG 'color' property	Not supported. Content must specify the color directly by using the `fill` and `stroke` properties.
11.16 Gradients	In addition to the facilities defined in SVG Tiny 1.1, Mars includes patterns from SVG 1.1 and several extensions to support PDF features around shading and gradients.
12 Multimedia	SVG multimedia features are not supported.
13 Interactivity	Interactivity is not supported. Such unsupported interacitivity properties include any event and the `focusable`, `nav-next`, and `focus-highlight` properties.
14.1.4 Reference Restrictions	Any references to files outside the document package are not supported. See "SVG page content references to other document package objects" on page 52.
14.1.5 IRI reference attributes	Any xlink attributes other than `href` are ignored.
14.1.6 Externally referenced documents	Any references to files outside the document package are not supported. See "SVG page content references to other document package objects" on page 52.
14.2 Links out of SVG content: the 'a' element	The `a` element is not supported. Instead, use link annotations to represent external links.

Section in SVG Tiny 1.2 Specification	**Exclusion or limitation**
14.3 Linking into SVG content: IRI fragments and SVG views	Addressing SVG content with fragment identifiers is not supported.
15 Scripting	Scripting is not supported. Features not supported include the script element, XML Events, handler element, listener element, Event handling
16 Animation	Animation is not supported. Features not supported include `SMIL`, `animate` element and its attributes, `animation-related` attributes on other elements, attributes `begin`, `dur`, `end`, `min`, `max`, `restat`, `repeatCount`, `repeatDur`, `fill`, elements `discard` `animateMotion`, `animateColor`, `animateTransform`, and `mpath`.

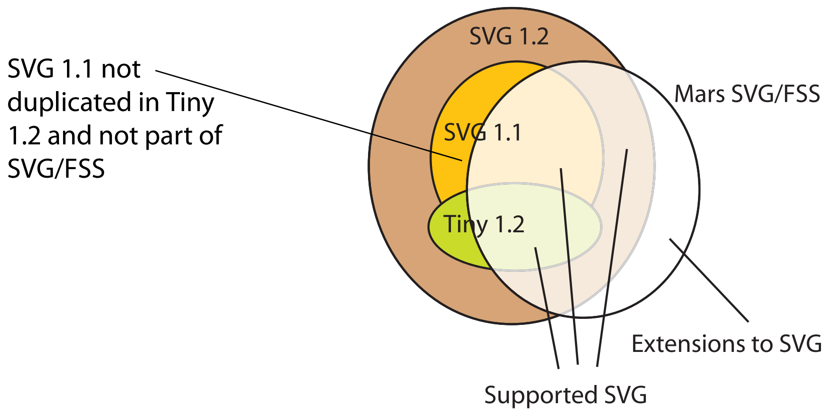
17.8.2 The 'font-face' element	This usage is standard within SVG with the interpretation of the `svg:font-face-uri` as referencing a file contained in the document package.
	Initial declaration of font in the `defs` part of the SVG:
	`<svg:font-face font-family="F1">`
	`<svg:font-face-src>`
	`<svg:font-face-uri xlink:href=`
	`  "…URL of font within package…"/>`
	`</svg:font-face-src>`
	`</svg:font-face>`
	A reference to the font looks like this code:
	`<svg:text font-family="F1">hello</svg:text>`

18 Metadata	Metadata and the `metadata` element are not supported. Instead, XMP metadata should be associated with the page.
19.2.1 The 'foreignObject' element	The `foreignObject` element is not supported.

SVG 1.1 expressions excluded from SVG/FSS

This section describes the parts of the *Scalable Vector Graphics (SVG) 1.1 Specification* that are not duplicated in Tiny 1.2 and that are not included in SVG/FSS. The following figure shows the constituents of SVG/FSS, which include extensions in the pdf namespace ("Mars extensions to SVG" on page 32) and parts of Tiny 1.2, SVG 1.1, and SVG 1.2. The SVG 1.1 expressions that this section excludes correspond to the area in unobscured orange.

Components of the Mars SVG/FSS



The following table identifies sections from the SVG 1.1 Specification that are not part of Tiny 1.2 and that are excluded from SVG/FSS. If a section is not listed and has not been excluded in "Tiny 1.2 expressions excluded from SVG/FSS" on page 44, the expressions it contains are included in SVG/FSS.

Section in SVG 1.1 Specification	Exclusion or limitation
5.5 The 'symbol' element	Symbol Resolution: All non-URI references for symbols, fonts, glyphs, gradients, patterns, clipping paths, and masks must be local and must resolve to an element earlier in the document tree. **Note:** See 5.6 "The 'use' element" (below).
5.6 The 'use' element	Instantiate a template defined by <symbol>. The `use` elements can refer only to <svg:symbol> elements within the same file. In general, names must be defined textually before they are used so that the file can be processed in a single pass.
12.3 Color profile descriptions	Defines managed color profiles. See also Section 12 "Color Spaces" on page 84. <color-profile> element and color-profile attribute on <image> elements are included but not CSS styling.
13.3 Patterns	<pattern> element
14.3.5 Establishing a new clipping	Path <clipPath> element creates a new clipping path.

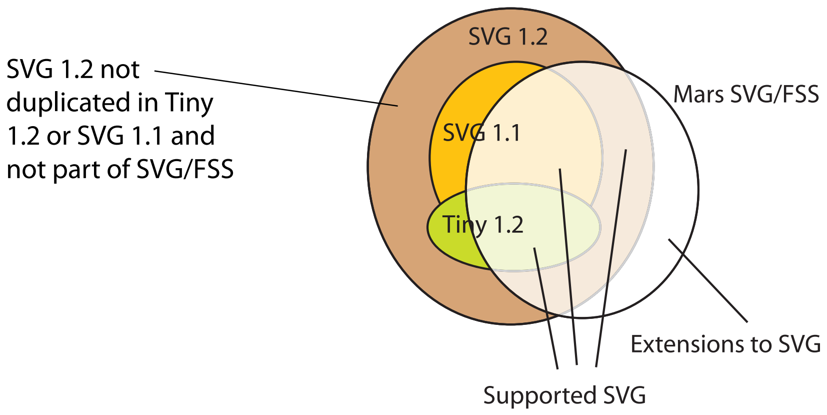
Section in SVG 1.1 Specification	Exclusion or limitation
14.4 Masking	The SVG <mask> element is supported. PDF soft masks are mapped to <pdf:softMask> elements.
20.7 The 'hkern' and 'vkern' elements	The `vkern` and `hkern` elements are supported.

SVG 1.2 expressions excluded from SVG/FSS

This section describes the parts of the *Scalable Vector Graphics (SVG) 1.2 Specification* that are not duplicated in Tiny 1.2 or SVG 1.1 and that are included in SVG/FSS. The following figure shows the constituents of SVG/FSS, which include extensions in the pdf namespace ("Mars extensions to SVG" on page 32) and parts of Tiny 1.2, SVG 1.1, and SVG 1.2. The SVG 1.1 expressions this section excludes correspond to the area in unobscured orange.

The following figure shows the constituents of SVG/FSS, which include extensions in the pdf namespace ("Mars extensions to SVG" on page 32) and parts of Tiny 1.2, SVG 1.1, and SVG 1.2. The SVG 1.2 expressions this section excludes correspond to the area in unobscured brown.

Components of the Mars SVG/FSS


SVG 1.2 not duplicated in Tiny 1.2 or SVG 1.1 and not part of SVG/FSS
SVG 1.2
SVG 1.1
Tiny 1.2
Mars SVG/FSS
Extensions to SVG
Supported SVG

The following table identifies sections from the SVG 1.2 Specification that are not part of Tiny 1.2 or SVG 1.1 and that are excluded from SVG/FSS. If a section is not listed and has not been excluded in "Tiny 1.2 expressions excluded from SVG/FSS" on page 44 or "SVG 1.1 expressions excluded from SVG/FSS" on page 50, the expressions it contains are included in SVG/FSS.

Section in SVG 1.2 Specification	Exclusion or limitation
11.5 Using device colors	Use of device-color to support additional PDF color spaces is not supported.

PDF features not supported in the Mars SVG subset

The following table lists the PDF page content features that are either partially supported or not supported in the Mars SVG subset:

Name	Description
`bx` / `ex` operator	In PDF, compatibility sections wrapped in `bx` and `ex` are content that can be ignored if it is not recognized. Such a construct is not supported in Mars (although the SVG switch construct is similar). When converting PDF to Mars, the `bx` / `ex` operators are removed and the enclosed content is retained.
CID fonts	CID fonts are not supported in Mars. When converting PDF to Mars, CID fonts are converted to OpenType and cannot be converted back to CID fonts. The representation of characters on the pages may be changed as part of this conversion.
Pass-through PostScript®	PostScript XObjects are not supported. When Acrobat converts from PDF to Mars, the presence of pass-through PostScript will result in user warnings and removal of the PostScript content.

SVG page content references to other document package objects

SVG page content references other objects in the document package. References outside the document package are not allowed. The following table shows the references from SVG to other document page objects:

SVG page content source	Document package target
`<color-profile xlink:href="target"`	The ICC profile file.
`<text font-family="F1">`	`<font-face font-family="F1">`

`<font-face font-family="F2"`
`               font-weight="400">`
`<font-face-src>`
`  <font-face-uri xlink:href="../fonts/14.otf"/>`
`  <font-face-name name="Myriad Pro"/>`
`  </font-face-src>`
`</font-face>`
	File `/fonts/14.otf` within the package.

`<image xlink:href="myimage.png">`	The image file within the package. It can be of type .jpeg, .png, .jp2, or .jbig2.
`<image color-profile="cs-2"`	`<pdf:pdfColorProfile name="cs-2">`

`<path fill="url(#sh-247)" d="M-0.318,0.0343h1 […] >`
`<path fill="url(/shaders/common.res#sh-247)" d="M-0.318,0.0343h1 […] >`
	`<defs><FunctionShader ID="sh-247" …`
	in file `/shaders/common.res:`
	`<FunctionShader ID="sh-247" …`

For information on explicit references from non-SVG Mars expressions to SVG objects, see “Explicit resource references” on page 27

4 Mars Elements and Attributes

3D activation dictionary

three_d_activation_dictionary

```
three_d_activation_dictionary =
  attribute ActivateOn {  "PageOpen" |  "PageVisible" |  "Explicit" }?
  & attribute ArtworkOn {  "Instantiate" |  "Live" }?
  & attribute DeactivateOn {  "PageClose" |  "PageInvisible" |  "Explicit"
  }?
  & attribute ArtworkOff {  "Instantiate" |  "Uninstantiated" |  "Live" }?
  & attribute ToolbarDisplay { boolean }?
  & attribute UIDisplay { boolean }?
```

Description

An activation dictionary determines how the state of the annotation and its associated artwork can change. For more information, see section 9.5.1, the `3DA` entry in Table 9.33, and Table 9.34 in the *PDF Reference*.

Element or attribute	Description	Default Value
ActivateOn	See the `A` entry in Table 9.34 in the *PDF Reference*.	
ArtworkOn	See the `AIS` entry in Table 9.34 in the *PDF Reference*.	
DeactivateOn	See the `D` entry in Table 9.34 in the *PDF Reference*.	
ArtworkOff	See the `DIS` entry in Table 9.34 in the *PDF Reference*.	
ToolbarDisplay	See the `TB` entry in Table 9.34 in the *PDF Reference*.	
UIDisplay	See the `NP` entry in Table 9.34 in the *PDF Reference*.	

3D animation style dictionary

animation_style_dictionary

```
animation_style_dictionary =
  attribute Style { pdf_text_name }?
  & attribute PlayCount { integer }?
  & attribute TimeMultiplier { number }?
```

Description

Specifies the preferred method that viewer applications should use to determine timeline scaling that is applied to a keyframe animation. It can also specify that keyframe animations be played repeatedly. For more information, see *3D Animation Style Dictionaries* in section 9.5.2 in the *PDF Reference*.

Element or attribute	Description	Default Value
Style	See the `Subtype` entry in Table 9.36 in the *PDF Reference*.	None
PlayCount	See the `PC` entry in Table 9.36 in the *PDF Reference*.	0
TimeMultiplier	See the `TM` entry in Table 9.36 in the *PDF Reference*.	1

3D annotation

three_d_annotation_dictionary

```
three_d_annotation_dictionary =
  element Artwork { stream_or_reference_dictionary }
  & element Initial { initial_3d_view }?
  & element Activation { three_d_activation_dictionary }?
  & attribute Interactive { boolean }?
  & element ViewBox { rectangle }?
three_d_annotation_dictionary &= common_annotation_dictionary
```

stream_or_reference_dictionary

```
stream_or_reference_dictionary =
  element Stream { three_d_stream_dictionary }
stream_or_reference_dictionary |=
  element Shared { three_d_reference_dictionary }
```

Description

3D annotations are the means by which 3D artwork is represented in a Mars document. For more information, see section 9.5.1 in the *PDF Reference*.

Element or attribute	Description	Default Value
Artwork	See the `3DD` entry in Table 9.33 in the *PDF Reference*.	
Initial	See the `3DV` entry in Table 9.33 in the *PDF Reference*.	The default view in the 3D stream object specified by `Artwork`.
Activation	See the `3DA` entry in Table 9.33 in the *PDF Reference*.	

Interactive	See the `3DI` entry in Table 9.33 in the *PDF Reference.*	true
ViewBox	See the `3DB` entry in Table 9.33 in the *PDF Reference.*	The annotation's `rect` entry, expressed in the target coordinate system.

3D cross section dictionary

cross_section_dictionary

```
cross_section_dictionary =
  element Center { center_of_rotation_array }?
  & element Orientation { orientation_array }?
  & attribute PlaneOpacity { number }?
  & element PlaneColor { auxiliary_color_array }?
  & attribute IntersectionVisibility { boolean }?
  & element IntersectionColor { auxiliary_color_array }?
```

center_of_rotation_array

```
center_of_rotation_array =
  attribute X { number }
  & attribute Y { number }
  & attribute Z { number }
```

orientation_array

```
orientation_array =
  attribute XRot { number }
  & attribute YRot { number }
  & attribute ZRot { number }
```

Description

Specifies how a portion of the 3D artwork is clipped for the purpose of showing artwork cross sections. For more information, see *3D Cross Section Dictionaries* in section 9.5.3 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
Center	See the `C` entry in Table 9.46 in the *PDF Reference.*	
Orientation	See the `O` entry in Table 9.46 in the *PDF Reference.*	
PlaneOpacity	See the `PO` entry in Table 9.46 in the *PDF Reference.*	0.5
PlaneColor	See the `PC` entry in Table 9.46 in the *PDF Reference.*	

IntersectionVisibility	See the `IV` entry in Table 9.46 in the *PDF Reference*.	false
IntersectionColor	See the `IC` entry in Table 9.46 in the *PDF Reference*.	

3D node dictionaries

array_of_three_d_node_dictionaries

```
array_of_three_d_node_dictionaries = element Node { three_d_node_dictionary
}*
```

three_d_node_dictionary

```
three_d_node_dictionary =
   attribute Name { pdf_text_string }
   & attribute Opacity { number }?
   & attribute Visible { boolean }?
   & attribute Matrix { three_d_transformation_matrix }?
```

three_d_transformation_matrix

```
three_d_transformation_matrix =  ( list { number number number number number
number number number number number number number } )
```

Description

A 3D node dictionary, specified by a 3D view, that specifies particular areas of 3D artwork and the opacity and visibility with which individual nodes are displayed. For more information, see *3D Node Dictionaries* in section 9.5.3 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
Name	See the `N` entry in Table 9.47 in the *PDF Reference*.	
Opacity	See the `O` entry in Table 9.47 in the *PDF Reference*.	
Visible	See the `V` entry in Table 9.47 in the *PDF Reference*.	
Transform	See the `M` entry in Table 9.47 in the *PDF Reference*.	

3D reference dictionary

three_d_reference_dictionary

```
three_d_reference_dictionary =
  element Stream { attribute ref { pdf_reference } }
```

Description

A run-time instance of the 3D artwork that can be shared by multiple annotations. For more information, see the *3D Reference Dictionaries* in section 9.5.2 in the *PDF Reference*.

Element or attribute	Description	Default Value
Stream	See the `3DD` entry in Table 9.36 in the *PDF Reference*.	

3D stream

three_d_stream_dictionary

```
three_d_stream_dictionary =
  element Views { array_of_view_dictionary }?
  & element Default { initial_3d_view }?
  & element Resources { view_parameter_name_tree* }?
three_d_stream_dictionary &=
  element OnInstantiate { javascript_action_js }?
  & element AnimationStyle { animation_style_dictionary }?
  & attribute src { string }
  & common_stream_dictionary
```

Description

The specification of 3D artwork is contained in a 3D stream. 3D stream dictionaries can provide a set of predefined views of the artwork, as well as a default view. They can also provide scripts and resources for providing customized behaviors or presentations. For more information, see section 9.5.2 in the *PDF Reference*.

Element or attribute	Description	Default Value
Views	See the `VA` entry in Table 9.35 in the *PDF Reference*.	
Default	See the `DV` entry in Table 9.35 in the *PDF Reference*.	
AnimationStyle	See the `AN` entry in Table 9.35 in the *PDF Reference*.	
src		

3D view

three_d_view_dictionary

```
three_d_view_dictionary =
  attribute ExternalName { pdf_text_string }
  & attribute InternalName { pdf_text_string }?
  & attribute CameraTransformation { pdf_text_name }?
  & attribute Camera2World { array_of_number }?
  & element U3DPathName { string_or_u3d_path }?
  & attribute CenterOrbitDistance { number }?
  & element Projection { projection_dictionary }?
  & element Overlay { form_dictionary }?
  & element Background { background_dictionary }?
  & element RenderMode { render_mode_dictionary }?
  & element LightingScheme { lighting_scheme_dictionary }?
  & element Sections { array_of_cross_section_dictionaries }?
  & element Nodes { array_of_three_d_node_dictionaries }?
  & attribute NodeStateRestore { boolean }?
```

lighting_scheme_dictionary

```
lighting_scheme_dictionary =
  attribute Subtype { pdf_text_name }
```

array_of_cross_section_dictionaries

```
array_of_cross_section_dictionaries = element Section {
cross_section_dictionary }*
```

Description

A 3D view specifies parameters to be applied to the virtual camera associated with a 3D annotation. The names used in the lighting scheme dictionary are the schemes listed in Table 9.45 in the *PDF Reference*. For more information, see section 9.5.3 in the *PDF Reference*.

Element or attribute	Description	Default Value
ExternalName	See the `XN` entry in Table 9.37 in the *PDF Reference*.	
InternalName	See the `IN` entry in Table 9.37 in the *PDF Reference*.	
CameraTransformation	See the `MS` entry in Table 9.37 in the *PDF Reference*.	
Camera2World	See the `C2W` entry in Table 9.37 in the *PDF Reference*.	
U3DPathName	See the `U3DPath` entry in Table 9.37 in the *PDF Reference*.	

CenterOrbitDistance	See the `CO` entry in Table 9.37 in the *PDF Reference.*	
Projection	See the `P` entry in Table 9.37 in the *PDF Reference.*	A projection dictionary where the value of `Subtype` is `Perspective`, the value of `FieldOfView` is `90`, and all other entries take their default values.
Overlay	See the `O` entry in Table 9.37 in the *PDF Reference.*	
Background	See the `BG` entry in Table 9.37 in the *PDF Reference.*	
RenderMode	See the `RM` entry in Table 9.37 in the *PDF Reference.*	
LightingScheme	See the `LS` entry in Table 9.37 in the *PDF Reference.*	
Sections	See the `SA` entry in Table 9.37 in the *PDF Reference.*	
Nodes	See the `NA` entry in Table 9.37 in the *PDF Reference.*	
NodeStateRestore	See the `NR` entry in Table 9.37 in the *PDF Reference.*	false

3D view

three_d_view

```
three_d_view =
  attribute ref { pdf_reference }
three_d_view |=
  attribute Index { number }
three_d_view |=
  attribute Name { pdf_text_string }
three_d_view |=
  attribute Key { name }
  & attribute Key_enc { token }?
```

Description

The view to use in a go-to-3D-view action. For more information, see the `V` entry in Table 8.66 in the *PDF Reference.*

Element or attribute	Description	Default Value
Index	An integer specifying an index into the `Views` array in the 3D stream.	
Name	A name that indicates the first (`F`), last (`L`), next (`N`), previous (`P`), or default (`D`) entries in the `Views` array.	
Key	A text string matching the `InternalName` entry in one of the views in the `Views` array.	

3D view dictionaries

array_of_view_dictionary

```
array_of_view_dictionary = element View { three_d_view_dictionary }*
```

Description

An array of 3D view dictionaries, each of which specifies a named preset view of the 3D artwork. For more information, see the `VA` entry in Table 9.35 in the *PDF Reference*.

ASCII string object

pdf_ascii_string

```
pdf_ascii_string = string
```

Description

An ASCII string represented as a sequence of bytes. For more information, see section 3.2.3 in the *PDF Reference*.

Additional-actions dictionary

page_additional_actions_dictionary

```
page_additional_actions_dictionary =
   element OnPageOpen { action_dictionary }?
   & element OnPageClose { action_dictionary }?
```

action_dictionary

```
action_dictionary =
  element GoTo { goto_action_dictionary }
action_dictionary |=
  element GoToR { gotor_action_dictionary }
action_dictionary |=
  element GoToE { gotoe_action_dictionary }
action_dictionary |=
  element Launch { launch_action_dictionary }
action_dictionary |=
  element Thread { thread_action_dictionary }
action_dictionary |=
  element URI { uri_action_dictionary }
action_dictionary |=
  element Sound { sound_action_dictionary }
action_dictionary |=
  element Movie { movie_action_dictionary }
action_dictionary |=
  element Hide { hide_action_dictionary }
action_dictionary |=
  element Named { named_action_dictionary }
action_dictionary |=
  element SubmitForm { submitform_action_dictionary }
action_dictionary |=
  element ResetForm { resetform_action_dictionary }
action_dictionary |=
  element ImportData { importdata_action_dictionary }
action_dictionary |=
  element JavaScript { javascript_action_dictionary }
action_dictionary |=
  element SetOCGState { setocgstate_action_dictionary }
action_dictionary |=
  element Rendition { rendition_action_dictionary }
action_dictionary |=
  element Trans { trans_action_dictionary }
action_dictionary |=
  element Goto3DView { goto_3d_view_action_dictionary }
action_dictionary |=
  element CustomAction { custom_action_dictionary }
```

Description

An annotation, page object, interactive form field, or document catalog dictionary may include an entry that specifies an additional-actions dictionary that extends the set of events that can trigger the execution of an action.

For more information describing additional-actions dictionaries, see section 8.5 in the *PDF Reference*. For descriptions of each of the action types, see Table 8.44 in the *PDF Reference*.

Element or attribute	Description	Default Value

OnPageOpen	See the `O` entry in Table 8.45 in the *PDF Reference*.
OnPageClose	See the `C` entry in Table 8.45 in the *PDF Reference*.

Alternate image dictionary

alternate_image_dictionary

```
alternate_image_dictionary =
   element Image { image_dictionary }
   & attribute DefaultForPrinting { boolean }?
   & element ContentGroup { oc_group_or_membership_dictionary }?
```

Description

A dictionary specifying a variant representation of a base image. Each alternate image dictionary contains an image XObject for one variant and specifies its properties. For more information, see `Alternate Images` in section 4.8.4 in the *PDF Reference*.

Element or attribute	Description	Default Value
ContentGroup	See the `OC` entry in Table 7.11 in the *PDF Reference*.	

Alternate images

array_of_alternate_image_dictionary

```
array_of_alternate_image_dictionary = element Alternate {
alternate_image_dictionary }*
```

Description

An array of alternate image dictionaries that are variant representations for a base image. For more information, see `Alternate Images` in section 4.8.4 in the *PDF Reference*.

Alternate presentations

alternate_presentations_name_tree

```
alternate_presentations_name_tree =
   element Presentation { attribute Key { text }
   & alternate_presentation_dictionary }
```

alternate_presentation_dictionary

```
alternate_presentation_dictionary =
  element StartResource { attribute _enc { token }?
    & pdf_byte_string }
```

Description

A name tree mapping name strings to alternate presentations. For more information, see section 9.4, as well as the `AlternatePresentations` entry in Table 3.28 in the *PDF Reference*.

Alternate text descriptions

multi_language_text_array

```
multi_language_text_array = element Val { pdf_text_string }*
```

Description

An array of `Val` elements, each of which provides alternate text descriptions for a media clip section in case in cannot be played.

Annotation additional actions dictionary

annotation_additional_actions_dictionary

```
annotation_additional_actions_dictionary =
  element OnCursorEnter { action_dictionary }?
  & element OnCursorExit { action_dictionary }?
  & element OnMouseDown { action_dictionary }?
  & element OnMouseUp { action_dictionary }?
  & element OnPageOpen { action_dictionary }?
  & element OnPageClose { action_dictionary }?
  & element OnPageVisible { action_dictionary }?
  & element OnPageInvisible { action_dictionary }?
```

form_field_additional_actions_dictionary

```
form_field_additional_actions_dictionary =
  element OnKeystroke { javascript_action_dictionary }?
  & element BeforeFormat { javascript_action_dictionary }?
  & element OnValueChange { javascript_action_dictionary }?
  & element OnCalculate { javascript_action_dictionary }?
```

widget_annotation_additional_actions_dictionary

```
widget_annotation_additional_actions_dictionary =
  element OnFocusIn { action_dictionary }?
  & element OnFocusOut { action_dictionary }?
widget_annotation_additional_actions_dictionary &=
annotation_additional_actions_dictionary
widget_annotation_additional_actions_dictionary &=
form_field_additional_actions_dictionary
```

Description

An annotation, page object, or interactive form field may include an entry that specifies an additional-actions dictionary that extends the set of events that can trigger the execution of an action. For more information, see section 8.5.2 in the *PDF Reference*.

Example

```
<Actions>
  <OnCursorExit>
    <URI URI="http://www.adobe.com"></URI>
  </OnCursorExit>
  <OnCursorEnter>
    <Hide>
      <Target>
        <Field Pathname="Check Box2"/>
      </Target>
    </Hide>
  </OnCursorEnter>
  <BeforeFormat><Script>MyNamedJavaScript(3, 0);</Script></BeforeFormat>
</Actions>
```

Element or attribute	Description	Default Value
OnCursorEnter	See the `E` entry in Table 8.44 in the *PDF Reference*.	
OnCursorExit	See the `X` entry in Table 8.44 in the *PDF Reference*.	
OnMouseDown	See the `D` entry in Table 8.44 in the *PDF Reference*.	
OnMouseUp	See the `U` entry in Table 8.44 in the *PDF Reference*.	
OnFocusIn	See the `Fo` entry in Table 8.44 in the *PDF Reference*.	
OnFocusOut	See the `Bl` entry in Table 8.44 in the *PDF Reference*.	
OnPageOpen	See the `PO` entry in Table 8.44 in the *PDF Reference*.	

OnPageClose	See the `PC` entry in Table 8.44 in the *PDF Reference*.
OnPageVisible	See the `PV` entry in Table 8.44 in the *PDF Reference*.
OnPageInvisible	See the `PI` entry in Table 8.44 in the *PDF Reference*.
OnKeystroke	See the `K` entry in Table 8.46 in the *PDF Reference*.
BeforeFormat	See the `F` entry in Table 8.46 in the *PDF Reference*.
OnValueChange	See the `V` entry in Table 8.46 in the *PDF Reference*.
OnCalculate	See the `C` entry in Table 8.46 in the *PDF Reference*.

Annotation appearance name tree

annotation_appearances_file

```
annotation_appearances_file =
  element AnnotationAppearances { annotation_appearance_name_tree* }
```

annotation_appearance_name_tree

```
annotation_appearance_name_tree =
  element Graphic { attribute Key { text }
    & form_dictionary }
annotation_appearance_name_tree |=
  element NoGraphic { attribute Key { text } }
```

Description

A name tree mapping name strings to annotation appearance streams. For more information, see section 8.4.4 as well as the `AP` entry in Table 3.28 in the *PDF Reference*.

Element or attribute	Description	Default Value
Key		
Graphic		
NoGraphic		

Annotation dictionary

common_annotation_dictionary

```
common_annotation_dictionary =
  element Contents { pdf_text_string }?
  & attribute Rect { rectangle_list }
  & attribute Name { pdf_text_string }?
  & attribute ModDate { date_or_string }?
  & attribute Flags {  ( list {  "Invisible"? "Hidden"? "Print"? "NoZoom"?
  "NoRotate"? "NoView"? "ReadOnly"? "Locked"? "ToggleNoView"?
  "LockedContents"? } ) }?
  & element Appearance { appearance_dictionary }?
  & attribute AppearanceState { name }?
  & attribute AppearanceState_enc { token }?
  & attribute Color { rgb_color_array }?
  & element ContentGroup { oc_group_or_membership_dictionary }?
  & element Border { border_style_dictionary }?
```

date_or_string

```
date_or_string = pdf_text_string
```

Description

An annotation associated with the given page. Annotations are split into two groups: markup annotations and content annotations.

Content annotations are explicitly referenced by the page information file, and represent material that is considered a standard part of page content. This group includes form fields and link annotations.

Markup annotations represent human-reader created marks on the page as part of some review and approval cycle. These are implicitly associated with the page and document and can be added without changing any other files in the document.

Annotations are placed in separate files, grouped by page. For each page, there can be two annotation files, one containing markup annotations such as sticky-notes and text comments, and another containing content-oriented annotations such as form field widgets and hyperlinks.

For more information, see section 8.4.1 in the *PDF Reference*.

Element or attribute	Description	Default Value
Contents	See the `Contents` entry in Table 8.15 in the *PDF Reference*.	
Rect	See the `Rect` entry in Table 8.15 in the *PDF Reference*.	
Name	See the `NM` entry in Table 8.15 in the *PDF Reference*.	
ModDate	See the `M` entry in Table 8.15 in the *PDF Reference*.	

Flags	See the `F` entry in Table 8.15 in the *PDF Reference*.
Appearance	See the `AP` entry in Table 8.15 in the *PDF Reference*.
AppearanceState	See the `AS` entry in Table 8.15 in the *PDF Reference*.
Color	See the `C` entry in Table 8.15 in the *PDF Reference*.
ContentGroup	See the `OC` entry in Table 8.15 in the *PDF Reference*.
Border	See the `Border` entry in Table 8.15 in the *PDF Reference*.

Annotation rectangle

rectangle_list

```
rectangle_list =  ( list { number "," number "," number "," number } )
```

Description

The annotation rectangle, defining the location of the annotation on the page in default user space units.

Annotations, content (file)

array_of_content_annotation_file

```
array_of_content_annotation_file =
  element Annotations { array_of_content_annotation_dictionary }
```

array_of_content_annotation_dictionary

```
array_of_content_annotation_dictionary = content_annotation_dictionary*
```

content_annotation_dictionary

```
content_annotation_dictionary =
  element Link { link_annotation_dictionary }
content_annotation_dictionary |=
  element Movie { movie_annotation_dictionary }
content_annotation_dictionary |=
  element Screen { screen_annotation_dictionary }
content_annotation_dictionary |=
  element Widget { widget_annotation_dictionary }
content_annotation_dictionary |=
  element PrinterMark { printers_mark_annotation_dictionary }
content_annotation_dictionary |=
  element TrapNet { trap_network_annotation_dictionary }
content_annotation_dictionary |=
  element Watermark { watermark_annotation_dictionary }
content_annotation_dictionary |=
  element A3D { three_d_annotation_dictionary }
```

Description

Content annotations are explicitly referenced by the page information file, and represent material that is considered a standard part of page content. This group includes form fields and link annotations.

PDF defines a set of annotations that appear effectively on a plane above the page. These annotations provide additional graphic and interactive content in the document. In PDF documents, annotations for a page are stored in the page dictionary for that page. In Mars documents, annotations are also stored with the page, but in as many as two separate files associated with the page.

PDF documents do not distinguish between annotation types, and treat them all as a group. Mars documents segregate the annotations into two groups. Content annotations are considered part of page content, and are the following annotations:

- Link Annotation (hyperlinks)
- Movie Annotation (embedded multimedia movie using platform player)
- Screen Annotation (area where media clips may be played)
- Widget Annotation (interactive form fields)
- PrinterMark Annotation (registration target, color bar, cut mark, etc.)
- Trap Network Annotation (define trapping characteristics for a page)
- Watermark Annotation (background graphics)
- 3D Annotation (3D objects)

Annotations, markup (file)

array_of_markup_annotation_file

```
array_of_markup_annotation_file =
  element Annotations { array_of_markup_annotation_dictionary }
```

array_of_markup_annotation_dictionary

```
array_of_markup_annotation_dictionary = markup_annotation_dictionary*
```

markup_annotation_dictionary

```
markup_annotation_dictionary =
  element Text { text_annotation_dictionary }
markup_annotation_dictionary |=
  element FreeText { free_text_annotation_dictionary }
markup_annotation_dictionary |=
  element Line { line_annotation_dictionary }
markup_annotation_dictionary |=
  element Square { shape_annotation_dictionary }
markup_annotation_dictionary |=
  element Circle { shape_annotation_dictionary }
markup_annotation_dictionary |=
  element Polygon { poly_annotation_dictionary }
markup_annotation_dictionary |=
  element PolyLine { poly_annotation_dictionary }
markup_annotation_dictionary |=
  element Highlight { text_markup_annotation_dictionary }
markup_annotation_dictionary |=
  element Underline { text_markup_annotation_dictionary }
markup_annotation_dictionary |=
  element Squiggly { text_markup_annotation_dictionary }
markup_annotation_dictionary |=
  element StrikeOut { text_markup_annotation_dictionary }
markup_annotation_dictionary |=
  element Caret { caret_annotation_dictionary }
markup_annotation_dictionary |=
  element Stamp { stamp_annotation_dictionary }
markup_annotation_dictionary |=
  element Ink { ink_annotation_dictionary }
markup_annotation_dictionary |=
  element FileAttachment { file_attachment_annotation_dictionary }
markup_annotation_dictionary |=
  element Sound { sound_annotation_dictionary }
markup_annotation_dictionary |=
  element Redact { redaction_annotation_dictionary }
markup_annotation_dictionary |=
  element Custom { custom_annotation_dictionary }
```

Description

Markup annotations represent human reader-created marks on the page as part of some review and approval cycle. These are implicitly associated with the page and document, and can be added without changing any other files in the document.

Markup annotations are stored in a separate file in the document package. Unlike content annotations, the markup annotations file is not directly named by the document XML. Instead, the package relationship class mechanism is used. There is an annotation relationship defined between a file with name `N` and a file with name `N.ann`. Markup annotations for a page whose SVG page contents file is typically named `pg.svg` would appear in `pg.svg.ann`.

When a document is opened, to answer the question of whether markup annotations are present on a page, the presence of the file whose root matches the page content file and whose suffix is `".ann"` in the document package must be checked.

Markup annotations include the following annotation types:

- Text
- FreeText
- Line
- Square
- Circle
- Polygon
- Polyline
- Highlight
- Underline
- Squiggly
- StrikeOut
- Caret
- Stamp
- Ink
- Popup
- FileAttachment
- Sound
- Redaction

Appearance characteristics dictionary

appearance_characteristics_dictionary

```
appearance_characteristics_dictionary =
  attribute Rotate { integer }?
  & attribute BorderColor { array_of_number }?
  & attribute BackgroundColor { array_of_number }?
  & attribute Caption { pdf_text_string }?
  & attribute RolloverCaption { pdf_text_string }?
  & attribute DownCaption { pdf_text_string }?
  & element Icon { form_dictionary }?
  & element RolloverIcon { form_dictionary }?
  & element DownIcon { form_dictionary }?
  & element IconFit { icon_fit_dictionary }?
  & attribute TextPosition { integer }?
appearance_characteristics_dictionary &= variable_text_dictionary_fields
```

Description

A dictionary containing additional information for constructing the annotation's appearance stream. For more information, see *Widget Annotations* in section 8.4.5 in the *PDF Reference*.

Element or attribute	Description	Default Value
Rotate	See the `R` entry in Table 8.40 in the *PDF Reference*.	0
BorderColor	See the `BC` entry in Table 8.40 in the *PDF Reference*.	
BackgroundColor	See the `BG` entry in Table 8.40 in the *PDF Reference*.	
Caption	See the `CA` entry in Table 8.40 in the *PDF Reference*.	
RolloverCaption	See the `RC` entry in Table 8.40 in the *PDF Reference*.	
DownCaption	See the `AC` entry in Table 8.40 in the *PDF Reference*.	
Icon	See the `I` entry in Table 8.40 in the *PDF Reference*.	
RolloverIcon	See the `RI` entry in Table 8.40 in the *PDF Reference*.	
DownIcon	See the `IX` entry in Table 8.40 in the *PDF Reference*.	
IconFit	See the `IF` entry in Table 8.40 in the *PDF Reference*.	
TextPosition	See the `TP` entry in Table 8.40 in the *PDF Reference*.	0

Appearance dictionary

appearance_dictionary

```
appearance_dictionary =
   element Normal { form_or_appearance_dictionary }
   & element Rollover { form_or_appearance_dictionary }?
   & element Down { form_or_appearance_dictionary }?
```

form_or_appearance_dictionary

```
form_or_appearance_dictionary =
   element Graphic { form_dictionary }
form_or_appearance_dictionary |=
   element Graphics { appearance_subdictionary }
```

appearance_subdictionary

```
appearance_subdictionary = element Graphic { attribute StateName { text }
form_dictionary }+
```

Description

Specifies how an annotation is presented visually on the page. For more information, see section 8.4.4 in the *PDF Reference*.

Element or attribute	Description	Default Value
Normal	See the `N` entry in Table 8.19 in the *PDF Reference*.	
Rollover	See the `R` entry in Table 8.19 in the *PDF Reference*.	The value of the `Normal` entry.
Down	See the `D` entry in Table 8.19 in the *PDF Reference*.	The value of the `Normal` entry.

Array object

default_array

```
default_array = default_array_element*
```

default_array_element

```
default_array_element =
  element Dict { default_dictionary }
default_array_element |=
  element Stream { array_stream_object }
default_array_element |=
  element Array { default_array }
default_array_element |=
  element String { attribute Value { pdf_byte_string }
  & attribute Value_enc { token }? }
default_array_element |=
  element Null { null }
default_array_element |=
  element Name { attribute Value { name }
  & attribute Value_enc { token }? }
default_array_element |=
  element Int { attribute Value { integer } }
default_array_element |=
  element Number { attribute Value { fixed } }
default_array_element |=
  element Bool { attribute Value { boolean } }
```

array_of_integer

```
array_of_integer =  ( list { integer } )
```

array_of_number

```
array_of_number =  ( list { number } )
```

array_of_name

```
array_of_name = element Name { attribute Value { name } }*
```

Description

A one-dimensional collection of objects arranged sequentially. These are used for representing user extensions embedded in PDF documents for which there is no known schema. The default_array_element is used for elements of arrays that are of unknown meaning. For more information, see the default_dictionary and section 3.2.5 in the *PDF Reference*.

Articles

article_threads_array

```
article_threads_array = element ArticleThread { thread_dictionary }*
```

thread_dictionary

```
thread_dictionary =
   bead_item_dictionary_reference_with_tag*
   & element ThreadInfo { document_information_dictionary }?
```

bead_item_dictionary_reference_with_tag

```
bead_item_dictionary_reference_with_tag =
   element Bead { bead_item_dictionary }
```

Description

An array of thread dictionaries representing the document's article threads. Threads and beads define a mechanism to link together a series of rectangles in the document. This can give the user a means to navigate the document in a logical way assuming each series of rectangles represents a flow of logical content, such as magazine articles which are continued over a series of non-contiguous pages. Article thread information is all stored within the Mars document backbone within the `Threads` element, which consists of a series of beads, each of which is a rectangular region which points to the page on which it lies. There is no information on the page related to threads and beads. For more information, see section 8.3.2 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
Bead	See the `F` entry in Table 8.11 in the *PDF Reference*.	
ThreadInfo	See the `I` entry in Table 8.11 in the *PDF Reference*.	

Attribute class

structure_class

```
structure_class =
   element Class { attribute Name { name }
   & attribute Name_enc { token }? }
structure_class |=
   element Classes { structure_class_array }
```

structure_class_array

```
structure_class_array = element Class { class_array_element }*
```

class_array_element

```
class_array_element = attribute Name { name }
   & attribute Name_enc { token }?
```

Description

An attribute class name or array of class names associated with a structure element. If many structure elements share the same set of attribute values, they can be defined as an attribute class sharing the identical attribute object. Structure elements refer to the class by name.

For more information, see *Attribute Classes* in section 10.6.4 in the *PDF Reference*.

Auxiliary color array

auxiliary_color_array

```
auxiliary_color_array =
   element ColorSpace { color_space }
   & element ColorValue { number }*
```

Description

An array that specifies the auxiliary color to be used when rendering the 3D image. For more information, see the `AC` entry in Table 9.42 in the *PDF Reference*.

Background dictionary

background_dictionary

```
background_dictionary = colorspace_name_or_array?
background_dictionary &= attribute BackgroundColor { rgb_color_array }?
   & attribute BackgroundColorsEntireAnnotation { boolean }?
```

Description

A 3D background dictionary defines the background over which a 3D view is to be drawn. For more information, see *Background Dictionaries* in section 9.5.3 in the *PDF Reference*.

Element or attribute	Description	Default Value
BackgroundColor	See the `C` entry in Table 9.39 in the *PDF Reference*.	
BackgroundColorsEntireAnnotation	See the `EA` entry in Table 9.39 in the *PDF Reference*.	false

Base64 string object

pdf_base64_string

```
pdf_base64_string = string
```

Description

A string containing a base 64-encoded representation of binary data.

Bead

bead_item_dictionary

```
bead_item_dictionary = attribute Page_ref { string }
bead_item_dictionary &= rectangle
```

Description

Beads are the individual content items that make up an article thread. A bead is a rectangular region which points to the page on which it lies.

For more information, see section 8.3.2 in the *PDF Reference*.

Bead dictionary

bead_item_dictionary

```
bead_item_dictionary =
  attribute Page_ref { string }
bead_item_dictionary &= rectangle
```

Description

A thread consists of a series of beads, each of which is a rectangular region which points to the page on which it lies. For more information, see section 8.3.2 in the *PDF Reference*.

Element or attribute	Description	Default Value
Page_ref	See the `P` entry in Table 8.12 in the *PDF Reference*.	

Bookmarks file

outline_dictionary_file

```
outline_dictionary_file =
   element Bookmarks { outline_dictionary }
```

outline_dictionary

```
outline_dictionary =
   outline_item_dictionary_reference_with_tag*
   & element Count { integer }?
```

outline_item_dictionary_reference_with_tag

```
outline_item_dictionary_reference_with_tag =
   element Bookmark { outline_item_dictionary }
```

outline_item_dictionary

```
outline_item_dictionary =
   element Title { pdf_text_string }
   & element Action { action_dictionary }?
   & attribute Color { rgb_color_array }?
   & attribute Styles {  ( list {  "Italic"? "Bold"? } ) }?
   & outline_item_dictionary_reference_with_tag*
   & element Count { integer }?
```

Description

The root of the document's outline hierarchy.

Bookmarks, also called outline items, are an indexing structure that is displayed and processed by PDF viewers. Each bookmark has a title and an action to be performed when it is activated.

Bookmarks are part of the Mars backbone. Note that bookmarks can reference page ordinals, the structure tree, or contain arbitrary actions.

For more information, see the `Outlines` entry in Table 3.25, and section 8.2.2 in the *PDF Reference*.

Element or attribute	Description	Default Value
Bookmarks	See the `Outlines` entry in Table 3.25 in the *PDF Reference*.	
Title	See the `Title` entry in Table 8.4 in the *PDF Reference*.	

Action	See the `A` entry in Table 8.4 in the *PDF Reference*.
Color	See the `C` entry in Table 8.4 in the *PDF Reference*.
Styles	See the `F` entry in Table 8.4 in the *PDF Reference*.
Count	See the `Count` entry in Table 8.3 in the *PDF Reference*.

Border effect dictionary

border_effect_dictionary

```
border_effect_dictionary =
  attribute Style {  "None" |  "Cloudy" }?
  & attribute Intensity { number }?
```

Description

Specifies an effect to be applied to the border of the annotations. For more information, see section 8.4.3 and Table 8.18 in the *PDF Reference*.

Element or attribute	Description	Default Value
Style	See the `S` entry in Table 8.18 in the *PDF Reference*.	None
Intensity	See the `I` entry in Table 8.18 in the *PDF Reference*.	0

Border style dictionary

border_style_dictionary

```
border_style_dictionary =
  attribute Width { number }?
  & attribute Style {  "Solid" |  "Dash" |  "Beveled" |  "Inset" |
  "Underline" }?
border_style_dictionary &= dash_array?
```

Description

A dictionary containing border characteristics. For more information, see section 8.4.3 in the *PDF Reference*.

Element or attribute	Description	Default Value

Width	See the `W` entry in Table 8.17 in the *PDF Reference*.	1
Style	See the `S` entry in Table 8.17 in the *PDF Reference*.	Solid

Box color information dictionary

box_color_information_dictionary

```
box_color_information_dictionary =
   element CropBox { box_style_dictionary }?
   & element BleedBox { box_style_dictionary }?
   & element TrimBox { box_style_dictionary }?
   & element ArtBox { box_style_dictionary }?
```

Description

A box color information dictionary specifies the colors and other visual characteristics to be used in displaying guidelines on the screen for the various page boundaries. For more information, see the `BoxColorInfo` entry in Table 3.27 and section 10.10.2 in the *PDF Reference*.

Box style dictionary

box_style_dictionary

```
box_style_dictionary =
   element Color { color_array_of_3_numbers }
   & attribute Width { number }?
   & attribute Style { pdf_text_name }?
   & element Pattern { line_dash_pattern_array }?
```

Description

A dictionary specifying the visual characteristics for displaying guidelines for a page's crop box, bleed box, trim box, or art box. For more information, see Tables 10.46 and 10.47 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
Color	See the `C` entry in Table 10.47 in the *PDF Reference*.	R="0.0" G="0.0" B="0.0"
Width	See the `W` entry in Table 10.47 in the *PDF Reference*.	1
Style	See the `S` entry in Table 10.47 in the *PDF Reference*.	S
Pattern	See the `D` entry in Table 10.47 in the *PDF Reference*.	On="3"

Button field

button_field_dictionary_reference

```
button_field_dictionary_reference =
   element DefaultValue {
   attribute _enc { token }?
   & name }?
   & element UnicodeValue { string_or_array_of_string }?
button_field_dictionary_reference &= common_field_dictionary
```

Description

An interactive control on the screen that the user can manipulate with the mouse. Unlike PDF, the form field values are represented in a separate file rather than in this field dictionary. That separate file follows the XFDF schema for form fields; it does not support XFDF-defined annotations. The file must be named /form/form_data.xfdf. For more information, see *Button Fields* in section 8.6.3 in the *PDF Reference*.

Element or attribute	Description	Default Value
DefaultValue	See the `DV` entry in Table 8.69 in the *PDF Reference*.	
UnicodeValue	See the `Opt` entry in Table 8.76 in the *PDF Reference*.	

Byte string object

pdf_byte_string

```
pdf_byte_string = string
```

Description

A string represented as a sequence of bytes. For more information, see section 3.2.3 in the *PDF Reference*.

CID mapping

cidmap_stream_or_name

```
cidmap_stream_or_name = cidmap_stream_object
cidmap_stream_or_name |=  "Identity"
```

cidmap_stream_object

```
cidmap_stream_object =
   attribute src { string }
   & common_stream_dictionary
```

common_stream_dictionary

```
common_stream_dictionary =
  element File { file_spec_dictionary }?
  & element FileFilters { filter_name_or_array }?
  & element FDecodeParms { parm_dictionary_or_array }?
```

Description

A specification of the mapping from CIDs to glyph indices. For Type 2 CID Fonts, a mapping can be specified for converting CID identifiers to glyph indices. This mapping is a binary array indexed by CID.

The map is stored in a separate package level file. The file can be located in any legal directory, but, by convention, it should be placed in the same directory as the font descriptor for the font. The file suffix and mime type must be `".cidmap"`.

For more information, see section 5.6.3 and the `CIDToGIDMap` entry in Table 5.14 in the *PDF Reference*.

CIDFont

CIDFont_dictionary

```
CIDFont_dictionary =
  attribute Subtype { pdf_text_name }
  & attribute BaseFont { name }
  & attribute BaseFont_enc { token }?
  & element CIDSystemInfo { CIDSystemInfo_dictionary }
  & element FontDescriptor { Font_descriptor_dictionary }
  & attribute DW { integer }?
  & element Widths { array }?
  & element DW2 { array_of_2_numbers }?
  & element VerticalWidths { array }?
  & element CIDToGIDMap { cidmap_stream_or_name }?
```

array

```
array = element * {  default_array_element }*
```

array_of_2_numbers

```
array_of_2_numbers =
  attribute XFraction { number }
  & attribute YFraction { number }
```

Description

A CIDFont program contains glyph descriptions that are accessed using a CID as the character selector. For more information, see section 5.6.3 in the *PDF Reference*.

When CID fonts are subsetted, an element (<CIDSet>) can be present in the font descriptor that references a separate package level file. This file is a bit array, indexed by CID, that indicates which CIDs are present in the font subset. For more information, see section 5.2.7 in the *PDF Reference*.

The file containing the bit array should be placed in the same directory as the font. It must have a suffix and mime type of `".cidset"`.

Element or attribute	Description	Default Value
Subtype		
BaseFont		
CIDSystemInfo		
FontDescriptor		
DW		
Widths	See the `W` entry in Table 5.14 in the *PDF Reference*.	
DW2		
VerticalWidths	See the `W2` entry in Table 5.14 in the *PDF Reference*.	
CIDToGIDMap		
XFraction	The first of two numbers specifying the default metrics for vertical writing. See the `DW2` entry in Table 5.14 in the *PDF Reference*.	
YFraction	The second of two numbers specifying the default metrics for vertical writing. See the `DW2` entry in Table 5.14 in the *PDF Reference*.	

CIDSystemInfo dictionary

CIDSystemInfo_dictionary

```
CIDSystemInfo_dictionary =
  attribute Registry { pdf_ascii_string }
  & attribute Ordering { pdf_ascii_string }
  & attribute Supplement { integer }
```

Description

CIDFont and CMap dictionaries contain a `CIDSystemInfo` entry specifying the character collection assumed by the CIDFont associated with the CMap - that is, the interpretation of the CID numbers used by the CIDFont. For more information, see section 5.6.2 in the *PDF Reference*.

CalGray color space

CalGray_color_space_dictionary

```
CalGray_color_space_dictionary =
  attribute WhitePoint { tristimulus_value }
  & attribute BlackPoint { tristimulus_value }?
  & attribute Gamma { number }?
```

Description

A CalGray color space is a CIE-based A color space with only one transformation stage instead of two. In this type of space, A represents the gray component of a calibrated gray space. For more information, see *CalGray Color Spaces* in section 4.5.4 in the *PDF Reference*.

CalRGB color space

CalRGB_color_space_dictionary

```
CalRGB_color_space_dictionary =
  attribute WhitePoint { tristimulus_value }
  & attribute BlackPoint { tristimulus_value }?
  & attribute Gamma { gamma_value }?
  & attribute Matrix { linear_interpretation_array }?
```

Description

A CIE-based ABC color space with only one transformation stage instead of two. In this type of space, A, B, and C represent calibrated red, green, and blue color values. For more information, see *CalRGB Color Spaces* in section 4.5.4 in the *PDF Reference*.

Caret annotation

caret_annotation_dictionary

```
caret_annotation_dictionary = common_annotation_dictionary
caret_annotation_dictionary &= common_markup_annotation_dictionary
caret_annotation_dictionary &=
  attribute Fringe { rectangle_blank_sep }?
  & attribute Symbol {  "paragraph" |  "none" }?
```

Description

A visual symbol that indicates the presence of text edits. For more information, see *Caret Annotations* in section 8.4.5 in the *PDF Reference*.

Element or attribute	Description	Default Value
Fringe	See the `RD` entry in Table 8.31 in the *PDF Reference.*	
Symbol	See the `Sy` entry in Table 8.31 in the *PDF Reference.*	None

Catalog dictionary

catalog_dictionary

```
catalog_dictionary =
  element Pages { page_tree_dictionary }
  & element PageLabels { page_label_dictionary_wrapper* }?
catalog_dictionary &= name_dictionary?
catalog_dictionary &=
  element ViewerPreferences { viewer_preferences_dictionary }?
  & attribute PageLayout { layout_type }?
  & attribute PageMode { pagemode_type }?
  & element Bookmarks { attribute src { string } }?
  & element Threads { attribute src { string } }?
  & element AfterOpen { action_or_destination }?
catalog_dictionary &= document_additional_actions_dictionary?
catalog_dictionary &=
  element URI { uri_base_dictionary }?
  & element AcroForm { acroform_dictionary }?
  & element Metadata { attribute src { string } }?
  & element StructureTypes { structure_tree_root_dictionary }?
  & element Marked { mark_information_dictionary }?
  & attribute Lang { pdf_text_string }?
  & element WWW { web_capture_information_dictionary }?
  & element OutputIntents { array_of_output_intent_dictionary }?
  & element ApplicationDatasets { attribute src { string } }?
  & element OCProperties { oc_properties_dictionary }?
  & element Legal { legal_dictionary }?
  & element Requirements { array_of_requirement_dictionaries }?
  & element Collection { collection_dictionary }?
  & attribute NeedsRendering { boolean }?
  & attribute PDFVersion { pdf_text_string }?
  & attribute Version { string }
  & attribute Class { text }
  & attribute DocumentID { string }?
  & attribute InstanceID { string }?
```

array_of_output_intent_dictionary

```
array_of_output_intent_dictionary = element OutputIntent {
output_intent_dictionary }*
```

array_of_requirement_dictionaries

```
array_of_requirement_dictionaries = element Requirement {
requirement_dictionary }*
```

Description

The root of the Mars document is the Mars backbone in the file "/backbone.mars". This file contains direct references to other files that comprise the Mars document. Files referenced from the backbone may in turn reference other files that are part of the document but are not directly referenced from the backbone. The core XML structure of a Mars file includes the basic content of the catalog dictionary in a PDF file. As with most Mars XML, the child elements can appear in any order. In addition, files may be part of the document through implicit resource associations.

Implicit resources associations are used when there is not a specific pointer from one Mars resource to a referenced resource. The existence of the resource under a particular name is used to form the association between the two resources. For example, annotations and metadata can be connected to particular Mars document resources by their existence in the Mars package under a specific name.

For example, `/page/3/im_123.jpeg` and `/page/3/im_123.jpeg.xmp` would be related using the metadata association rule, even though there is no explicit naming of the `.xmp` file in the `.jpeg` file or anywhere else.

Classes of implicit associations based on filename are defined in the `META-INF/package.xml` file. These relationships provide implied references between pairs of package-level files. Implicit relationships are used because they allow addition or removal of information without modifying either the referencer or the referenced component. This enables changes such as adding annotations or metadata that do not affect digital signatures or document content.

For more information, see section 3.6.1 in the *PDF Reference*.

Element or attribute	Description	Default Value
Bookmarks	The location of the file containing the document's outline dictionary. The root of this file is described in "Outline dictionary." The outline dictionary enables the viewer application to display a navigation panel. The user interacts with this panel to navigate interactively from one part of the document to another or to initiate other actions, as described in the `Outlines` entry in Table 3.25 in the *PDF Reference*.	
Threads	The location of the file containing the document's articles, which enable the user to navigate from page to page within a document. Articles chain together areas of content within the document that are logically connected but not physically sequential. (See the section "Articles.")	

StructureTypes	The location of the file containing the documents structure tree. The structure tree describes the logical structure of the document. The root in this file is described in "Structure tree root." The information in this file is equivalent to the `StructTreeRoot` entry in Table 3.25 in the *PDF Reference*.
Marked	Information about the document's usage of tagged PDF conventions. See Section 10.6 in the *PDF Reference*.
WWW	A Web Capture information dictionary. See Section 10.9.1 in the *PDF Reference*.
ApplicationDatasets	The location of the page-piece dictionary associated with the document. The root element in this file is described in "Page-piece dictionary." The information in this file is equivalent to the information described in Section 10.4 in the *PDF Reference*.
Class	The high order Mars version. It is used to determine whether the version has changed and if the Mars document is readable. As of this document's release date, the Mars plug-in supports the `Class` value "`04-18-07`".
Version	The Mars schema version. As of this document's release date, the Mars plug-in supports the `Version` value "`0.8.0`".
PDFVersion	The PDF version represented by this document. The value of this attribute corresponds to the version information provided in the first line of the PDF file header. Section 3.4.1 in the *PDF Reference* describes this value as `%PDF-`*`m.n`*, where the *m.n* represents the version information. For example, `1.7`.

Choice field

choice_field_dictionary_reference

```
choice_field_dictionary_reference =
  element DefaultValue { string_or_array_of_string }?
  & element Choices { choice_field_opt_array }?
  & attribute TopIndex { integer }?
  & attribute Selected { array_of_integer }?
choice_field_dictionary_reference &= common_field_dictionary
```

Description

A field containing several text items, one or more of which can be selected as the field value. Unlike PDF, the form field values are represented in a separate file rather than in this field dictionary. That separate file follows the XFDF schema for form fields; it does not support XFDF-defined annotations. The file must be named /form/form_data.xfdf. For more information, see *Choice Fields* in section 8.6.3 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
DefaultValue	See the `DV` entry in Table 8.69 in the *PDF Reference*.	
Choices	See the `Opt` entry in Table 8.80 in the *PDF Reference*.	
TopIndex	See the `TI` entry in Table 8.80 in the *PDF Reference*.	
Selected	See the `I` entry in Table 8.80 in the *PDF Reference*.	

Collection dictionary

collection_dictionary

```
collection_dictionary =
  element Schema { collection_schema_dictionary }?
  & attribute EmbeddedFilename { pdf_byte_string }?
  & attribute EmbeddedFilename_enc { token }?
  & attribute View {  "Details" |  "Title" |  "Hidden" }?
  & element Sort { collection_sort_dictionary }?
```

Description

Specifies the viewing and organizational characteristics of portable collections. For more information, see section 8.2.4 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
Schema	A collection schema dictionary. See the `Schema` entry in Table 8.6 in the *PDF Reference*.	
EmbeddedFilename	See the `D` entry in Table 8.6 in the *PDF Reference*.	
View	The initial view. See the `View` entry in Table 8.6 in the *PDF Reference*.	
Sort	A collection sort dictionary, which specifies the order in which items in the collection should be sorted in the user interface. See the `Sort` entry in Table 8.6 in the *PDF Reference*.	

Collection field dictionary

collection_field_dictionary

```
collection_field_dictionary =
   attribute Subtype { name }
   & attribute Subtype_enc { token }?
   & attribute FieldName { pdf_text_string }
   & attribute Order { integer }?
   & attribute Visible { boolean }?
   & attribute Editable { boolean }?
```

Description

A dictionary that describes the attributes of a particular field in a portable collection, including the type of data stored in the field and the lookup key used to locate the field data in the file specification dictionary. For more information, see Table 8.8 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
FieldName	See the `N` entry in Table 8.8 in the *PDF Reference*.	
Order	See the `O` entry in Table 8.8 in the *PDF Reference*.	
Visible	See the `V` entry in Table 8.8 in the *PDF Reference*.	true
Editable	See the `E` entry in Table 8.8 in the *PDF Reference*.	false

Collection item dictionary

collection_item_dictionary_reference

```
collection_item_dictionary_reference = collection_item_dictionary
```

collection_item_dictionary

```
collection_item_dictionary = element Field { attribute Name { text }
collection_item_value }+
```

collection_item_value

```
collection_item_value = attribute Prefix { pdf_text_string }?
collection_item_value |=
   attribute Value { pdf_text_string | number }
   attribute type {  "string" |  "number" }
```

Description

Contains the data described by the collection schema dictionary for a particular file in a collection. For more information, see section 3.10.5 in the *PDF Reference*.

Element or attribute	Description	Default Value
Value	See the `D` entry in Table 3.46 in the *PDF Reference*.	
Prefix	See the `P` entry in Table 3.46 in the *PDF Reference*.	

Collection schema dictionary

collection_schema_dictionary

```
collection_schema_dictionary =
   element Field { attribute Name { text }
      attribute Name_enc { token }? collection_field_dictionary }+
```

Description

A dictionary that consists of a variable number of individual collection field dictionaries. Each collection field dictionary has a key chosen by the producer, which is used to associate a field with data in a file specification. For more information, see Table 8.7 in the *PDF Reference*.

Element or attribute	Description	Default Value
Field	See *Other keys chosen by producer* in Table 8.7 in the *PDF Reference*.	
Sort	See *Sort* dictionary in Table 8.6 in the *PDF Reference*.	

Collection sort dictionary

collection_sort_dictionary

```
collection_sort_dictionary = collection_sort_field_name_or_array
collection_sort_dictionary &= collection_sort_ascending?
```

collection_sort_field_name_or_array

```
collection_sort_field_name_or_array =
   element SortField { attribute _enc { token }?
      & name }
collection_sort_field_name_or_array |=
   element SortFields { collection_sort_field_elements }
```

collection_sort_ascending

```
collection_sort_ascending =
   attribute SortAscending { collection_sort_ascending_elements }
```

collection_sort_ascending_elements

```
collection_sort_ascending_elements =  ( list { boolean+ } )
```

collection_sort_field_elements

```
collection_sort_field_elements =
   element SortField {
      attribute _enc { token }?
      & name }
```

Description

A collection sort dictionary identifies the fields that are used to sort items in the collection. The type of sorting depends on the type of data:

- Text strings are ordered lexically from smaller to larger, if ascending order is specified.
- Numbers are ordered numerically from smaller to larger, if ascending order is specified.
- Dates are ordered from oldest to newest, if ascending order is specified.

For more information, see Table 8.9 in the *PDF Reference*..

Element or attribute	**Description**	**Default Value**
SortField	See the `S` entry in Table 8.9 in the *PDF Reference*.	
SortFields	Used to specify multiple sort fields. See the `S` entry in Table 8.9 in the *PDF Reference*.	
SortAscending	See the `A` entry in Table 8.9 in the *PDF Reference*.	

Color space array

color_space_array

```
color_space_array = CalGray_color_space_array
color_space_array |= CalRGB_color_space_array
color_space_array |= Lab_color_space_array
color_space_array |= ICC_profile_array
color_space_array |=
   element Separation { separation_color_space }
color_space_array |=
   element DeviceN { device_n_color_space }
color_space_array |=
   element Indexed { indexed_color_space }
color_space_array |=
   element PatternCS { pattern_color_space }
```

CalGray_color_space_array

```
CalGray_color_space_array =
   element CalGray { CalGray_color_space_dictionary }
```

CalRGB_color_space_array

```
CalRGB_color_space_array =
  element CalRGB { CalRGB_color_space_dictionary }
```

Lab_color_space_array

```
Lab_color_space_array =
  element Lab { Lab_color_space_dictionary }
```

ICC_profile_array

```
ICC_profile_array =
  element ICCBased { ICC_profile_stream_dictionary }
```

Description

A color space is defined by an array object whose first element is a name object identifying the color space family. The remaining array elements, if any, are parameters that further characterize the color space; their number and types vary according to the particular family. For families that do not require parameters, the color space can be specified simply by the family name itself instead of an array. For more information, see section 4.5.2 in the *PDF Reference*.

Element or attribute	Description	Default Value
CalGray	CalGray color space array.	
CalRGB	CalRGB color space array.	
Lab	Lab color space array.	
ICCBased	ICC profile array.	

Color space map

color_space_map

```
color_space_map =  ( list { integer } )
```

Description

An array of numbers describing how to map image samples into the range of values appropriate for the image's color space. For more information, see the `Decode` entry in Table 4.39 in the *PDF Reference*.

Color space name

colorspace_name_or_array

```
colorspace_name_or_array =
  attribute ColorSpaceName { name }
  & attribute ColorSpaceName_enc { token }?
colorspace_name_or_array |=
  element ColorSpace { color_space_array }
```

Description

The color space of a 3D background dictionary. Some color information in Mars documents is converted to ICC color profiles, which are stored in separate package objects.

PDF documents have the following color spaces that cannot be represented as ICCBased:

- Separation
- DeviceN
- Indexed
- Pattern

The *PDF Reference* defines the term *color spaces*, which includes not only different color systems (device color and CIE-based color), but also *special color spaces* (Indexed, Separation and DeviceN). For Mars documents, PDF color is mapped into SVG markup:

- Each distinct color space used within a page results in the generation of an `<svg:color-profile>` element at the top of the SVG content. This applies to device color, CIE-based color, Indexed, Separation and DeviceN.
- All color values specified via `fill` and `stroke` properties include both the sRGB equivalent for the color value (as required by the SVG language specification) and an `icc-color(...)` function value containing the color values that are actually used on import. Note that the `icc-color(...)` function is used for all color spaces, even ones that do not use ICC profiles such as device color.

Each `<color-profile>` element would include an `xlink:href` attribute. This attribute can be used in two cases:

- For well-known color spaces, the value represents a unique URL which identifies that color space. This might be an existing ICC color space, or might be a new color space which we invent (for example, DeviceCMYK colors might have a URL that looks something like `"http://ns.adobe.com/xpdf/DeviceCMYK"`). This approach applies to the following color spaces: DeviceCMYK, DeviceRGB, DeviceGray, CalGray, CalRGB, Lab, Indexed, Separation and DeviceN.
- For an ICCBased color space, the ICC profiles are split apart into separate files within the package containing the Mars document.

For more information, see the `CS` entry in Table 9.41 in the *PDF Reference*.

Element or attribute	Description	Default Value
ColorSpaceName	The name of the color space.	
ColorSpace	The color space.	

Color space

color_space

```
color_space = name
color_space |= color_space_array
```

Description

A color space is a system that can be used to specify abstract colors in a device-independent way. Some color spaces are related to device color representation (grayscale, RGB, CMYK), others to human visual perception.

Some PDF color spaces are converted to ICC color profiles. There are a number of places they can appear, but most are in the page resource files. The ICC profiles are often separate package files referred to by the SVG page content or resource file entries.

For more information, see section 4.5 in the *PDF Reference*.

Colorant dictionary

colorant_dictionary

```
colorant_dictionary =
   element Colorant { attribute Name { text }
      attribute Name_enc { token }? separation_color_space }+
```

Description

A dictionary identifying the individual colorants associated with a printer's mark, such as a color bar. For each entry in this dictionary, the key is a colorant name and the value is an array defining a Separation color space for that colorant. The key must match the colorant name given in that color space. For more information, see the `Colorants` entry in Table 10.49 in the *PDF Reference*.

Common action dictionary

common_action_dictionary

```
common_action_dictionary = element Next { action_dictionary_or_array }?
```

action_dictionary_or_array

```
action_dictionary_or_array = action_dictionary*
```

Description

A common action dictionary contains entries common to all action dictionaries. For more information, see Table 8.43 in the *PDF Reference*.

Common field dictionary

common_field_dictionary

```
common_field_dictionary =
  element SubFields { array_of_field_dictionary }?
  & attribute Name { pdf_text_string }?
  & attribute UIName { pdf_text_string }?
  & attribute MapTo { pdf_text_string }?
  & attribute Flags {  ( list {  "ReadOnly"? "Required"? "NoExport"?
  "Multiline"? "Password"? "NoToggleOff"? "Radio"? "PushButton"? "Combo"?
  "Edit"? "Sort"? "FileSelect"? "MultiSelect"? "DoNotSpellCheck"?
  "DoNotScroll"? "Comb"? "RadiosInUnison"? "CommitOnSelChange"?
  "RichText"? } ) }?
common_field_dictionary &=
widget_annotation_additional_actions_dictionary?
common_field_dictionary &= variable_text_dictionary_fields
common_field_dictionary &= common_field_widget
```

common_field_widget

```
common_field_widget = attribute Widget_ref { object_reference }?
```

Description

Each field in a document's interactive form is defined by a field dictionary, which must be an indirect object. For more information, see section 8.6.2 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
SubFields	See the `Kids` entry in Table 8.69 in the *PDF Reference*.	
Name	See the `T` entry in Table 8.69 in the *PDF Reference*.	
UIName	See the `TU` entry in Table 8.69 in the *PDF Reference*.	
MapTo	See the `TM` entry in Table 8.69 in the *PDF Reference*.	
Flags	See the `Ff` entry in Table 8.69 in the *PDF Reference*.	0

Common font dictionary

common_font_dictionary

```
common_font_dictionary =
  attribute Name { name }?
  & attribute Name_enc { token }?
  & attribute BaseFont { name }
  & attribute BaseFont_enc { token }?
  & attribute FirstChar { integer }?
  & attribute LastChar { integer }?
  & element Widths { array_of_integer }
  & element FontDescriptor { Font_descriptor_dictionary }
```

Description

A dictionary containing attributes common to Type 0, Type 1, Type 3, and TrueType font dictionaries. (Font dictionaries do not apply to SVG fonts.) Each font consists of a font descriptor file, the font file itself, and possibly other files. The font descriptors are represented in XML and can be shared among pages in a Mars document. The fonts themselves are represented in OpenType® format where possible.

Each font descriptor is a separate package level file. Font descriptor files may be in any non-reserved location in the package but by convention should be placed in either the `/fonts` directory or the directory for the page on which the font descriptor is used.

The font descriptor contains information about the font including widths and Unicode mappings that are useful if a font substitution should be required. This information allows the document to be rendered with layout similar to the original. Note that the Mars font descriptor files correspond to the information in PDF Type 0, Type 1, Type 3, and TrueType font dictionaries.

For more information, see Chapter 5 in the *PDF Reference*.

Example

```
<Font>
  <Type1 LastChar="255" BaseFont="Bembo" FirstChar="0">
    <FontDescriptor StemV="67" FontName="Bembo" Flags="Serif Nonsymbolic"
      Descent="-233" Ascent="673" CapHeight="622" XHeight="396"
      FontBBox="-206 -233 1006 896" />
    <Widths>278 278 278 ..... 677 594 677 594 </Widths>
  </Type1>
</Font>
```

Common markup annotation dictionary

common_markup_annotation_dictionary

```
common_markup_annotation_dictionary =
   attribute Title { pdf_text_string }?
   & element Popup { popup_annotation_dictionary }?
   & attribute Opacity { number }?
   & element RTContents { rt_string_or_stream }?
   & attribute CreationDate { date }?
   & attribute Subj { pdf_text_string }?
   & attribute InReplyTo { string }?
   & attribute ReplyType { pdf_text_name }?
   & attribute Intent { pdf_text_name }?
   & element ExData { external_data_dictionary }?
```

Description

A dictionary containing entries common to all annotation dictionaries. For more information, see *Markup Annotations* in section 8.4.5 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
Title	See the `T` entry in Table 8.21 in the *PDF Reference*.	
Opacity	See the `CA` entry in Table 8.21 in the *PDF Reference*.	
RTContents	See the `RC` entry in Table 8.21 in the *PDF Reference*.	
InReplyTo	See the `IRT` entry in Table 8.21 in the *PDF Reference*.	
ReplyType	See the `RT` entry in Table 8.21 in the *PDF Reference*.	R
Intent	See the `IT` entry in Table 8.21 in the *PDF Reference*.	

Common media clip dictionary

common_media_clip_dictionary

```
common_media_clip_dictionary =
   attribute Subtype { name }
   & attribute Subtype_enc { token }?
   & attribute Name { pdf_text_string }?
```

Description

A dictionary containing entries common to all media clip dictionaries. For more information, see Table 9.8 in the *PDF Reference*.

Element or attribute	Description	Default Value
Subtype	See the `S` entry in Table 9.8 in the *PDF Reference*.	
Name	See the `N` entry in Table 9.8 in the *PDF Reference*.	

Common rendition dictionary

common_rendition_dictionary

```
common_rendition_dictionary = attribute Name { pdf_text_string }?
common_rendition_dictionary &= common_must_honor_best_effort
```

Description

A dictionary containing entries common to all rendition dictionaries. For more information, see section 9.1.2 in the *PDF Reference*.

Element or attribute	Description	Default Value
Name	See the `N` entry in Table 9.1 in the *PDF Reference*.	

Common stream dictionaries

common_stream_dictionary

```
common_stream_dictionary =
   element File { file_spec_dictionary }?
   & element FileFilters { filter_name_or_array }?
   & element FDecodeParms { parm_dictionary_or_array }?
```

Description

A common stream dictionary describes a file containing non-specific data.

Content set object array

array_of_object_reference

```
array_of_object_reference = element Obj { attribute ref { pdf_reference } }+
```

Description

An array of indirect references to the objects belonging to the content set. For more information, see the `O` entry in Table 10.38 in the *PDF Reference*.

Custom action dictionary

custom_action_dictionary

```
custom_action_dictionary = common_action_dictionary
```

Description

Enables support of custom action dictionaries.

Custom annotation

custom_annotation_dictionary

```
custom_annotation_dictionary = common_annotation_dictionary
custom_annotation_dictionary &= common_markup_annotation_dictionary
```

Date object

date

```
date = date
```

Description

A standard date format that closely follows the international standard ASN.1 (Abstract Syntax Notation One), defined in ISO/IEC 8824. For example, December 23, 2006, at 7:52 PM, U.S. Pacific Standard Time, is represented by the string `D:200612231952'08'00'`. For more information, see section 3.8.3 in the *PDF Reference*.

Destinations, named and explicit

destination

```
destination = destination_array
destination |=
   attribute Name { pdf_byte_string }
   & attribute Name_enc { token }?
```

destination_array

```
destination_array = tagged_xyz_destination_array
destination_array |= tagged_fit_destination_array
destination_array |= tagged_fith_destination_array
destination_array |= tagged_fitv_destination_array
destination_array |= tagged_fitr_destination_array
destination_array |= tagged_fitb_destination_array
destination_array |= tagged_fitbh_destination_array
destination_array |= tagged_fitbv_destination_array
```

tagged_xyz_destination_array

```
tagged_xyz_destination_array =
  element XYZ { xyz_destination_array }
```

xyz_destination_array

```
xyz_destination_array =
  destination_array_common
  & attribute Left { number }?
  & attribute Top { number }?
  & attribute Zoom { number }?
```

tagged_fit_destination_array

```
tagged_fit_destination_array =
  element Fit { fit_destination_array }
```

fit_destination_array

```
fit_destination_array = destination_array_common
```

tagged_fith_destination_array

```
tagged_fith_destination_array =
  element FitH { fith_destination_array }
```

fith_destination_array

```
fith_destination_array =
  destination_array_common
  & attribute Top { number }?
```

tagged_fitv_destination_array

```
tagged_fitv_destination_array =
  element FitV { fitv_destination_array }
```

fitv_destination_array

```
fitv_destination_array =
  destination_array_common
  & attribute Left { number }?
```

tagged_fitr_destination_array

```
tagged_fitr_destination_array =
  element FitR { fitr_destination_array }
```

fitr_destination_array

```
fitr_destination_array =
  destination_array_common
  & attribute Left { number }
  & attribute Bottom { number }
  & attribute Right { number }
  & attribute Top { number }
```

tagged_fitb_destination_array

```
tagged_fitb_destination_array =
  element FitB { fitb_destination_array }
```

fitb_destination_array

```
fitb_destination_array = destination_array_common
```

tagged_fitbh_destination_array

```
tagged_fitbh_destination_array =
  element FitBH { fitbh_destination_array }
```

fitbh_destination_array

```
fitbh_destination_array =
  destination_array_common
  & attribute Top { number }?
```

tagged_fitbv_destination_array

```
tagged_fitbv_destination_array =
  element FitBV { fitbv_destination_array }
```

fitbv_destination_array

```
fitbv_destination_array =
  destination_array_common
  & attribute Left { number }?
```

destination_array_common

```
destination_array_common = page_link
```

page_link

```
page_link =
   attribute Page_ref { string }
page_link |=
   attribute PageNum { integer }
```

Description

A destination defines a particular view of a document to be rendered when a bookmark is activated, an annotation is opened, or an action is performed. The view consists of this information:

- The page to be displayed
- The location of the page within the document window
- The magnification (zoom) factor to use when displaying the page

A destination can be specified either explicitly by specifying the page and location on the page, or indirectly by specifying the name of a destination that resolves to an explicit destination.

For more information, see Specifying Destinations in the *Mars Developer Guide*.

Device colorant

colorant_name_or_string

```
colorant_name_or_string =
   element ColorantName {
      attribute _enc { token }?
      & name }
colorant_name_or_string |=
   element Colorantstring {
      attribute _enc { token }?
      & pdf_byte_string }
```

Description

The name of the device colorant to be used in rendering a separation. For more information, see the `DeviceColorant` entry in Table 10.50 in the *PDF Reference*.

Element or attribute	Description	Default Value
ColorantName	A name representing the device colorant.	
Colorantstring	A string representing the device colorant.	

DeviceN color space

device_n_color_space

```
device_n_color_space =
  element Colorants { array_of_name }
  & element AlternateSpace { color_space }
  & element TintXform { function_dictionary }
  & element Attrs { device_n_color_space_attributes_dictionary }?
```

device_n_color_space_attributes_dictionary

```
device_n_color_space_attributes_dictionary =
  element Colorants { colorant_dictionary }?
  & attribute Subtype { pdf_text_name }?
  & element Process { process_dictionary }?
  & element MixingHints { mixing_hints_dictionary }?
```

Description

DeviceN color spaces can contain an arbitrary number of color components. They provide greater flexibility than is possible with standard device color spaces. For more information, see *DeviceN Color Spaces* in section 4.5.5 in the *PDF Reference*.

Element or attribute	Description	Default Value
Colorants	An array of name objects specifying the individual color components.	
AlternateSpace	An array or name object that can be any device or CIE-based color space but not another special color space (Pattern, Indexed, Separation, or DeviceN).	
TintXform	Specifies a function that is used to transform the tint values into the alternate color space.	
Attrs	A dictionary containing additional information about the components of color space that consumer applications may use.	

DeviceN mixing hints dictionary

mixing_hints_dictionary

```
mixing_hints_dictionary =
  element Solidities { ink_density_dictionary }
  & element PrintingOrder { array_of_name }
```

ink_density_dictionary

```
ink_density_dictionary =
   element Density { number
      & attribute Name { pdf_byte_string } }
```

Description

A dictionary that specifies optional attributes of the inks to be used in blending calculations when used as an alternative to the tint transformation function. For more information, see Table 4.23 in the *PDF Reference*.

DeviceN process dictionary

process_dictionary

```
process_dictionary =
   element ColorSpace { colorspace_name_or_array }
   & element Components { array_of_name }
```

Description

A dictionary that describes the process color space whose components are included in this color space. For more information, see Table 4.22 in the *PDF Reference*.

DeviceRGB color space

color_array_of_3_numbers

```
color_array_of_3_numbers =
   attribute R { number }
   & attribute G { number }
   & attribute B { number }
```

Description

An array of three numbers in the range `0.0` to `1.0`, representing the components in the DeviceRGB color space of the color to be used in a box style dictionary. For more information, see the `C` entry in Table 10.47 in the *PDF Reference*.

Element or attribute	Description	Default Value
R	A value representing the red component of the DeviceRGB color space.	
G	A value representing the green component of the DeviceRGB color space.	
B	A value representing the blue component of the DeviceRGB color space.	

Dictionary object

default_dictionary

```
default_dictionary = element * { default_dict_element }*
```

default_dict_element

```
default_dict_element =
   attribute type { "null" }
default_dict_element |=
   attribute type { "string" }
   & attribute Value { pdf_byte_string }
   & attribute Value_enc { token }?
default_dict_element |=
   attribute type { "integer" }
   & attribute Value { integer }
default_dict_element |=
   attribute type { "number" }
   & attribute Value { number }
default_dict_element |=
   attribute type { "boolean" }
   & attribute Value { boolean }
default_dict_element |=
   attribute type { "name" }
   & attribute Value { name }
   & attribute Value_enc { token }?
default_dict_element |=
   attribute type { "dictionary" }
   & element * { default_dict_element }*
default_dict_element |=
   attribute type { "array" }
   & default_array_element*
default_dict_element |=
   attribute type { "stream" }
   & attribute src { string }
   & common_stream_dictionary
```

Description

An associative table containing pairs of objects known as the dictionary's entries. The first element of each entry is the *key* and the second element is the *value*. These are used for representing PDF constructs not defined by the *PDF Reference*. For more information, see section 3.2.6 in the *PDF Reference*.

Document catalog additional actions dictionary

document_additional_actions_dictionary

```
document_additional_actions_dictionary =
  element BeforeClose { javascript_action_dictionary }?
  & element BeforeSave { javascript_action_dictionary }?
  & element AfterSave { javascript_action_dictionary }?
  & element BeforePrint { javascript_action_dictionary }?
  & element AfterPrint { javascript_action_dictionary }?
```

Description

An additional-actions dictionary defining the actions to be taken in response to various trigger events affecting the document as a whole. For more information, see the `AA` entry in Table 3.25, as well as Table 8.47, in the *PDF Reference*.

Element or attribute	Description	Default Value
BeforeClose	See the `DC` entry in Table 8.47 in the *PDF Reference*.	
BeforeSave	See the `WS` entry in Table 8.47 in the *PDF Reference*.	
AfterSave	See the `DS` entry in Table 8.47 in the *PDF Reference*.	
BeforePrint	See the `WP` entry in Table 8.47 in the *PDF Reference*.	
AfterPrint	See the `DP` entry in Table 8.47 in the *PDF Reference*.	

Document information dictionary

document_information_dictionary

```
document_information_dictionary =
  attribute Title { pdf_text_string }?
  & attribute Author { pdf_text_string }?
  & attribute Subject { pdf_text_string }?
  & attribute Keywords { pdf_text_string }?
  & attribute Creator { pdf_text_string }?
  & attribute Producer { pdf_text_string }?
  & attribute CreationDate { date }?
  & attribute ModDate { date }?
  & attribute Trapped { pdf_text_name }?
  & attribute SavedBy { pdf_text_string }?
```

Description

A dictionary containing metadata for the document. For more information, see section 10.2.1 in the *PDF Reference*.

Document requirements

array_of_requirement_handler_dictionaries

```
array_of_requirement_handler_dictionaries = element Handler {
requirement_handler_dictionary }*
```

requirement_dictionary

```
requirement_dictionary =
  attribute RequirementType { name }
  & attribute RequirementType_enc { token }?
  & element Handlers { array_of_requirement_handler_dictionaries }?
```

requirement_handler_dictionary

```
requirement_handler_dictionary =
  attribute HandlerType { name }
  & attribute HandlerType_enc { token }?
```

Description

A dictionary used by a document to specify requirements that must be present in a PDF consumer application in order for the document to function properly. For more information, see section 8.9 in the *PDF Reference*.

Element or attribute	Description	Default Value
RequirementType	See the `S` entry in Table 8.113 in the *PDF Reference*.	
Handlers	See the `RH` entry in Table 8.113 in the *PDF Reference*.	
DocumentID	A URI that uniquely identifies the document.	
InstanceID	A URI that identifies an instance of the document.	

Embedded file parameter dictionary

embedded_file_stream_parameter_dictionary

```
embedded_file_stream_parameter_dictionary =
  attribute Size { integer }?
  & attribute CreationDate { date }?
  & attribute ModDate { date }?
  & element Mac { embedded_file_stream_parameter_mac_dictionary }?
  & attribute Checksum { pdf_base64_string }?
```

Description

A dictionary containing additional, file-specific information to be used in an embedded file specification dictionary. For more information, see the `Params` entry in Table 3.41 in the *PDF Reference*.

Embedded file stream dictionary

embedded_file_stream_information_dictionary

```
embedded_file_stream_information_dictionary =
  element FileData { embedded_file_stream_dictionary }?
  & element UFileData { pdf_text_string }?
```

embedded_file_stream_dictionary

```
embedded_file_stream_dictionary =
  attribute FileType { pdf_text_name }?
  & element Params { embedded_file_stream_parameter_dictionary }
  & attribute src { string }
  & common_stream_dictionary
```

Description

Embedded file streams allow the contents of referenced files to be embedded directly within the body of the Mars file. For more information, see section 3.10.3 in the *PDF Reference*.

Element or attribute	Description	Default Value
FileData	See the `F` entry in Table 3.41 in the *PDF Reference*.	
UFileData	See the `UF` entry in Table 3.41 in the *PDF Reference*.	
FileType	See the `Subtype` entry in Table 3.42 in the *PDF Reference*.	

Embedded files

embedded_files_file

```
embedded_files_file =
  element EmbeddedFiles { embedded_file_stream_name_tree* }
```

embedded_file_stream_name_tree

```
embedded_file_stream_name_tree =
  element EmbeddedFile {
    attribute Key { text }
    & file_spec_dictionary1 }
```

Description

A mapping of name strings to file specifications for embedded file streams. There is one entry per embedded file.

Embedded files represent supplementary files that travel with the document but are not part of the displayed document contents. Embedded files can be, but need not be, associated with a file attachment annotation which appears on a document page.

There are three components concerned with embedded files:

- The file itself, which is represented as a separate file in the document package. The file must appear in the `/file` directory of the package or a subdirectory thereof. The embedded file can have any name that is a legal name as defined by Zip. Note that the file name within the package is used only to reference the file from the backbone or annotation. A separate attribute within the XML (`Name` and/or `UName`) holds the file name as it will be displayed for the user.
- In the document backbone, an element `<EmbeddedFile>` appears within the element `<EmbeddedFiles>` for each embedded file. This element contains the file name and optional information such as a user interface description of the file, MIME type, creation date, checksum, and last modification date.
- Optionally, a file attachment annotation which contains the same information as the `<EmbeddedFile>` element, as well as the information needed to display the annotation on its page. The annotation information is stored in the `.ann` file associated with the page on which it appears. An embedded file annotation is a markup annotation.

For more information, see section 3.10.3, as well as the `EmbeddedFiles` entry in Table 3.28 in the *PDF Reference*.

Example

```
<EmbeddedFiles>
  <EmbeddedFile Key="Untitled Object" UName="arch_pic.bmp"
    Name="arch_pic.bmp">
    <FileData src="/files/ef_4209265555-3" FileType="image/bmp" >
      <Params Creation="D:20060706161059-07'00'"
        ModDate="D:20060706161059-07'00'" Size="506934"
        Checksum="48uBet+wuxfygp5JKshUcQ=="/>
    </File>
    <Desc>This is a sample image that can be projected with the EXR-55</Desc>
    </EmbeddedFile>
</EmbeddedFiles>
```

Embedded go-to action

gotoe_action_dictionary

```
gotoe_action_dictionary =
   element File { file_spec_dictionary }?
   & element Dest { destination }
   & attribute NewWindow { boolean }?
   & element Target { target_dictionary }?
gotoe_action_dictionary &= common_action_dictionary
```

Description

An embedded go-to action is similar to a remote go-to action, but allows jumping to or from a Mars document that is embedded in another Mars document. For more information, see *Embedded Go-To Actions* in section 8.5.3 in the *PDF Reference* and in *Representing Actions* in the guide *Developing Applications Using Mars*.

Element or attribute	**Description**	**Default Value**
File	See the `F` entry in Table 8.51 in the *PDF Reference*.	
Dest	See the `D` entry in Table 8.51 in the *PDF Reference*.	
Target	See the `T` entry in Table 8.51 in the *PDF Reference*.	

Explicit resource reference

pdf_reference

```
pdf_reference = string
```

Description

Explicit resource references are used when there is a specific need to reference a file, such as a font or image, within the document package.

At the reference point, an element with a `src` attribute names the component via a URI. The URI must be either relative or root-based, but cannot specify a different scheme and must refer to an object within the same package as the referencer. When more than one reference must be placed on the same element, an attribute other than `src` is used.

An explicit element reference is a URI that can include a file reference and an identifier reference is used to reference a specific XML element in one resource from another resource. The URI reference includes the name of the referenced resource much like an explicit resource reference. The URI also includes an identifier that identifies a specific element via its `ID` attribute.

References can also be between elements of a single XML file.

External data dictionary

external_data_dictionary

```
external_data_dictionary =
  attribute Subtype { pdf_text_name }
  & element Annotation { annotation_reference_or_name }
  & element View { attribute ref { pdf_reference } }
  & attribute Checksum { pdf_base64_string }?
```

annotation_reference_or_name

```
annotation_reference_or_name =
  attribute Name { pdf_byte_string }
  & attribute Name_enc { token }?
annotation_reference_or_name |=
  attribute ref { pdf_reference }
```

Description

An external data dictionary specifying data to be associated with the annotation. For more information, see Table 9.48 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
Annotation	See the `3DA` entry in Table 9.48 in the *PDF Reference*.	
View	See the `3DV` entry in Table 9.48 in the *PDF Reference*.	
Checksum	See the `MDS` entry in Table 9.48 in the *PDF Reference*.	

Field dictionary array

array_of_field_dictionary

```
array_of_field_dictionary = field_dictionary*
```

Description

An array of references to field dictionaries. For more information, see the `Fields` entry in Table 8.67 and the `Kids` entry in Table 8.69 in the *PDF Reference*.

Field dictionary

field_dictionary

```
field_dictionary =
  element Button { button_field_dictionary_reference }
field_dictionary |=
  element TextField { text_field_dictionary_reference }
field_dictionary |=
  element Choice { choice_field_dictionary_reference }
field_dictionary |=
  element Signature { signature_field_dictionary_reference }
field_dictionary |=
  element Container { container_field_dictionary_reference }
```

container_field_dictionary_reference

```
container_field_dictionary_reference = common_field_dictionary
```

Description

A document field. Adobe PDF forms as implemented in Acrobat 5 and earlier (and supported in Acrobat 6 and later) include a PDF description of form fields and their behavior. This description in Mars appears directly in the Mars backbone.

In Mars, the default appearances for form fields are represented in SVG. These SVG fragments can appear inline in the form field widget entry or in separate package-level files (which are directly referenced from the appearance element).

For more information see section 8.6.3 in the *PDF Reference*.

Element or attribute	Description	Default Value
Button	Button field.	
Text	Text field.	
Choice	Choice field.	
Signature	Signature field.	

Fields array

fields_array

```
fields_array = string_or_field_dictionary_reference*
```

string_or_field_dictionary_reference

```
string_or_field_dictionary_reference =
  element Field { attribute Pathname { pdf_text_string } }
string_or_field_dictionary_reference |=
  element Field { attribute ref { pdf_reference } }
```

Description

The `Fields` array in a Submit-Form or Reset-Form action. For more information, see the `Fields` entries in Tables 8.81 and 8.83 in the *PDF Reference*.

File attachment annotation

file_attachment_annotation_dictionary

```
file_attachment_annotation_dictionary = common_annotation_dictionary
file_attachment_annotation_dictionary &=
common_markup_annotation_dictionary
file_attachment_annotation_dictionary &=
  element File { file_spec_dictionary }
  & attribute IconName { name }?
  & attribute IconName_enc { token }?
```

Description

Contains a reference to a file, which typically is embedded in the PDF file.

Embedded file information is stored within the document backbone. File attachment annotations are considered markup annotations stored with other annotations associated with a page. The attached file itself is stored as a separate file within the document package.

For more information, see *File Attachment Annotations* in section 8.4.5 in the *PDF Reference*.

Element or attribute	Description	Default Value
File	See the `FS` entry in Table 8.35 in the *PDF Reference*.	
IconName	See the `Name` entry in Table 8.35 in the *PDF Reference*.	

File identifier

array_of_2_strings

```
array_of_2_strings =
  attribute PermanentId { pdf_byte_string }
  & attribute PermanentId_enc { token }?
  & attribute ContentId { pdf_byte_string }
  & attribute ContentId_enc { token }?
```

Description

An array of two strings constituting a file identifier. For more information, see the `ID` entry in Table 3.41 in the *PDF Reference*.

File specification dictionary

file_spec_dictionary

```
file_spec_dictionary = file_spec_dictionary1
file_spec_dictionary |=
   attribute Name { pdf_byte_string }
   & attribute Name_enc { token }?
```

file_spec_dictionary1

```
file_spec_dictionary1 =
   attribute FSType { pdf_byte_string }?
   & attribute FSType_enc { token }?
   & attribute Name { pdf_byte_string }
   & attribute Name_enc { token }?
   & attribute UName { pdf_text_string }?
   & element FileId { array_of_2_strings }?
   & attribute Volatile { boolean }?
file_spec_dictionary1 &= embedded_file_stream_information_dictionary?
file_spec_dictionary1 &=
   element RelatedFiles { file_spec_related_files }?
   & element Desc { pdf_text_string }?
   & element Collection { collection_item_dictionary_reference }?
```

Description

The dictionary form of file specification provides more flexibility than the string form, allowing different files to be specified for different file systems or platforms, or for file systems other than the standard ones (DOS/Windows®, Mac OS, and UNIX®). For more information, see section 3.10.2 in the *PDF Reference*.

Element or attribute	Description	Default Value
FSType	See the `FS` entry in Table 3.41 in the *PDF Reference*.	
FSType_enc	A PDFDocEncoded string representing the same information as the `FSType` attribute. For information on string encoding, see Section 3.8.1 of the *PDF Reference*.	
Name	See the `F` entry in Table 3.41 in the *PDF Reference*.	
Name_enc	A PDFDocEncoded string representing the same information as the `Name` attribute. For information on string encoding, see Section 3.8.1 of the *PDF Reference*.	
UName	See the `UF` entry in Table 3.41 in the *PDF Reference*.	
FileId	See the `ID` entry in Table 3.41 in the *PDF Reference*.	

Volatile	See the `V` entry in Table 3.41 in the *PDF Reference*.
RelatedFiles	See the `RF` entry in Table 3.41 in the *PDF Reference*.
Collection	See the `CI` entry in Table 3.41 in the *PDF Reference*.

Filters

filter_name_or_array

```
filter_name_or_array = element Filter {
  attribute _enc { token }?
  & name }*
```

Description

The name of a filter or an array of such filters.

Fixed number

fixed

```
fixed = number
```

Description

Real objects approximate mathematical real numbers, but with limited range and precision. For more information, see section 3.2.2 in the *PDF Reference*.

Fixed print dictionary

fixed_print_dictionary

```
fixed_print_dictionary =
  attribute Matrix { coordinate_map_array }?
  & attribute HorizontalTranslate { number }?
  & attribute VerticalTranslate { number }?
```

Description

A dictionary that specifies how the annotation should be drawn relative to the dimensions of the target media. For more information, see the *FixedPrint* entry in Table 8.41 in the *PDF Reference*.

Element or attribute	Description	Default Value
HorizontalTranslate	See the `H` entry in Table 8.42 in the *PDF Reference.*	0
VerticalTranslate	See the `V` entry in Table 8.42 in the *PDF Reference.*	0

Font descriptor

font_file

```
font_file =
  element Font { font_dictionary }
```

font_dictionary

```
font_dictionary =
  element Type0 { type0_font_dictionary }
font_dictionary |=
  element Type1 { type1_font_dictionary }
font_dictionary |=
  element TrueType { truetype_font_dictionary }
```

Font_descriptor_dictionary

```
Font_descriptor_dictionary =
  attribute FontName { name }
  & attribute FontName_enc { token }?
  & attribute FontFamily { pdf_byte_string }?
  & attribute FontFamily_enc { token }?
  & attribute FontStretch { name }?
  & attribute FontStretch_enc { token }?
  & attribute FontWeight { number }?
  & attribute Flags {  ( list {  "FixedPitch"? "Serif"? "Symbolic"?
  "Script"? "Nonsymbolic"? "Italic"? "AllCap"? "SmallCap"? "ForceBold"? } )
  }
  & attribute FontBBox { rectangle_blank_sep }
  & attribute ItalicAngle { number }
  & attribute Ascent { number }
  & attribute Descent { number }
  & attribute Leading { number }?
  & attribute CapHeight { number }
  & attribute XHeight { number }?
  & attribute StemV { number }
  & attribute StemH { number }?
  & attribute AvgWidth { number }?
  & attribute MaxWidth { number }?
  & attribute MissingWidth { number }?
  & attribute CharSet { pdf_byte_string }?
  & attribute CharSet_enc { token }?
```

Description

A font descriptor contains information about the font, including widths and Unicode mappings that are useful if a font substitution should be required. This information allows the document to be rendered with layout similar to that of the original. The Mars font descriptor files correspond to the information in Type 0, Type 1, Type 3, and TrueType font dictionaries in PDF documents.

Each font descriptor is a separate package level file. Font descriptor files may be in any non-reserved location in the package but by convention should be placed in either the /fonts directory or the directory for the page on which the font descriptor is used.

For more information, see section 5.9.2, Table 5.19, and Table 5.21 in the *PDF Reference*. Also, see *Referencing and Embedding Fonts* in the *Developing Applications Using Mars* guide.

Example

Sample Font Descriptor File Contents

```
<Font>
  <Type1 LastChar="255" BaseFont="Bembo" FirstChar="0">
    <FontDescriptor StemV="67" FontName="Bembo" Flags="Serif Nonsymbolic"
      Descent="-233" Ascent="673" CapHeight="622" XHeight="396"
      FontBBox="-206 -233 1006 896" />
    <Widths>278 278 278 ..... 677 594 677 594 </Widths>
  </Type1>
</Font>
```

Element or attribute	Description	Default Value
Overrides	See the `FD` entry in Table 5.21 in the *PDF Reference*.	

Font file

font_file

```
  font_file =
    element Font { font_dictionary }
```

Description

A font file. Each font consists of a font descriptor file, the font file itself, and possibly other files. The font descriptors are represented in XML and can be shared among pages in a Mars document. The fonts themselves are represented in OpenType format where possible.

Fonts are stored in separate binary files in the document package. By convention, they should be placed in the `/fonts` directory or in the directory of the page on which they are referenced.

Note that per the *PDF Reference*, a different tag (`FontFile`, `FontFile2`, or `FontFile3`) will be used to reference the actual font based on whether the font is Type 1, TrueType, or based on the font dictionary `Subtype` entry.

Example

```
<FontDescriptor StemV="67" FontName="Bembo" Flags="Serif Nonsymbolic"
  Descent="-233" Ascent="673" CapHeight="622" XHeight="396" >
  FontBBox="-206 -233 1006 896" />
```

Element or attribute	Description	Default Value
Font	The font dictionary.	

Form dictionary

form_dictionary

```
form_dictionary =
  attribute Name { name }?
  & attribute Name_enc { token }?
  & attribute LastModified { date }?
  & attribute BBox { rectangle_blank_sep }
  & attribute Matrix { coordinate_map_array }?
  & element Attributes { group_attributes_dictionary }?
  & element Ref { reference_dictionary }?
form_dictionary &= metadata_item
form_dictionary &=
  element ApplicationDatasets { page_piece_dictionary }?
  & element OPI { opi_version_dictionary }?
  & element ContentGroup { oc_group_or_membership_dictionary }?
  & element MarkStyle { pdf_text_string }?
  & element Colorants { colorant_dictionary }?
  & attribute PCM { name }?
  & attribute PCM_enc { token }?
  & element SeparationColorNames { array_of_name }?
  & element TrapRegions { trap_array_of_object_references }?
  & attribute TrapStyles { pdf_text_string }?
form_dictionary &= common_stream_dictionary
form_dictionary &=
  attribute src { string }
```

trap_array_of_object_references

```
trap_array_of_object_references =
  element Obj { attribute ref { pdf_reference } }+
```

metadata_item

```
metadata_item
```

Description

Every form XObject has a form type, which determines the format and meaning of the entries in its form dictionary. For more information, see section 4.9.1 and Tables 4.45, 10.49, and 10.53 in the *PDF Reference*.

Element or attribute	Description	Default Value
ApplicationDatasets	See the `PieceInfo` entry in Table 4.45 in the *PDF Reference.*	
ContentGroup	See the `OC` entry in Table 4.45 in the *PDF Reference.*	
src	References the file containing the SVG graphic content used in the appearance.	

Free text annotation

free_text_annotation_dictionary

```
free_text_annotation_dictionary = common_annotation_dictionary
free_text_annotation_dictionary &= common_markup_annotation_dictionary
free_text_annotation_dictionary &=
  element DefaultAppearance {
    attribute Font { text }
    attribute TextSize { number }?
    attribute FillColorSpace { "Gray" | "RGB" | "CMYK" }?
    attribute FillColor { number | ( list { number number numbernumber? } )
    }?
    attribute StrokeColorSpace { "Gray" | "RGB" | "CMYK" }?
    attribute StrokeColor { number | ( list { number number numbernumber? }
    ) }?
    attribute CharacterSpacing { number }?
    attribute TextLeading { number }?
    attribute TextMatrix { ( list { number number number number number
    number } ) }?
    attribute RenderMode { number }?
    attribute TextRise { number }?
    attribute WordSpacing { number }?
    attribute HorizontalScaling { number }? }
free_text_annotation_dictionary &=
  attribute Justification { "Left" | "Centered" | "Right" }?
  & attribute DefaultStyle { pdf_text_string }?
  & attribute StartPoint { ( list { number "," number } ) }
  & attribute KneePoint { ( list { number "," number } ) }?
  & attribute EndPoint { ( list { number "," number } ) }?
  & element BorderEffect { border_effect_dictionary }?
  & attribute Fringe { rectangle_blank_sep }?
  & attribute Style { "Square" | "Circle" | "Diamond" | "OpenArrow" |
  "ClosedArrow" | "None" | "Butt" | "ROpenArrow" | "RClosedArrow" |
  "Slash" }?
```

Description

A free text annotation displays text directly on the page. For more information, see *Free Text Annotations* in section 8.4.5 in the *PDF Reference.*

Element or attribute	Description	Default Value
DefaultAppearance	See the `DA` entry in Table 8.25 in the *PDF Reference*.	
Justification	See the `Q` entry in Table 8.25 in the *PDF Reference*.	"left"
DefaultStyle	See the `DS` entry in Table 8.25 in the *PDF Reference*.	
BorderEffect	See the `BE` entry in Table 8.25 in the *PDF Reference*.	
Fringe	See the `RD` entry in Table 8.25 in the *PDF Reference*.	
StartPoint	The starting coordinates of the line in default user space.	
KneePoint	The knee point coordinates of the line in default user space.	
EndPoint	The ending coordinates of the line in default user space.	

Function dictionary

function_dictionary

```
function_dictionary =
   element SampledFunction { sampled_function_dictionary }
function_dictionary |=
   element InterpolatedFunction { interpolated_function_dictionary }
function_dictionary |=
   element StitchingFunction { stitching_function_dictionary }
function_dictionary |=
   element PostscriptFunction { postscript_function_dictionary }
```

sampled_function_dictionary

```
sampled_function_dictionary =
   attribute Domain { array_of_number }
   & attribute Range { array_of_number }
   & attribute Size { array_of_integer }
   & attribute BitsPerSample { integer }
   & attribute Order { number }?
   & attribute Encode { array_of_number }?
   & attribute Decode { array_of_number }?
   & attribute src { string }
   & common_stream_dictionary
```

interpolated_function_dictionary

```
interpolated_function_dictionary =
   attribute Domain { array_of_number }
   & attribute Range { array_of_number }
   & attribute C0 { array_of_number }?
   & attribute C1 { array_of_number }?
   & attribute Exponent { number }
```

stitching_function_dictionary

```
stitching_function_dictionary =
   attribute Domain { array_of_number }
   & attribute Range { array_of_number }
   & element StitchingFunctions { array_of_function_dictionary }
   & attribute Bounds { array_of_number }
   & attribute Encode { array_of_number }
```

postscript_function_dictionary

```
postscript_function_dictionary =
   attribute Domain { array_of_number }
   & attribute Range { array_of_number }
   & attribute src { string }
   & common_stream_dictionary
```

Description

A static, self-contained numerical transformation. For more information, see section 3.9 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
SampledFunction	A sampled function. See the `FunctionType` entry in Table 3.34 in the *PDF Reference*.	
InterpolatedFunction	An exponential interpolation function. See the `FunctionType` entry in Table 3.34 in the *PDF Reference*.	
StitchingFunction	A stitching function. See the `FunctionType` entry in Table 3.34 in the *PDF Reference*.	
PostscriptFunction	A PostScript calculator function. See the `FunctionType` entry in Table 3.34 in the *PDF Reference*.	

Gamma value

gamma_value

```
gamma_value =  ( list { number number number } )
```

Description

Three numbers specifying the gamma for the red, green, and blue (A, B, and C) components of the color space. For more information, see the `Gamma` entry in Table 4.14 in the *PDF Reference*.

Go-to action

goto_action_dictionary

```
goto_action_dictionary =
   element Dest { destination }
goto_action_dictionary &= common_action_dictionary
```

Description

A go-to action changes the view to a specified destination (page, location, and magnification factor). For more information, see *Go-To Actions* in section 8.5.3 in the *PDF Reference* and in *Representing Actions* in the guide *Developing Applications Using Mars*.

Example

```
<!-- A GoTo action that provides an explicit destination-- >
<Action>
   <GoTo>
      <Dest>
         <XYZ Left="72" Top="205" Zoom="" Page_ref="/backbone.xml#0"/>
      </Dest>
   </GoTo>
</Action>

<!-- A GoTo action that provides a named destination-->
<Action>
   <GoTo>
      <Dest Name="F2" />
   </GoTo>
</Action>
```

Element or attribute	Description	Default Value
Dest	See the `D` entry in Table 8.49 in the *PDF Reference*.	

Go-to-3D-view action

goto_3d_view_action_dictionary

```
goto_3d_view_action_dictionary =
   element Target { attribute ref { pdf_reference } }
   & element View { three_d_view }
goto_3d_view_action_dictionary &= common_action_dictionary
```

Description

A go-to-3D-view action identifies a 3D annotation and specifies a view for the annotation to use. For more information, see *Go-To-3D-View Actions* in section 8.5.3 in the *PDF Reference*.

Element or attribute	Description	Default Value
Target	See the `TA` entry in Table 8.66 in the *PDF Reference*.	
View	See the `V` entry in Table 8.66 in the *PDF Reference*.	

Group attributes dictionary

group_attributes_dictionary

```
group_attributes_dictionary =
   attribute Type { name }
   & attribute Type_enc { token }?
   & element ColorSpace { color_space }?
   & attribute Isolated { boolean }?
   & attribute Knockout { boolean }?
```

Description

Describes the properties of a group of graphical elements. For more information, see section 4.9.2 in the *PDF Reference*.

Element or attribute	Description	Default Value
Type	See the `S` entry in Tables 4.46 and 7.13 in the *PDF Reference*.	
ColorSpace	See the `CS` entry in Table 7.13 in the *PDF Reference*.	
Isolated	See the `I` entry in Table 7.13 in the *PDF Reference*.	
Knockout	See the `K` entry in Table 7.13 in the *PDF Reference*.	

Hide action

hide_action_dictionary

```
hide_action_dictionary =
   element Target { hide_action_target }
   & attribute Hide { boolean }?
hide_action_dictionary &= common_action_dictionary
```

Description

A hide action hides or shows one or more annotations on the screen. For more information, see *Hide Actions* in section 8.5.3 in the *PDF Reference*.

Element or attribute	Description	Default Value
Target	See the `T` entry in Table 8.60 in the *PDF Reference*.	
Hide	See the `H` entry in Table 8.60 in the *PDF Reference*.	true

Hide action target

hide_action_target

```
hide_action_target =
   element Field { attribute Pathname { pdf_text_string } }
hide_action_target |=
   element Annotation { attribute ref { pdf_reference } }
hide_action_target |=
   element Targets { hide_action_target_array }
```

hide_action_target_array

```
hide_action_target_array = string_or_annotation_dictionary_reference*
```

Description

The annotation or annotations to be hidden or shown in a Hide Action. For more information, see the `T` entry in Table 8.56 in the *PDF Reference*.

Element or attribute	Description	Default Value
Pathname	A string giving the fully qualified field name of an interactive form field whose associated widget annotation or annotations are to be affected.	
Annotation	An indirect reference to an annotation dictionary.	
Targets	An array of dictionaries (`AnnotationRef`) or strings (`Pathname`) representing the annotation or annotations to be hidden or shown.	

ICCBased color space

ICC_profile_stream_dictionary

```
ICC_profile_stream_dictionary =
  attribute Count { integer }
  & element Alternate { color_space }?
  & attribute Range { array_of_number }?
ICC_profile_stream_dictionary &= metadata_item
ICC_profile_stream_dictionary &=
  attribute src { string }
  & common_stream_dictionary
```

Description

ICCBased color spaces are based on a cross-platform color profile as defined by the International Color Consortium (ICC). For more information, see *ICCBased Color Spaces* in section 4.5.4 in the *PDF Reference*.

Icon fit dictionary

icon_fit_dictionary

```
icon_fit_dictionary =
  attribute ScaleCondition {  "Always" |  "IconBigger" |  "IconSmaller" |
  "Never" }?
  & attribute Scaling {  "Anamorphic" |  "Proportional" }?
  & element BottomLeftSpace { space_fraction_array }?
  & attribute FitWithinBounds { boolean }?
```

space_fraction_array

```
space_fraction_array =
  attribute XFraction { number }
  & attribute YFraction { number }
```

Description

A dictionary specifying how to display a button's icon within the annotation rectangle of its widget annotation. For more information, see *FDF Fields* in section 8.6.6 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
ScaleCondition	See the `SW` entry in Table 8.97 in the *PDF Reference*.	Always
Scaling	See the `S` entry in Table 8.97 in the *PDF Reference*.	Proportional

BottomLeftSpace	See the `A` entry in Table 8.97 in the *PDF Reference*.	
FitWithinBounds	See the `FB` entry in Table 8.97 in the *PDF Reference*.	false

Image dictionary

image_dictionary

```
image_dictionary =
  attribute Width { integer }
  & attribute Height { integer }
  & element ColorSpace { color_space }
  & attribute BitsPerComponent { integer }
  & attribute Intent { name }?
  & attribute Intent_enc { token }?
  & attribute ImageMask { boolean }?
  & element Mask { image_mask }?
  & element SMask { soft_mask_image_dictionary }?
  & attribute SMaskInData { integer }?
  & element ColorSpaceMap { color_space_map }?
  & attribute Interpolate { boolean }?
  & element Alternates { array_of_alternate_image_dictionary }?
  & attribute Name { name }?
  & attribute Name_enc { token }?
  & attribute WebCaptureId { pdf_base64_string }?
  & element OPI { opi_version_dictionary }?
  & element ContentGroup { oc_group_or_membership_dictionary }?
image_dictionary &= metadata_item
image_dictionary &=
  attribute src { string }
  & common_stream_dictionary
```

Description

The dictionary portion of a stream representing an image XObject. Images are stored in separate files in the document package. These files may be in any non-reserved area of the package directory structure, but, by convention, should only appear in the `/images` directory or the page directory for the page on which the image is referenced.

Images in PDF documents are stored in streams and compressed using one or more compression filters appropriate for the type of image data. Images in Mars documents are stored using a standard representation. Inline images are not supported in Mars. They are converted to external images. It is important to round-trip the actual image data faithfully. In particular: lossless image encodings should round-trip exactly, and lossy image encodings should round-trip without introducing additional loss.

For more information, see section 4.8.4 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
ColorSpaceMap	See the `Decode` entry in Table 4.39 in the *PDF Reference*.	
WebCaptureId	See the `ID` entry in Table 4.39 in the *PDF Reference*.	
ContentGroup	See the `OC` entry in Table 4.39 in the *PDF Reference*.	

Image mask

image_mask

```
image_mask =
  element Image { image_dictionary }
image_mask |= element ColorRangeMask { integer }*
```

Description

An image XObject defining an image mask to be applied to this image, or an array specifying a range of colors to be applied to it as a color key mask. For more information, see section 4.8.5 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
Image	An image XObject to be used as an explicit mask specifying which areas of the image to paint and which to mask out. For more information, see section 4.8.5 in the *PDF Reference*.	
ColorRangeMask	An array specifying a range of colors to be masked out. Samples in the image that fall within this range are not painted, allowing the existing background to show through. For more information, see *Color Key Masking* in section 4.8.5 in the *PDF Reference*.	

Image set reference counts

ref_integer_or_array

```
ref_integer_or_array =
  attribute Count { integer }
ref_integer_or_array |= element Count { integer }*
```

Description

The reference counts for the image XObjects belonging to the image set. For more information, see the `R` entry in Table 10.40 in the *PDF Reference*.

Import-data action

importdata_action_dictionary

```
importdata_action_dictionary =
  element File { file_spec_dictionary }
importdata_action_dictionary &= common_action_dictionary
```

Description

An import-data action imports Forms Data Format (FDF) data into the document's interactive form from a specified file. For more information, see *Import-Data Actions* in section 8.6.4 in the *PDF Reference*.

Element or attribute	Description	Default Value
File	See the `F` entry in Table 8.89 in the *PDF Reference*.	

Indexed color space

indexed_color_space

```
indexed_color_space =
  element Base { color_space }
  & attribute HiVal { integer }
  & element LookupTable { lut_string_or_stream }
```

lut_string_or_stream

```
lut_string_or_stream = pdf_base64_string
lut_string_or_stream |=
  attribute src { string }
  & common_stream_dictionary
```

Description

An Indexed color space allows a content stream to use small integers as indices into a color map or color table of arbitrary colors in some other space. For more information, see *Indexed Color Spaces* in section 4.5.5 in the *PDF Reference*.

Element or attribute	Description	Default Value
Base	An array or name that identifies the base color space in which the values in the color table are to be interpreted.	
HiVal	An integer that specifies the maximum valid index value.	

LookupTable	A stream or a string that provides the mapping between index values and the corresponding colors in the base color space.
src	

Initial 3D view

initial_3d_view

```
initial_3d_view =
  element View { three_d_view_dictionary }
initial_3d_view |=
  attribute ViewNumber { integer }
initial_3d_view |=
  attribute ViewName { pdf_text_string }
initial_3d_view |=
  attribute ViewKey { name }
  & attribute ViewKey_enc { token }?
```

Description

An object that specifies the default initial view of the 3D artwork that should be used when the annotation is activated. For more information, see the *3DV* entry in Table 9.33 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
View		
ViewNumber	An integer specifying an index into the `Views` array.	
ViewName	A text string matching the `InternalName` entry in one of the views in the `Views` array.	
ViewKey	A name that indicates the first (`F`), last (`L`), or default (`D`) entries in the `Views` array.	

Ink annotation

ink_annotation_dictionary

```
ink_annotation_dictionary = common_annotation_dictionary
ink_annotation_dictionary &= common_markup_annotation_dictionary
ink_annotation_dictionary &=
  element Inklist { array_of_path }
```

array_of_path

```
array_of_path = element Gesture { attribute Points { path } }+
```

path

```
path =  ( list { number } )
```

Description

Represents a freehand *scribble* composed of one or more disjoint paths. For more information, see *Ink Annotations* in section 8.4.5 in the *PDF Reference*.

Integer object

integer

```
integer = integer
```

Description

Integer objects represent mathematical integers within a certain interval centered at `0`. For more information, see section 3.2.2 in the *PDF Reference*.

Interactive form dictionary

acroform_dictionary

```
acroform_dictionary =
   element Fields { array_of_field_dictionary }
   & attribute NeedAppearances { boolean }?
   & element CalculationOrder { array_of_field_dictionary_reference }?
   & element DefaultAppearance {
      attribute Font { text }
      attribute TextSize { number }?
      attribute FillColorSpace {  "Gray" |  "RGB" |  "CMYK" }?
      attribute FillColor{ number |( list { number number number number? })
      }?
      attribute StrokeColorSpace {  "Gray" |  "RGB" |  "CMYK" }?
      attribute StrokeColor { number |(list{ number number number number?}
      )}?
      attribute CharacterSpacing { number }?
      attribute TextLeading { number }?
      attribute TextMatrix {  ( list { number number number number number
      number } ) }?
      attribute RenderMode { number }?
      attribute TextRise { number }?
      attribute WordSpacing { number }?
      attribute HorizontalScaling { number }?
   }
   & attribute Justification {  "Left" |  "Centered" |  "Right" }?
   & element XFA { stream_or_array_of_XFA }?
```

array_of_field_dictionary_reference

```
array_of_field_dictionary_reference =
  element Field { attribute ref { pdf_reference } }+
```

Description

The contents and properties of a document's interactive form are defined by an interactive form dictionary that is referenced from the `AcroForm` entry in the document catalog.

Form fields in Mars documents consist of three parts: some base information in the `Catalog` dictionary (the `Acroform` dictionary), form fields (each of which is represented by a field dictionary), and form field widgets (each of which is represented by a widget annotation dictionary). Base information about the form is in the `AcroForm` element within the document backbone. In addition to general information about the form, this element includes elements representing each form field. The form field widgets, representing the visible manifestations of the fields that are displayed on the page(s) of the document, are stored in the `content.ann` file associated with each page.

In PDF documents, the widget annotation dictionary and field dictionary can be shared if there is a one-to-one correspondence between them. Such sharing is not done in Mars documents.

The field elements refer to the widget annotation elements using an identifier reference scheme. This crosses files as the fields are in the backbone and the widgets are in annotation files associated with the pages on which they appear.

The widgets are likely to refer to appearance streams, which are SVG content files that describe how to render them. Conventionally, these SVG files are separate package objects that should be in the page directory (if they are used only on that page) or in the `/res` directory (if they are used on multiple pages).

For more information, see section 8.6.1 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
CalcOrder	See the `CO` entry in Table 8.67 in the *PDF Reference*.	
Resources	See the `DR` entry in Table 8.67 in the *PDF Reference*.	
DefaultAppearance	See the `DA` entry in Table 8.67 in the *PDF Reference*.	

JavaScript action name tree

javascript_action_name_tree

```
javascript_action_name_tree =
  element JavaScript {
  attribute Key { text }
  & javascript_action_dictionary }
```

Description

An element that maps name strings to document level JavaScript actions. For more information, see *JavaScript Actions* in section 8.6.4, as well as the `JavaScript` entry in Table 3.28 in the *PDF Reference*.

Element or attribute	Description	Default Value
Script	See the `JS` entry in Table 8.90 in the *PDF Reference*.	

JavaScript action

javascript_action_dictionary

```
javascript_action_dictionary =
   element Script { javascript_action_js }
javascript_action_dictionary &= common_action_dictionary
```

javascript_action_js

```
javascript_action_js = pdf_text_string
javascript_action_js |=
   attribute src { string }
   & common_stream_dictionary
```

Description

A JavaScript action causes a script to be compiled and executed by the JavaScript interpreter.

JavaScript actions consist of strings representing JavaScript code. The JavaScript code can generally appear inline in a <JavaScript> element, or the <JavaScript> element can reference a separate package-level file which contains the JavaScript source code.

JavaScript actions can appear in several places in a Mars document, including anywhere a predefined action can appear, and as a child of various event elements indicating what should happen for specific document and form-related events.

XML forms also define a rich JavaScript programming environment. Scripts related to XML forms are independent of document-related JavaScript actions and Acrobat form-related (non-XML form-related) JavaScript actions.

For more information, see *JavaScript Actions* in section 8.6.4 in the *PDF Reference*.

Element or attribute	Description	Default Value
Script	See the `JS` entry in Table 8.90 in the *PDF Reference*.	

JavaScript file

javascripts_file

```
javascripts_file =
  element JavaScripts { javascript_action_name_tree* }
```

Description

The JavaScript file contains named JavaScript segments that can be referenced from JavaScript actions.

Lab color space

Lab_color_space_dictionary

```
Lab_color_space_dictionary =
  attribute WhitePoint { tristimulus_value }
  & attribute BlackPoint { tristimulus_value }?
  & attribute Range { lab_color_range }?
```

lab_color_range

```
lab_color_range =  ( list { number number number number } )
```

Description

A Lab color space is a CIE-based ABC color space with two transformation stages (see Figure 4.14 in the *PDF Reference*). In this type of space, A, B, and C represent the L*, a*, and b* components of a CIE 1976 L*a*b* space. For more information, see *Lab Color Spaces* in section 4.5.4 in the *PDF Reference*.

Language identifiers

array_of_language_identifiers

```
array_of_language_identifiers = element Language { pdf_text_string }*
```

Description

A text string that specifies the language. A language identifier consists of a primary code optionally followed by one or more subcodes. For more information, see *Language Identifiers* in section 10.8.1 in the *PDF Reference*.

Launch action

launch_action_dictionary

```
launch_action_dictionary =
  element File { file_spec_dictionary }?
  & element Win { windows_launch_dictionary }?
  & attribute NewWindow { boolean }?
launch_action_dictionary &= common_action_dictionary
```

Description

A launch action launches an application or opens or prints a document. For more information, see *Launch Actions* in section 8.5.3 in the *PDF Reference*.

Element or attribute	Description	Default Value
File	See the `F` entry in Table 8.53 in the *PDF Reference*.	

Legal attestation dictionary

legal_dictionary

```
legal_dictionary =
  attribute JavaScriptActions { integer }?
  & attribute LaunchActions { integer }?
  & attribute URIActions { integer }?
  & attribute MovieActions { integer }?
  & attribute SoundActions { integer }?
  & attribute HideAnnotationActions { integer }?
  & attribute GoToRemote { integer }?
  & attribute AlternateImages { integer }?
  & attribute ExternalStreams { integer }?
  & attribute TrueTypeFonts { integer }?
  & attribute ExternalRefXObjects { integer }?
  & attribute ExternalOPIdicts { integer }?
  & attribute NonEmbeddedFonts { integer }?
  & attribute DevDepGS_OP { integer }?
  & attribute DevDepGS_HT { integer }?
  & attribute DevDepGS_TR { integer }?
  & attribute DevDepGS_UCR { integer }?
  & attribute DevDepGS_BG { integer }?
  & attribute DevDepGS_FL { integer }?
  & attribute Annotations { integer }?
  & attribute OptionalContent { boolean }?
  & attribute Attestation { pdf_text_string }?
```

Description

A dictionary that specifies all content that may result in unexpected rendering of the document contents. For more information, see section 8.7.4 in the *PDF Reference*.

Line annotation dictionary

line_annotation_dictionary

```
line_annotation_dictionary = common_annotation_dictionary
line_annotation_dictionary &= common_markup_annotation_dictionary
line_annotation_dictionary &=
  attribute Start {  ( list { number number } ) }
  & attribute End {  ( list { number number } ) }
line_annotation_dictionary &= line_ends_array?
line_annotation_dictionary &= attribute InteriorColor { rgb_color_array }?
line_annotation_dictionary &= number_or_array_of_2_numbers?
line_annotation_dictionary &=
  attribute LeaderLineExtension { number }?
  & attribute Caption { boolean }?
  & attribute LeaderLineOffset { number }?
  & attribute CaptionPosition { pdf_ascii_string }?
  & element Measure { measure_dictionary }?
line_annotation_dictionary &= caption_offset_array?
```

line_ends_array

```
line_ends_array = attribute HeadStyle {  "Square" |  "Circle" |  "Diamond" |
"OpenArrow" |  "ClosedArrow" |  "None" |  "Butt" |  "ROpenArrow" |
"RClosedArrow" |  "Slash" }?
line_ends_array &= attribute TailStyle {  "Square" |  "Circle" |  "Diamond" |
"OpenArrow" |  "ClosedArrow" |  "None" |  "Butt" |  "ROpenArrow" |
"RClosedArrow" |  "Slash" }?
```

number_or_array_of_2_numbers

```
number_or_array_of_2_numbers =
  attribute LeaderLineLength { number }
```

caption_offset_array

```
caption_offset_array =
  attribute CaptionOffsetX { number }
  & attribute CaptionOffsetY { number }
```

Description

Displays a single straight line on the page. For more information, see *Line Annotations* in section 8.4.5 in the *PDF Reference*.

Element or attribute	Description	Default Value
Start	See the `L` entry in Table 8.26 in the *PDF Reference*.	
InteriorColor	See the `IC` entry in Table 8.26 in the *PDF Reference*.	

LeaderLineLength	See the `LL` entry in Table 8.26 in the *PDF Reference*.	0
LeaderLineExtension_Length	See the `LLE` entry in Table 8.26 in the *PDF Reference*.	0
Caption	See the `Cap` entry in Table 8.26 in the *PDF Reference*.	false
LeaderLineOffset	See the `LLO` entry in Table 8.26 in the *PDF Reference*.	
CaptionPosition	See the `CP` entry in Table 8.26 in the *PDF Reference*.	Inline

Line dash pattern

line_dash_pattern_array

```
line_dash_pattern_array = dash_array
```

dash_array

```
dash_array =
  attribute On { number }
  & attribute Off { number }
```

Description

The line dash pattern controls the pattern of dashes and gaps used to stroke paths. For more information, see *Line Dash Pattern* in section 4.3.2 in the *PDF Reference*.

Element or attribute	Description	Default Value
Phase	The dash phase, which specifies the distance into the dash pattern at which to start the dash.	
On	The length of a dash within the dash pattern.	
Off	The length of a gap within the dash pattern.	

Linear interpretation array

linear_interpretation_array

```
linear_interpretation_array =  ( list { number number number number number
number number number number } )
```

Description

For a CalRGB color space dictionary, a linear interpretation array is comprised of nine numbers specifying the linear interpretation of the decoded A, B, and C components of the color space with respect to the final XYZ representation.

For a type 1 form dictionary, it is a matrix of six numbers specifying the form matrix, which maps form space into user space.

For a type 1 or type 2 pattern dictionary, it is a matrix of six numbers specifying the pattern matrix.

For a type 1 shading dictionary, it is a matrix of six numbers specifying a transformation matrix mapping the coordinate space specified by the Domain entry into the shading's target coordinate space.

For more information, see the `Matrix` entries in Tables 4.14, 4.25, 4.26, 4.29, and 4.45 in the *PDF Reference*.

Link annotation

link_annotation_dictionary

```
link_annotation_dictionary = common_annotation_dictionary
link_annotation_dictionary &=
  attribute Highlight {  "None" |  "Invert" |  "Outline" |  "Push" }?
  & element OriginalURI { uri_action_dictionary }?
  & attribute Coords { quadpoints_array }?
  & element OnActivation { action_dictionary }?
```

quadpoints_array

```
quadpoints_array =  ( list { number } )
```

Description

A link annotation represents either a hypertext link to a destination elsewhere in the document, or an action to be performed. For more information, see *Link Annotations* in section 8.4.5 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
OnActivation	See the `A` entry in Table 8.24 in the *PDF Reference*.	
Highlight	See the `H` entry in Table 8.24 in the *PDF Reference*.	"Invert"
OriginalURI	See the `PA` entry in Table 8.24 in the *PDF Reference*.	
Coords	See the `QuadPoints` entry in Table 8.24 in the *PDF Reference*.	

Mac OS file information dictionary

embedded_file_stream_parameter_mac_dictionary

```
embedded_file_stream_parameter_mac_dictionary =
  attribute FileType { integer }?
  & attribute Creator { integer }?
  & element ResFork { embedded_file_stream_dictionary }?
```

Description

For Mac OS files, the `Mac` entry in the embedded file parameter dictionary holds a further subdictionary containing Mac OS specific file information. For more information, see section 3.10.3 and Table 3.43 in the *PDF Reference.*

Element or attribute	**Description**	**Default Value**
FileType	See the `Subtype` entry in Table 3.43 in the *PDF Reference.*	

Mark information dictionary

mark_information_dictionary

```
mark_information_dictionary =
  attribute TaggedPDF { boolean }?
  & attribute UserProperties { boolean }?
  & attribute Suspects { boolean }?
```

Description

Contains information about the document's usage of tagged PDF conventions. For more information, see section 10.6 in the *PDF Reference* and the section *Specifying Marked Content, Logical Structure and Tagging* in the guide *Developing Applications Using Mars.*

Element or attribute	**Description**	**Default Value**
TaggedPDF	See the `Marked` entry in Table 10.8 in the *PDF Reference.*	false

Markup annotations

Description

See Annotations, markup (file).

Measure dictionary

measure_dictionary

```
measure_dictionary = rectilinear_measure_dictionary
```

rectilinear_measure_dictionary

```
rectilinear_measure_dictionary =
   attribute ScaleRatio { pdf_text_string }
   & element X { number_format_array }
   & element Y { number_format_array }?
   & element Distance { number_format_array }
   & element Area { number_format_array }
   & element Angle { number_format_array }?
   & element Slope { number_format_array }?
   & element Origin { origin_array }?
   & attribute YtoXFactor { number }?
```

Description

A measure dictionary specifies an alternate coordinate system for a region of a page. Along with the viewport dictionary, it provides the information needed to convert coordinates in the page's coordinate system to coordinates in the measuring coordinate system. For more information, see section 8.8 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
ScaleRatio	See the `R` entry in Table 8.111 in the *PDF Reference*.	
Distance	See the `D` entry in Table 8.111 in the *PDF Reference*.	
Area	See the `A` entry in Table 8.111 in the *PDF Reference*.	
Angle	See the `T` entry in Table 8.111 in the *PDF Reference*.	
Slope	See the `S` entry in Table 8.111 in the *PDF Reference*.	
Origin	See the `O` entry in Table 8.111 in the *PDF Reference*.	
YtoXFactor	See the `CYX` entry in Table 8.111 in the *PDF Reference*.	

Media clip data dictionary

media_clip_data_dictionary

```
media_clip_data_dictionary = common_media_clip_dictionary
media_clip_data_dictionary &=
  element Data { file_spec_or_form_XObject }
  & attribute ContentType { pdf_ascii_string }?
  & element Permissions { media_permissions_dictionary }?
  & element AlternateText { multi_language_text_array }?
  & element Players { media_players_dictionary }?
media_clip_data_dictionary &= common_must_honor_best_effort
```

file_spec_or_form_XObject

```
file_spec_or_form_XObject =
  element File { file_spec_dictionary }
file_spec_or_form_XObject |=
  element Form { form_dictionary }
```

Description

A media clip data dictionary defines the data for a media object that can be played. For more information, see *Media Clip Data Dictionary* in section 9.1.3 in the *PDF Reference*.

Element or attribute	Description	Default Value
Data	See the `D` entry in Table 9.9 in the *PDF Reference*.	
ContentType	See the `CT` entry in Table 9.9 in the *PDF Reference*.	
Permissions	See the `P` entry in Table 9.9 in the *PDF Reference*.	
AlternateText	See the `Alt` entry in Table 9.9 in the *PDF Reference*.	
Players	See the `PL` entry in Table 9.9 in the *PDF Reference*.	

Media clip dictionary

media_clip_dictionary

```
media_clip_dictionary = media_clip_data_dictionary
media_clip_dictionary |= media_clip_section_dictionary
```

Description

A media clip section or media clip data object. For more information, see section 9.1.3 in the *PDF Reference*.

Media clip section dictionary

media_clip_section_dictionary

```
media_clip_section_dictionary = common_media_clip_dictionary
media_clip_section_dictionary &=
  element Data { media_clip_dictionary }
  & element AlternateText { multi_language_text_array }?
media_clip_section_dictionary &= common_must_honor_best_effort
```

Description

Defines a continuous section of another media clip object. For more information, see *Media Clip Section* in section 9.1.3 in the *PDF Reference*.

Element or attribute	Description	Default Value
Data	See the `D` entry in Table 9.12 in the *PDF Reference*.	
AlternateText	See the `Alt` entry in Table 9.12 in the *PDF Reference*.	

Media criteria dictionary

media_criteria_dictionary

```
media_criteria_dictionary =
  attribute Audio { boolean }?
  & attribute Captions { boolean }?
  & attribute OverDubs { boolean }?
  & attribute Subtitles { boolean }?
  & attribute MinBandwidth { integer }?
  & element MinimumBitDepth { minimum_bit_depth_dictionary }?
  & element MinimumScreenSize { minimum_screen_size_dictionary }?
  & element AllowableViewers { array_of_software_identifier_dictionaries }?
  & element PDFVersion { array_of_PDF_versions }?
  & element AllowableLanguages { array_of_language_identifiers }?
```

Description

A media criteria dictionary behaves somewhat differently than other multimedia viability dictionary entries. The criteria specified by all of its entries must be met regardless of whether they are in an `Required` or a `BestEffort` dictionary.

Element or attribute	Description	Default Value
Audio	See the `A` entry in Table 9.3 in the *PDF Reference*.	
Captions	See the `C` entry in Table 9.3 in the *PDF Reference*.	
OverDubs	See the `O` entry in Table 9.3 in the *PDF Reference*.	

Subtitles	See the `S` entry in Table 9.3 in the *PDF Reference*.
MinBandwidth	See the `R` entry in Table 9.3 in the *PDF Reference*.
MinimumBitDepth	See the `D` entry in Table 9.3 in the *PDF Reference*.
MinimumScreenSize	See the `Z` entry in Table 9.3 in the *PDF Reference*.
AllowableViewers	See the `V` entry in Table 9.3 in the *PDF Reference*.
PDFVersion	See the `P` entry in Table 9.3 in the *PDF Reference*.
AllowableLanguages	See the `L` entry in Table 9.3 in the *PDF Reference*.

Media permissions dictionary

media_permissions_dictionary

```
media_permissions_dictionary = attribute TempFile {  "TEMPNEVER" |
"TEMPEXTRACT" |  "TEMPACCESS" |  "TEMPALWAYS" }?
```

Description

Specifies the manner in which the data referenced by the media may be used by a viewer application. For more information, see *Media Clip Data* in section 9.1.3 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
TempFile	See the `TF` entry in Table 9.10 in the *PDF Reference*.	"TEMPNEVER"

Media play parameters dictionary

media_play_parameters_dictionary

```
media_play_parameters_dictionary = element Players {
media_players_dictionary }?
media_play_parameters_dictionary &= common_must_honor_best_effort
```

Description

Specifies how a media object should be played. For more information, see section 9.1.4 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
Players	See the `PL` entry in Table 9.14 in the *PDF Reference*.	

Media player info dictionaries

media_player_info_array

```
media_player_info_array = element Player { media_player_info_dictionary }*
```

media_player_info_dictionary

```
media_player_info_dictionary =
  element PlayerId { software_identifier_dictionary }
media_player_info_dictionary &= common_must_honor_best_effort
```

Description

An array of media player info objects, referenced by the media players dictionary, that provide specific information about each player. A media player info dictionary provides a variety of information regarding a specific media player. Its entries allow information to be associated with a particular version or range of versions of a player. For more information, see *Media Players Dictionary* and *Media Player Info Dictionary* in section 9.1.6 in the *PDF Reference*.

Element or attribute	Description	Default Value
PlayerId	See the `PID` entry in Table 9.26 in the *PDF Reference*.	

Media players dictionary

media_players_dictionary

```
media_players_dictionary =
  element RequiredPlayers { media_player_info_array }?
  & element AllowedPlayers { media_player_info_array }?
  & element ForbiddenPlayers { media_player_info_array }?
```

Description

A media players dictionary can be referenced by media clip data and media play parameters dictionaries, and allows them to specify which players may or may not be used to play the associated media. For more information, see *Media Players Dictionary* in section 9.1.6 in the *PDF Reference*.

Element or attribute	Description	Default Value
RequiredPlayers	See the `MU` entry in Table 9.25 in the *PDF Reference*.	
AllowedPlayers	See the `A` entry in Table 9.25 in the *PDF Reference*.	
ForbiddenPlayers	See the `NU` entry in Table 9.25 in the *PDF Reference*.	

Media rendition

media_rendition_dictionary

```
media_rendition_dictionary = common_rendition_dictionary
media_rendition_dictionary &=
   element Clip { media_clip_dictionary }?
   & element PlayParameters { media_play_parameters_dictionary }?
   & element PlayLocation { media_screen_parameters_dictionary }?
```

Description

A media rendition dictionary. Its entries specify what media should be played (`Clip`), how (`PlayParameters`), and where (`PlayLocation`) it should be played. For more information, see *Media Renditions* in section 9.1.2 in the *PDF Reference*.

Element or attribute	Description	Default Value
Clip	See the `C` entry in Table 9.6 in the *PDF Reference*.	
PlayParameters	See the `P` entry in Table 9.6 in the *PDF Reference*.	
PlayLocation	See the `SP` entry in Table 9.6 in the *PDF Reference*.	

Media screen parameters dictionary

media_screen_parameters_dictionary

```
media_screen_parameters_dictionary = common_must_honor_best_effort
```

Description

Specifies where a media object should be played. For more information, see section 9.1.5 in the *PDF Reference*.

Minimum bit depth dictionary

minimum_bit_depth_dictionary

```
minimum_bit_depth_dictionary =
   attribute Depth { integer }
   & attribute Monitor { integer }?
```

Description

A dictionary specifying the minimum bit depth required in order for this object to be viable. For more information, see the `D` entry in Table 9.3 in the *PDF Reference*.

Element or attribute	Description	Default Value
Depth	See the `V` entry in Table 9.4 in the *PDF Reference*.	
Monitor	See the `M` entry in Table 9.4 in the *PDF Reference*.	

Minimum screen size dictionary

minimum_screen_size_dictionary

```
minimum_screen_size_dictionary = width_height_array
minimum_screen_size_dictionary &= attribute Monitor { integer }?
```

width_height_array

```
width_height_array =
   attribute Width { integer }
   & attribute Height { integer }
```

Description

Specifies the minimum screen size for a rendition object. For more information, see Table 9.5 in the *PDF Reference*.

Element or attribute	Description	Default Value
Monitor	See the `M` entry in Table 9.5 in the *PDF Reference*.	0
Width	The width of the monitor. See the `V` entry in Table 9.5 in the *PDF Reference*.	
Height	The height of the monitor. See the `V` entry in Table 9.5 in the *PDF Reference*.	

Movie action

movie_action_dictionary

```
movie_action_dictionary =
   element Annot { attribute ref { pdf_reference } }?
   & attribute Title { pdf_text_string }?
   & attribute Operation { pdf_text_name }?
movie_action_dictionary &= common_action_dictionary
movie_action_dictionary &= movie_activation_dictionary
```

Description

A movie action can be used to play a movie in a floating window or within the annotation rectangle of a movie annotation. For more information, see *Movie Actions* in section 8.5.3 in the *PDF Reference*.

Element or attribute	Description	Default Value
Annot	See the `Annotation` entry in Table 8.59 in the *PDF Reference.*	
Title	See the `T` entry in Table 8.59 in the *PDF Reference.*	

Movie annotation activation

movie_annotation_activation

```
movie_annotation_activation =
   element Start { boolean }
movie_annotation_activation |=
   element Activation { movie_activation_dictionary }
```

movie_activation_dictionary

```
movie_activation_dictionary =
   element Start { movie_time }?
   & element Duration { movie_time }?
   & attribute Rate { number }?
   & attribute Volume { number }?
   & attribute ShowControls { boolean }?
   & attribute Mode { pdf_text_name }?
   & attribute Synchronous { boolean }?
   & element FWScale { scale_ratio_array }?
   & element FWPosition { relative_position_array }?
```

scale_ratio_array

```
scale_ratio_array =
   attribute Numerator { number }
   & attribute Denominator { number }
```

relative_position_array

```
relative_position_array =
   attribute XFraction { number }
   & attribute YFraction { number }
```

movie_time

```
movie_time =
   attribute Scale { integer }?
   & attribute Time { integer }
movie_time |=
   attribute Time64 { pdf_base64_string }
```

Description

A flag or dictionary specifying whether and how to play the movie when the movie annotation is activated. For more information, see the `A` entry in Table 8.37, as well as Table 9.31, in the *PDF Reference*.

Movie annotation dictionary

movie_annotation_dictionary

```
movie_annotation_dictionary = common_annotation_dictionary
movie_annotation_dictionary &= movie_dictionary
movie_annotation_dictionary &=
   element Play { movie_annotation_activation }?
   & attribute Title { pdf_text_string }?
```

Description

A dictionary that contains animated graphics and sound to be presented on the computer screen and through the speakers. For more information, see *Movie Annotations* in section 8.4.5 in the *PDF Reference*.

Element or attribute	Description	Default Value
Play	See the `A` entry in Table 8.37 in the *PDF Reference*.	<Start>true</Start>
Title	See the `T` entry in Table 8.37 in the *PDF Reference*.	

Movie dictionary

movie_dictionary

```
movie_dictionary =
   element File { file_spec_dictionary1 }
   & element Aspect { width_height_array }?
   & attribute Rotate { integer }?
   & element Poster { movie_dictionary_poster }?
```

movie_dictionary_poster

```
movie_dictionary_poster =
   element Image { image_dictionary }
movie_dictionary_poster |=
   attribute Show { boolean }
```

Description

Describes the static characteristics of the movie. For more information, see section 9.3 in the *PDF Reference*.

Element or attribute	Description	Default Value
File	See the `F` entry in Table 9.30 in the *PDF Reference*.	

Multimedia viability dictionary

common_must_honor_best_effort

```
common_must_honor_best_effort =
  element Required { media_criteria_dictionary_wrap }?
  & element BestEffort { media_criteria_dictionary_wrap }?
```

media_criteria_dictionary_wrap

```
media_criteria_dictionary_wrap = media_criteria_dictionary?
```

Description

There are several entries in the multimedia object dictionaries whose values have an effect on viability. A multimedia viability dictionary is a dictionary whose entries must be honored to be considered viable, or need only be honored in a "best effort" sense. For more information, see section 9.1.2 in the *PDF Reference*.

Element or attribute	Description	Default Value
Required	The options specified by this entry must be honored; otherwise, the containing object is considered non-viable. See the `MH` entries in the tables appearing in section 9.1.2 in the *PDF Reference*.	
BestEffort	An attempt should be made to honor the options; however, if they cannot be honored, the containing object is still considered viable. `Required` and `BestEffort` are both elements, and the same entries are defined for both of them. In any dictionary where these entries are allowed, both entries may be present, or only one, or neither. See the `BE` entries in the tables appearing in section 9.1.2 in the *PDF Reference*.	

Name dictionary

name_dictionary

```
name_dictionary =
  element AnnotationAppearances { attribute src { string } }?
name_dictionary &= element JavaScripts { attribute src { string } }?
name_dictionary &= element Templates { template_page_name_tree* }?
name_dictionary &= element WebIds { attribute src { string } }?
name_dictionary &= element WebURLs { attribute src { string } }?
name_dictionary &= element EmbeddedFiles { attribute src { string } }?
name_dictionary &= element AlternatePresentations {
alternate_presentations_name_tree* }?
name_dictionary &= element Renditions { renditions_name_tree* }?
```

Description

In PDF, this would be the document's name dictionary. Its contents are in the `PDF` element. Some categories of objects can be referred to by name rather than by object reference. The correspondence between names and objects is established by the document's name dictionary, located by means of the `Names` entry in the document's catalog. Unlike PDF, the information that corresponds to name dictionary entries is not meant for efficient lookup by name; it is only meant to capture the information. For more information, see section 3.6.3 as well as the `Names` entry in Table 3.25 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
AnnotationAppearances	See the `AP` entry in Table 3.28 in the *PDF Reference*.	
JavaScripts	See the `JavaScript` entry in Table 3.28 in the *PDF Reference*.	
WebIds	See the `IDS` entry in Table 3.28 in the *PDF Reference*.	
WebURLs	See the `URLS` entry in Table 3.28 in the *PDF Reference*.	

Name objects

pdf_text_name

```
pdf_text_name = string
```

pdf_xml_name

```
pdf_xml_name = string
```

name

```
name = string
```

Description

A name object is an atomic symbol uniquely defined within its domain by a sequence of characters. Uniquely defined means that any two name objects made up of the same sequence of characters are identically the same object, provided those name objects are in the same domain. The name objects reflect specific domains: For more information, see section 3.2.4 in the *PDF Reference*.

- pdf_text_name is used for PDF names
- xml_text_name is used for XML names
- name is used for names that appear in user extensions

Named actions

named_action_dictionary

```
named_action_dictionary =
  attribute Name { pdf_byte_string }
  & attribute Name_enc { token }?
named_action_dictionary &= common_action_dictionary
```

Description

Named actions that Mars viewer applications are expected to support. For more information, see *Named Actions* in section 8.5.3 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
Name	See the `N` entry in Table 8.62 in the *PDF Reference*.	

Named destination cache file

cache

```
cache =
  element Cache { cache_header, cache_body }
```

cache_header

```
cache_header =
  element Query {
    element Files { element Pattern { attribute Value { text } }+ }?
    & attribute XPath { text }?
    & attribute Translate { text }?
  }?
  & attribute Identifier { text }?
  & element Timestamp {
    attribute Type { text }?
    & attribute Time { text }?
    & attribute DocumentID { pdf_base64_string }?
    & attribute InstanceID { pdf_base64_string }?
```

```
        }?
```

cache_body

```
cache_body =
  element Data { cache_data }
```

cache_data

```
cache_data = element Dest { dest_info }*
```

dest_info

```
dest_info =
  attribute Name { string }
  & attribute Page_ref { text }
```

Description

For named destinations, a cache file lists all of the named destination names and the page on which they are defined. This allows the location of each named destination in the document to be determined without requiring that all of the named destination files be read.

Example

```
<Cache Identifier="http://ns.adobe.com/pdf/2006/cache/destinations"
  xmlns="http://ns.adobe.com/pdf/2006">
  <Query>
    <Files>
      <Pattern Value="/page/*/dests.xml"/>
    </Files>
  </Query>
  <Data>
    <Dest Name="F2" Page_ref="/page/0/pg.pageinfo"/>
    <Dest Name="G2.997341" Page_ref="/page/1/pg.pageinfo"/>
  </Data>
</Cache>
```

Named destinations file

named_destinations_file

```
named_destinations_file =
  element Destinations { named_destinations_file_entry }
```

named_destinations_file_entry

```
named_destinations_file_entry = element Dest {
named_destinations_file_destination }*
```

named_destinations_file_destination

```
named_destinations_file_destination =
   attribute Name { pdf_byte_string }
   & attribute Name_enc { token }?
named_destinations_file_destination &= destination_array
```

Description

A file (`.dests`) in which the named destinations for a given page are stored. This file need be read only if a named destination it contains is to be used.

PDF documents store named destinations in a central name tree or array. In Mars documents, the named destinations are broken down, grouped, and stored with the page to which they refer. A separate cache file maps the names to the page file where the details of the destination are stored.

For more information, see section 8.2 in the *PDF Reference*.

Element or attribute	Description	Default Value
Name	The name of the named destination.	

Navigation node dictionary

navigation_node_dictionary

```
navigation_node_dictionary =
   element ForwardNavigationAction { action_dictionary }?
   & element BackwardNavigationAction { action_dictionary }?
   & element Next { navigation_node_dictionary }?
   & attribute AdvanceTime { number }?
```

Description

A navigation node dictionary specifies actions to execute when the user makes a navigation request; for example, by pressing an arrow key. For more information, see *Sub-page Navigation* in section 8.3.3 in the *PDF Reference*.

Element or attribute	Description	Default Value
ForwardNavigationAction	See the `NA` entry in Table 8.14 in the *PDF Reference*.	
BackwardNavigationAction	See the `PA` entry in Table 8.14 in the *PDF Reference*.	
AdvanceTime	See the `Dur` entry in Table 8.14 in the *PDF Reference*.	

Null object

null

```
null
```

Description

The null object has a type and value that are unequal to those of any other object. For more information, see section 3.2.8 in the *PDF Reference*.

Number format array

number_format_array

```
number_format_array = element Unit { number_format_dictionary }*
```

number_format_dictionary

```
number_format_dictionary =
   attribute UnitLabel { pdf_text_string }
   & attribute ConversionFactor { number }
   & attribute Fraction {  "Decimal" |  "Fraction" |  "Round" |  "Truncate" }?
   & attribute Precision { integer }?
   & attribute NoTruncate { boolean }?
   & attribute ThousandsDelimiter { pdf_text_string }?
   & attribute DecimalPoint { pdf_text_string }?
   & attribute LeftOfLabel { pdf_text_string }?
   & attribute AfterLabel { pdf_text_string }?
   & attribute LabelPosition {  "Suffix" |  "Prefix" }?
```

Description

A number format array is an array of number format dictionaries. Each number format dictionary represents a specific unit of measurement (such as miles or feet). It contains information about how each unit is expressed in text and factors for calculating the number of units. Number format arrays specify all the units that are to be used when expressing a specific measurement. For more information, see section 8.8 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
UnitLabel	See the `U` entry in Table 8.112 in the *PDF Reference*.	
ConversionFactor	See the `C` entry in Table 8.112 in the *PDF Reference*.	
Fraction	See the `F` entry in Table 8.112 in the *PDF Reference*.	
Precision	See the `D` entry in Table 8.112 in the *PDF Reference*.	

NoTruncate	See the `FD` entry in Table 8.112 in the *PDF Reference*.
ThousandsDelimiter	See the `RT` entry in Table 8.112 in the *PDF Reference*.
DecimalPoint	See the `RD` entry in Table 8.112 in the *PDF Reference*.
LeftOfLabel	See the `PS` entry in Table 8.112 in the *PDF Reference*.
AfterLabel	See the `SS` entry in Table 8.112 in the *PDF Reference*.
LabelPosition	See the `O` entry in Table 8.112 in the *PDF Reference*.

OPI CMYK color

opi_cmyk_color_array

```
opi_cmyk_color_array =
   attribute C { number }
   & attribute M { number }
   & attribute Y { number }
   & attribute K { number }
   & attribute colorName { pdf_byte_string }
   & attribute colorName_enc { token }?
```

Description

An array of four numbers and a string specifying the value and name of the color in which the image is to be rendered. For more information, see section 10.10.6 in the *PDF Reference*.

Element or attribute	Description	Default Value
C	The cyan component of the color.	
M	The magenta component of the color.	
Y	The yellow component of the color.	
K	The black component of the color.	
colorName	The name of the color.	

OPI crop rectangle

opi_crop_rect_array

```
opi_crop_rect_array =
  attribute Left { number }
  & attribute Top { number }
  & attribute Right { number }
  & attribute Bottom { number }
```

Description

A rectangle specifying the portion of the image to be used. For more information, see section 10.10.6 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
Left	The horizontal distance from the origin to the left side of the rectangle.	
Top	The vertical distance from the origin to the top side of the rectangle .	
Right	The horizontal distance from the origin to the right side of the rectangle.	
Bottom	The vertical distance from the origin to the bottom side of the rectangle.	

OPI image resolution

opi_resolution_array

```
opi_resolution_array =
  attribute horizRes { integer }
  & attribute vertRes { integer }
```

Description

An array of two numbers specifying the resolution of the image in samples per inch. For more information, see section 10.10.6 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
horizRes	The horizontal resolution.	
vertRes	The vertical resolution.	

OPI image size

opi_size_array

```
opi_size_array =
  attribute Width { integer }
  & attribute Height { integer }
```

Description

The dimensions of an image used in an OPI dictionary. For more information, see section 10.10.6 in the *PDF Reference*.

Element or attribute	Description	Default Value
Width	The width of the image.	
Height	The height of the image.	

OPI image type

opi_image_type_array

```
opi_image_type_array =
  attribute samples { integer }
  & attribute bits { integer }
```

Description

An array of two integers specifying the number of samples per pixel and bits per sample in the image. For more information, see section 10.10.6 in the *PDF Reference*.

Element or attribute	Description	Default Value
samples	The number of samples per pixel in the image.	
bits	The bits per sample in the image.	

OPI position

opi_position_array

```
opi_position_array =
  attribute llx { number }
  & attribute lly { number }
  & attribute ulx { number }
  & attribute uly { number }
  & attribute urx { number }
  & attribute ury { number }
  & attribute lrx { number }
  & attribute lry { number }
```

Description

An array of eight numbers specifying the location on the page of a cropped image. For more information, see section 10.10.6 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
llx	The x-coordinate of the lower left-hand corner.	
lly	The y-coordinate of the lower left-hand corner.	
ulx	The x-coordinate of the upper left-hand corner.	
uly	The y-coordinate of the upper left-hand corner.	
urx	The x-coordinate of the upper right-hand corner.	
ury	The y-coordinate of the upper right-hand corner.	
lrx	The x-coordinate of the lower right-hand corner.	
lry	The y-coordinate of the lower right-hand corner.	

Object reference

object_reference

```
object_reference = string
```

Description

A reference to a unique object identifier by which other objects can refer to that object. For more information, see section 3.2.9 in the *PDF Reference*.

Open action

action_or_destination

```
action_or_destination = action_dictionary
action_or_destination |= destination_goto_wrapper
```

destination_goto_wrapper

```
destination_goto_wrapper =
   element GoTo { destination_goto_wrapper_2 }
```

destination_goto_wrapper_2

```
destination_goto_wrapper_2 =
   element Dest { destination }
```

Description

A value specifying a destination to be displayed or an action to be performed when the document is opened. For more information, see the `OpenAction` entry in Table 3.25 in the *PDF Reference*.

Example

```
<AfterOpen>
  <GoTo>
    <Dest>
      <XYZ Left=" Top=" Zoom=" Page_ref="/backbone.xml#0" />
    </Dest>
  </GoTo>
</AfterOpen>
```

Open prepress interface (OPI) version dictionary

opi_version_dictionary

```
opi_version_dictionary =
  element OPI13 { opi_13_dictionary }?
  & element OPI20 { opi_20_dictionary }?
```

Description

A dictionary identifying the version of OPI to which an OPI proxy corresponds. For more information, see section 10.10.6 in the *PDF Reference*.

Element or attribute	Description	Default Value
OPI13	A version 1.3 OPI dictionary.	
OPI20	A version 2.0 OPI dictionary.	

Optional content configuration dictionary

oc_configuration_dictionary

```
oc_configuration_dictionary =
  attribute Name { pdf_text_string }?
  & attribute Creator { pdf_text_string }?
  & attribute BaseState { pdf_text_name }?
  & element ON { oc_group_dictionary_array }?
  & element OFF { oc_group_dictionary_array }?
  & element Intents { intent_name_or_array }?
  & element UsageApplications { usage_application_dictionary_array }?
  & element PresentationOrder { oc_order_array }?
  & attribute ListMode { pdf_text_name }?
  & element ExclusiveGroups { oc_radio_array }?
  & element Locked { oc_group_dictionary_reference_array }?
```

oc_group_dictionary_array

```
oc_group_dictionary_array =
  element Group { attribute ref { pdf_reference } }+
```

intent_name_or_array

```
intent_name_or_array =
  element Intent {
  attribute _enc { token }?
  & name }
intent_name_or_array |= element Intent {
  attribute _enc { token }?
  & name }*
```

oc_group_dictionary_reference_array

```
oc_group_dictionary_reference_array =
  element Group { attribute ref { pdf_reference } }+
```

Description

An element that represents different presentations of a document's optional content groups. For more information, see *Optional Content Configuration Dictionaries* in section 4.10.3 in the *PDF Reference*.

Element or attribute	Description	Default Value
Intents	See the `Intent` entry in Table 4.51 in the *PDF Reference*.	
UsageApplications	See the `AS` entry in Table 4.51 in the *PDF Reference*.	
PresentationOrder	See the `Order` entry in Table 4.51 in the *PDF Reference*.	
ExclusiveGroups	See the `RBGroups` entry in Table 4.51 in the *PDF Reference*.	

Optional content group

oc_group_or_membership_dictionary

```
oc_group_or_membership_dictionary =
  element Group { attribute ref { pdf_reference } }
oc_group_or_membership_dictionary |=
  element Membership { oc_membership_dictionary }
```

Description

An optional content group is an element representing a collection of graphics that can be made visible or invisible dynamically by users of viewer applications. Optional content groups, configurations, and so on are stored in the document backbone. For more information, see section 4.10.1 in the *PDF Reference*.

Optional content group

oc_group_dictionary

```
oc_group_dictionary =
   attribute Name { pdf_text_string }
   & element Intents { oc_intent_name_or_array }?
   & element Usage { oc_usage_dictionary }?
```

oc_intent_name_or_array

```
oc_intent_name_or_array =
   element Intent {
   attribute _enc { token }?
   & name }
oc_intent_name_or_array |= element Intent {
   attribute _enc { token }?
   & name }*
```

Description

Optional content in PDF consists of a number of structures that describe sets of content items and labels to use in a user interface to enable them to be selectively displayed. Individual content items are grouped and tagged using marked content containers that wrap the actual page content items.

In Mars, the structures are represented in the document backbone and the content information is represented on the pages, wrapping relevant SVG page content. The backbone information is the `<OCProperties>` element.

For more information, see section 4.10.1 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
Intents	See the `Intent` entry in Table 4.48 in the *PDF Reference*.	

Optional content membership dictionary

oc_membership_dictionary

```
oc_membership_dictionary =
   element Groups { oc_group_dictionary_or_array }?
   & attribute VisibilityPolicy { name }?
   & attribute VisibilityPolicy_enc { token }?
   & element Visibility { default_array }?
```

oc_group_dictionary_or_array

```
oc_group_dictionary_or_array =
   element Group { attribute ref { pdf_reference } }+
oc_group_dictionary_or_array |=
   element Group { attribute ref { pdf_reference } }
```

Description

Used to express more complex visibility policies for content that is not declared to belong directly to an optional content group. For more information, see section 4.10.1 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
Groups	See the OCGs entry in Table 4.49 in the *PDF Reference*.	
VisibilityPolicy	See the P entry in Table 4.49 in the *PDF Reference*.	
Visibility	See the VE entry in Table 4.49 in the *PDF Reference*.	AnyOn

Optional content properties dictionary

oc_properties_dictionary

```
oc_properties_dictionary =
  element Groups { oc_group_dictionary_definitions_array }
  & element Default { oc_configuration_dictionary }
  & element Configs { oc_configuration_dictionary_array }?
```

oc_configuration_dictionary_array

```
oc_configuration_dictionary_array = element Config {
oc_configuration_dictionary }*
```

oc_group_dictionary_definitions_array

```
oc_group_dictionary_definitions_array = element Group { oc_group_dictionary
}*
```

oc_configuration_dictionary

```
oc_configuration_dictionary =
  attribute Name { pdf_text_string }?
  & attribute Creator { pdf_text_string }?
  & attribute BaseState { pdf_text_name }?
  & element ON { oc_group_dictionary_array }?
  & element OFF { oc_group_dictionary_array }?
  & element Intents { intent_name_or_array }?
  & element UsageApplications { usage_application_dictionary_array }?
  & element PresentationOrder { oc_order_array }?
  & attribute ListMode { pdf_text_name }?
  & element ExclusiveGroups { oc_radio_array }?
  & element Locked { oc_group_dictionary_reference_array }?
```

Description

Contains a list of all the optional content groups in the document, as well as information about the default and alternate configurations for optional content. For more information, see the `OCProperties` entry in Table 3.25, as well as *Optional Content Properties Dictionary* in section 4.10.3 in the *PDF Reference*.

Element or attribute	Description	Default Value
Groups	See the `OCGs` entry in Table 4.50 in the *PDF Reference*.	
Default	See the `D` entry in Table 4.50 in the *PDF Reference*.	

Optional content usage dictionary

oc_usage_dictionary

```
oc_usage_dictionary =
  element CreatorInfo { oc_creatorinfo_dictionary }?
  & element Language { oc_language_dictionary }?
  & element Export { oc_export_dictionary }?
  & element Zoom { oc_zoom_dictionary }?
  & element Print { oc_print_dictionary }?
  & element View { oc_view_dictionary }?
  & element User { oc_user_dictionary }?
  & element PageElement { oc_pageelement_dictionary }?
```

oc_creatorinfo_dictionary

```
oc_creatorinfo_dictionary =
  attribute Creator { pdf_text_string }
  & attribute Subtype { name }
  & attribute Subtype_enc { token }?
```

oc_language_dictionary

```
oc_language_dictionary =
  attribute Lang { pdf_ascii_string }
  & attribute Preferred { pdf_text_name }?
```

oc_export_dictionary

```
oc_export_dictionary =
  attribute ExportState { pdf_text_name }
```

oc_zoom_dictionary

```
oc_zoom_dictionary =
  attribute Min { number }
  & attribute Max { number }
```

oc_print_dictionary

```
oc_print_dictionary =
  attribute Subtype { pdf_text_name }?
  & attribute PrintState { pdf_text_name }?
```

oc_view_dictionary

```
oc_view_dictionary =
  attribute ViewState { pdf_text_name }
```

oc_user_dictionary

```
oc_user_dictionary =
  attribute Type { pdf_text_name }
  & attribute Name { pdf_text_string }
```

oc_pageelement_dictionary

```
oc_pageelement_dictionary =
  attribute Subtype { pdf_text_name }
```

Description

A usage dictionary describing the nature of the content controlled by the optional content group. For more information, see the `Usage` entry in Table 4.48 in the *PDF Reference*.

Options array

choice_field_opt_array

```
choice_field_opt_array = element Choice { choice_field_opt_array_elt }*
```

choice_field_opt_array_elt

```
choice_field_opt_array_elt =
  attribute Value { pdf_text_string }
choice_field_opt_array_elt |=
  attribute Value { pdf_text_string }
  & attribute Display { pdf_text_string }
```

Description

An array of options to be presented to the user. For more information, see the `Opt` entry in Table 8.80 in the *PDF Reference*.

Element or attribute	Description	Default Value
Value	The option's export value.	
Display	The text to be displayed as the name of the option.	

Origin array

origin_array

```
origin_array =
  attribute X { number }
  & attribute Y { number }
```

Description

An array of two numbers specifying the origin of the measurement coordinate system in default user space coordinates. For more information, see the O entry in Table 8.111 in the *PDF Reference*.

Element or attribute	Description	Default Value
X	The x-coordinate specifying the origin of the measurement coordinate system in default user space coordinates.	
Y	The y-coordinate specifying the origin of the measurement coordinate system in default user space coordinates.	

Outline file

Description

See Bookmarks file.

Output intents

output_intent_dictionary

```
output_intent_dictionary &=
  attribute Name { text }
output_intent_dictionary =
  attribute Subtype { name }
  & attribute Subtype_enc { token }?
  & attribute OutputCondition { pdf_text_string }?
  & attribute OutputConditionIdentifier { pdf_ascii_string }
  & attribute RegistryName { pdf_ascii_string }?
  & attribute Info { pdf_text_string }?
  & element DestOutputProfile { ICC_profile_stream_dictionary }?
```

Description

Output intents provide a means for matching the color characteristics of a document with those of a target output device or production environment in which the document will be printed. For more information, see section 10.10.4 in the *PDF Reference*.

Element or attribute	Description	Default Value
Subtype	See the `S` entry in Table 10.51 in the *PDF Reference*.	
OutputCondition	See the `OutputCondition` entry in Table 10.51 in the *PDF Reference*.	
OutputConditionIdentifier	See the `OutputConditionIdentifier` entry in Table 10.51 in the *PDF Reference*.	
RegistryName	See the `RegistryName` entry in Table 10.51 in the *PDF Reference*.	
Info	See the `Info` entry in Table 10.51 in the *PDF Reference*.	
DestOutputProfile	See the `DestOutputProfile` entry in Table 10.51 in the *PDF Reference*.	

Page content annotations

Description

See Annotations, content (file).

PDF language version

array_of_PDF_versions

```
array_of_PDF_versions =
  attribute MinVersion { name }
  & attribute MinVersion_enc { token }?
  & attribute MaxVersion { name }?
  & attribute MaxVersion_enc { token }?
```

Description

An element containing one or two attributes that specify a minimum and optionally a maximum PDF language version. For more information, see the `P` entry in Table 9.3 in the *PDF Reference*.

Element or attribute	Description	Default Value
MinVersion	A string specifying a minimum PDF language version.	
MaxVersion	A string specifying a maximum PDF language version.	

Page dictionary references

array_of_page_dictionary_reference

```
array_of_page_dictionary_reference =
  element Page { attribute ref { pdf_reference } }+
```

Description

A series of elements referencing pages in the document. For more information, see the `Pages` entry in Table 10.50 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
ref	A page identifier that corresponds to a page element in the backbone file	

Page information file

page_file

```
page_file =
   element Page { page_dictionary }
```

page_dictionary

```
page_dictionary =
   attribute LastModified { date }?
   & element CropBox { rectangle }?
   & element BleedBox { rectangle }?
   & element TrimBox { rectangle }?
   & element ArtBox { rectangle }?
   & element BoxColorInfo { box_color_information_dictionary }?
   & element Contents { attribute src { string } }
   & attribute Rotate {  "0" |  "90" |  "180" |  "270" }?
   & element Attributes { group_attributes_dictionary }?
   & element Thumbnail { image_dictionary }?
   & attribute Duration { number }?
   & element Transition { transition_dictionary }?
   & element Annotations { attribute src { string } }?
page_dictionary &= page_additional_actions_dictionary?
page_dictionary &=
   element ApplicationDatasets { page_piece_dictionary }?
   & attribute WebCaptureId { pdf_base64_string }?
   & attribute PreferredZoom { number }?
   & element SeparationInfo { separation_dictionary }?
   & attribute Tabs { name }?
   & attribute Tabs_enc { token }?
   & attribute TemplateInstantiated { name }?
   & attribute TemplateInstantiated_enc { token }?
   & element PresSteps { navigation_node_dictionary }?
   & element ViewPorts { viewport_dictionary_array }?
   & element Destinations { attribute src { string } }?
   & element Structure { attribute src { string } }?
```

Description

The page information file contains basic information about the page. It must be read in order to know anything about the page. The page information file includes the following information:

- A reference to the page content file.
- A reference to the page named destination file, if any.
- Page size and orientation information.

In the document backbone, the pages are represented as a series of XML <Page> elements. Each of these points to page information which is stored in separate files. The graphic page content itself is represented using SVG. Some structure tree and marked content information is included on the page interspersed with the SVG itself. Additional structure information, named destination information, resources, and annotations are stored in separate files associated with the page to which they apply.

The basic list of pages is stored within the document backbone. The information about the page itself is broken into up to eight or more files: the page information file, the page content file, the page structure file, the named destinations file, the content annotations file, the markup annotations file, page-level metadata (xmp) file, the page resources file, and any images, fonts, font descriptors, functions, shaders, patterns, color profiles, and others. In addition, the page content itself can be broken into multiple files using the PDF *form xobject* mechanism (PDF) and the SVG symbol mechanism (Mars).

Element or attribute	**Description**	**Default Value**
LastModified	(Required if `ApplicationDatasets` is present; optional otherwise) The date and time when the page's contents were most recently modified. If `ApplicationDatasets` is present, the modification date is used to ascertain which of the application data sets that it contains correspond to the current content of the page.	
Contents	A reference to the page content file.	
Annotations	A reference to the content annotations file.	
Destinations	A reference to the page named destination file, if any.	
Structure	A reference to the page structure file.	
Attributes	See the `Group` entry in Table 3.27 in the *PDF Reference*.	
Thumbnail	See the `Thumb` entry in Table 3.27 in the *PDF Reference*.	
Actions	See the `AA` entry in Table 3.27 in the *PDF Reference*.	
ApplicationDatasets	See the `PieceInfo` entry in Table 3.27 in the *PDF Reference*.	
WebCaptureId	See the `ID` entry in Table 3.27 in the *PDF Reference*.	
PreferredZoom	See the `PZ` entry in Table 3.27 in the *PDF Reference*.	
ViewPorts	See the `VP` entry in Table 3.27 in the *PDF Reference*.	

Page label dictionary

page_label_dictionary_wrapper

```
page_label_dictionary_wrapper =
  attribute Start { text }
page_label_dictionary_wrapper |=
  element PageLabel { page_label_dictionary }
```

page_label_dictionary

```
page_label_dictionary =
  attribute Style {  "Numeric" |  "Roman_Uppercase" |  "Roman_Lowercase" |
  "Letter_Uppercase" |  "Letter_Lowercase" }?
  & attribute Prefix { pdf_text_string }?
  & attribute FirstInRange { integer }?
```

Description

A document's labeling ranges are defined by the `PageLabels` entry in the `PageLabel` element in the document backbone. The value of this entry defines a range of pages and the labeling characteristics for the pages in that range. The element must include a value for page index `0`. For more information, see the `PageLabels` entry in Table 3.25, and section 8.3.1 in the *PDF Reference*.

Element or attribute	Description	Default Value
Style	See the `S` entry in Table 8.10 in the *PDF Reference*.	
Prefix	See the `P` entry in Table 8.10 in the *PDF Reference*.	
FirstInRange	See the `St` entry in Table 8.10 in the *PDF Reference*.	

Page layout

layout_type

```
layout_type = name
```

Description

A name object specifying the page layout to be used when the document is opened. For more information, see the `PageLayout` entry in Table 3.25 in the *PDF Reference*.

Page mode

pagemode_type

```
pagemode_type = name
```

Description

A name object specifying how the document should be displayed when opened. For more information, see the `PageMode` entry in Table 3.25 in the *PDF Reference*.

Page structure file

structure_file

```
structure_file =
  element Structure { structure_file_contents }
```

structure_file_contents

```
structure_file_contents =
  element Elem { structure_file_element }*
  & element MCR { marked_content_reference_dictionary }*
```

Description

The structure tree in PDF provides a logical description of the document contents. Its view is more along the lines of an HTML document with chapters, sections, paragraphs, tables, figures, articles, and so on. The structure tree contains interior nodes which typically describe chapters, sections, tables, and so on, and leaf nodes which typically describe content items such as paragraphs, table cells, headings, and so on.

In Mars, the structure tree is not represented as a single tree but is broken into separate pieces of information that are stored in different locations in the document package.

The page structure file in Mars lists structure information for interior nodes of the structure tree that are parents of leaf nodes that appear on the page. The SVG page content itself contains the structure information for the leaf nodes of the structure tree. Each page includes structure information for the structure tree elements that span from the structure tree root to the leaves that appear on the page. Thus, each page contains complete information for the portion of the structure tree needed to describe content on it (via a combination of the page content file and the page structure file).

All structure information is distributed onto individual pages, which contain the content items ultimately referred to by the structure tree.

For more information, see section 10.6.1 in the *PDF Reference*

Element or attribute	Description	Default Value
Element	Describes one node in the structure tree.	

Page-piece dictionary file

application_datasets_file

```
application_datasets_file =
  element ApplicationDatasets { page_piece_dictionary }
```

page_piece_dictionary

```
page_piece_dictionary = element ApplicationData { attribute Owner { text }
application_data_dictionary }+
```

application_data_dictionary

```
application_data_dictionary =
  attribute LastModified { date }
  & element Private { default_dict_element }
```

Description

A page-piece dictionary can be used to hold private application-specific extension data that can be associated with the document, individual pages, and fragments of pages (form XObjects).

The XML markup representing the application-specific information associated with the document is stored in a separate file, referenced from the document backbone. The root of this file is the `ApplicationDatasets` element.

The `ApplicationDatasets` element and its contained application-specific information can also appear in the page information file as part of the <Page> element, or in the resource file as part of the <Resources>, <XObjects>, and <Form> elements, and in various other files and locations where a content fragment (form_dictionary) appears.

The externally-defined elements in the `ApplicationDatasets` element are flagged with the underlying type so that they can be consistently processed. Scalar and structured values can appear in private data.

For more information, see section 10.4 in the *PDF Reference*.

Example

```
<ApplicationDatasets>
  <ApplicationData Owner="Jones Formatter 1.2" LastModified="20430626221926">
    <Private type="name" Value="NoRecallSpellingHollars"/>
  </ApplicationData>
  <ApplicationData Owner="Flex 1.1" LastModified="20030226221926">
    <Private type="dict">
      <Server type="string" Value="Acrolab1"/>
      <Age type="int" Value="23"/>
    </Private>
  </ApplicationData>
<ApplicationDatasets>
```

Element or attribute	Description	Default Value
ApplicationData	See Table 10.5 in the *PDF Reference*.	

Page tree

page_tree_dictionary

```
page_tree_dictionary = element Page {
   attribute src { string }
   & rectangle
   & attribute UserUnit { number }? }+
```

page_dictionary_wrapper

```
page_dictionary_wrapper =
   element Page { page_dictionary_wrap1 }
```

page_dictionary_wrap1

```
page_dictionary_wrap1 = page_tree_dictionary
```

Description

The page tree node that is the root of the document's page tree. For more information, see section 3.6.2 as well as the `Pages` entry in Table 3.25 in the *PDF Reference*.

Example

```
<Pages>
   <Page src="/page/0/info.xml" x1="0" y1="0" x2="612" y2="792" ID="0" />
   <Page src="/page/1/info.xml" x1="0" y1="0" x2="612" y2="792" />
</Pages>
```

Parameter dictionaries

parm_dictionary_or_array

```
parm_dictionary_or_array = element ParmDictionary { parm_dictionary }*
```

parm_dictionary

```
parm_dictionary = element * {  default_dict_element }*
```

Description

A parameter dictionary or an array of such dictionaries.

Pattern color space

pattern_color_space

```
pattern_color_space = element UnderlyingSpace { color_space }?
```

Description

A Pattern color space enables a content stream to paint an area with a pattern rather than a single color. For more information, see *Pattern Color Spaces* in section 4.5.5 in the *PDF Reference*.

Element or attribute	Description	Default Value
UnderlyingSpace	The underlying color space.	

Polygon and polyline annotation

poly_annotation_dictionary

```
poly_annotation_dictionary = common_annotation_dictionary
poly_annotation_dictionary &= common_markup_annotation_dictionary
poly_annotation_dictionary &=
  attribute Path { vertices_array }
poly_annotation_dictionary &= line_ending_style_array?
poly_annotation_dictionary &=
  attribute InteriorColor { rgb_color_array }?
  & element BorderEffect { border_effect_dictionary }?
  & element Measure { measure_dictionary }?
```

vertices_array

```
vertices_array =  ( list { number } )
```

line_ending_style_array

```
line_ending_style_array =
  attribute Head { name }
  & attribute Head_enc { token }?
  & attribute Tail { name }
  & attribute Tail_enc { token }?
```

Description

Polygon annotations display closed polygons on the page. Such polygons may have any number of vertices connected by straight lines. Polyline annotations are similar to polygons, except that the first and last vertex are not implicitly connected. For more information, see *Polygon and Polyline Annotations* in section 8.4.5 in the *PDF Reference*.

Element or attribute	Description	Default Value
Path	See the `Vertices` entry in Table 8.29 in the *PDF Reference*.	
InteriorColor	See the `IC` entry in Table 8.29 in the *PDF Reference*.	
BorderEffect	See the `BE` entry in Table 8.29 in the *PDF Reference*.	

Pop-up annotation

popup_annotation_dictionary

```
popup_annotation_dictionary = common_annotation_dictionary
popup_annotation_dictionary &= attribute Open { boolean }?
```

Description

Displays text in a pop-up window for entry and editing. For more information, see *Pop-up Annotations* in section 8.4.5 in the *PDF Reference*.

Presentation order

oc_order_array

```
oc_order_array = oc_order_array_element*
```

oc_order_array_element

```
oc_order_array_element =
   element Group { attribute ref { pdf_reference } }
oc_order_array_element |=
   element Groups { oc_order_array_element_1 }
```

oc_order_array_element_1

```
oc_order_array_element_1 = string_or_oc_group_dictionary_or_array*
```

string_or_oc_group_dictionary_or_array

```
string_or_oc_group_dictionary_or_array = oc_order_array_element
string_or_oc_group_dictionary_or_array |= oc_order_array_element
string_or_oc_group_dictionary_or_array |=
   element Label { pdf_text_string }
```

Description

A series of elements specifying the recommended order for presentation of optional content groups in a user interface. For more information, see the `Order` entry in Table 4.51 in the *PDF Reference*.

Printers mark annotation dictionary

printers_mark_annotation_dictionary

```
printers_mark_annotation_dictionary = common_annotation_dictionary
printers_mark_annotation_dictionary &= element MarkName {
  attribute _enc { token }?
  & name }?
```

Description

Printer's mark annotations provide a mechanism for incorporating a printer's marks into the PDF representation of a page, while keeping them separate from the actual page content. For more information, see section 10.10.2 in the *PDF Reference*.

Element or attribute	Description	Default Value
MarkName	See the `MN` entry in Table 10.48 in the *PDF Reference*.	

Projection dictionary

projection_dictionary

```
projection_dictionary =
  attribute Subtype {  "Orthographic" |  "Perspective" }
  & attribute ClipStyle { pdf_text_name }?
  & attribute FarClipDistance { number }?
  & attribute NearClipDistance { number }?
  & attribute FieldOfView { number }?
projection_dictionary &= projection_scaling?
projection_dictionary &=
  attribute ScaleFactor { number }?
  & attribute ScaleCoords { pdf_text_name }?
```

Description

A projection dictionary defines the mapping of 3D camera coordinates onto the target coordinate system of the annotation. For more information, see *Projection Dictionaries* in section 9.5.3 in the *PDF Reference*.

Element or attribute	Description	Default Value
ClipStyle	See the `CS` entry in Table 9.38 in the *PDF Reference*.	ANF

FarClipDistance	See the `F` entry in Table 9.38 in the *PDF Reference*.	
NearClipDistance	See the `N` entry in Table 9.38 in the *PDF Reference*.	
FieldOfView	See the `FOV` entry in Table 9.38 in the *PDF Reference*.	
ScaleFactor	See the `OS` entry in Table 9.38 in the *PDF Reference*.	
ScaleCoords	See the `OB` entry in Table 9.38 in the *PDF Reference*.	Absolute

Projection scaling

projection_scaling

```
projection_scaling =
   attribute ProjectionDiameter { number }
projection_scaling |=
   attribute ProjectionViewBox { name }
   & attribute ProjectionViewBox_enc { token }?
```

Description

An object that specifies the scaling used when projecting the 3D artwork onto the annotation's target coordinate system. For more information, see the `PS` entry in Table 9.38 in the *PDF Reference*.

Element or attribute	Description	Default Value
ProjectionDiameter	A positive number that explicitly specifies the diameter as a distance in the annotation's target coordinate system.	
ProjectionViewBox	A name specifying that the diameter should be set to the width (`W`), height (`H`), minimum of width and height (`Min`), or maximum of width and height (`Max`) of the annotation's 3D view box.	W

RGB color array

rgb_color_array

```
rgb_color_array = string
```

Description

An array of three numbers in the range `0.0` to `1.0`, representing the red, green, and blue values, respectively, of an RGB color space.

Radio button optional content groups

oc_radio_array

```
oc_radio_array = element ExclusiveGroup { oc_radio_array_element }*
```

oc_radio_array_element

```
oc_radio_array_element = element Group { attribute ref { pdf_reference } }+
```

Description

A element consisting of one or more subelements, each of which represents a collection of optional content groups whose states are intended to follow a radio button paradigm. For more information, see the `RBGroups` entry in Table 4.51 in the *PDF Reference*.

Real object

number

```
number = float
number |= integer
```

Description

A representation of a real number. This type is used only for extensions to the Mars grammar.

Rectangle

rectangle_blank_sep

```
rectangle_blank_sep =  ( list { number number number number } )
```

rectangle

```
rectangle =
   attribute x1 { number }
   & attribute y1 { number }
   & attribute x2 { number }
   & attribute y2 { number }
```

Description

An object that can be used to describe a location on a page or a bounding box for a variety of objects. For more information, see section 3.8.4 in the *PDF Reference*.

Rectangle

rectangle

```
rectangle =
  attribute x1 { number }
  & attribute y1 { number }
  & attribute x2 { number }
  & attribute y2 { number }
```

Description

Used to describe a location on a page and a bounding box for a variety of objects, such as fonts. A rectangle is written as four numbers giving the coordinates of a pair of diagonally opposite corners. For more information, see section 3.8.4 in the *PDF Reference*.

Redaction annotation

redaction_annotation_dictionary

```
redaction_annotation_dictionary = common_annotation_dictionary
redaction_annotation_dictionary &=
  attribute Coords { quadpoints_array }?
  & attribute InteriorColor { rgb_color_array }?
  & element OverlayAppearance { form_dictionary }?
  & attribute OverlayText { pdf_text_string }?
  & attribute Repeat { boolean }?
  & element DefaultAppearance { attribute Font { text }
    attribute TextSize { number }?
    attribute FillColorSpace {  "Gray" |  "RGB" |  "CMYK" }?
    attribute FillColor { number |  ( list { number number numbernumber? } )
    }?
    attribute StrokeColorSpace {  "Gray" |  "RGB" |  "CMYK" }?
    attribute StrokeColor { number |  ( list { number number numbernumber? }
    ) }?
    attribute CharacterSpacing { number }?
    attribute TextLeading { number }?
    attribute TextMatrix {  ( list { number number number number number
    number } ) }?
    attribute RenderMode { number }?
    attribute TextRise { number }?
    attribute WordSpacing { number }?
    attribute HorizontalScaling { number }?
  }
  & attribute Justification {  "Left" |  "Centered" |  "Right" }?
```

Description

A redaction annotation identifies content that is intended to be removed from the document. For more information, see the *PDF Redaction: Addendum for the PDF Reference, sixth edition, version 1.7*.

Element or attribute	**Description**	**Default Value**
Coords	See the `QuadPoints` entry in Table 1 in the *PDF Redaction: Addendum for the PDF Reference, sixth edition, version 1.7*.	
InteriorColor	See the `IC` entry in Table 1 in the *PDF Redaction: Addendum for the PDF Reference, sixth edition, version 1.7*.	
OverlayAppearance	See the `RO` entry in Table 1 in the *PDF Redaction: Addendum for the PDF Reference, sixth edition, version 1.7*.	
OverlayText	See the `OverlayText` entry in Table 1 in the *PDF Redaction: Addendum for the PDF Reference, sixth edition, version 1.7*.	
Repeat	See the `Repeat` entry in Table 1 in the *PDF Redaction: Addendum for the PDF Reference, sixth edition, version 1.7*.	
DefaultAppearance	See the `DA` entry in Table 1 in the *PDF Redaction: Addendum for the PDF Reference, sixth edition, version 1.7*.	
Justification	Specifying the form of quadding (justification) to be used in laying out the overlay text. The following values are defined: • `Left` - The overlay text is left justified. • `Centered` - The overlay text is centered. • `Right` - The overlay text is right justified. This entry is ignored if the `RO` entry is present.	`Left`

Reference dictionary

reference_dictionary

```
reference_dictionary =
  element File { file_spec_dictionary }
  & element Page { pdf_text_string }
  & element FileId { array_of_2_strings }?
```

Description

Represents a single, complete page of imported content. For more information, see section 4.9.3 in the *PDF Reference*.

Element or attribute	Description	Default Value
File	See the `F` entry in Table 4.47 in the *PDF Reference*.	
FileId	See the `ID` entry in Table 4.47 in the *PDF Reference*.	

Related files array

file_spec_related_files

```
file_spec_related_files =
  element FileData { embedded_file_stream_dictionary }
file_spec_related_files |=
  element Files { related_files_array }
```

related_files_array

```
related_files_array = related_files_array_element*
```

related_files_array_element

```
related_files_array_element =
  element FileData { embedded_file_stream_dictionary }
related_files_array_element |=
  element Name {
  attribute _enc { token }?
  & pdf_byte_string }
```

Description

A reference to a group of related files, such as the set of five files that make up a DCS 1.0 color-separated image. For more information, see *Related Files Arrays* in section 3.10.3 in the *PDF Reference*.

Remote go-to action

gotor_action_dictionary

```
gotor_action_dictionary =
  element File { file_spec_dictionary }
  & element Dest { destination }
  & attribute NewWindow { boolean }?
gotor_action_dictionary &= common_action_dictionary
```

Description

A remote go-to action is similar to an ordinary go-to action, but jumps to a destination in another Mars document instead of the current document. For more information, see *Remote Go-To Actions* in and in *Representing Actions* in the guide *Developing Applications Using Mars* in the *PDF Reference*.

Element or attribute	Description	Default Value
File	See the `F` entry in Table 8.50 in the *PDF Reference*.	
Dest	See the `D` entry in Table 8.50 in the *PDF Reference*.	

Render mode dictionary

render_mode_dictionary

```
render_mode_dictionary =
   attribute Subtype { pdf_text_name }
   & element AuxColor { auxiliary_color_array }?
   & element FaceColor { face_color }?
   & attribute Opacity { number }?
   & attribute CreaseAngle { number }?
```

face_color

```
face_color =
   attribute Color { pdf_text_name }
face_color |= auxiliary_color_array
```

Description

An element that enables document authors to customize the rendered appearance of 3D artwork to suit the needs of the intended consumer, without reauthoring the artwork. For more information, see *3D Render Mode Dictionaries* in section 9.5.3 in the *PDF Reference*.

Element or attribute	Description	Default Value
AuxColor	See the `AC` entry in Table 9.42 in the *PDF Reference*.	
FaceColor	See the `FC` entry in Table 9.42 in the *PDF Reference*.	BG
Opacity	See the `O` entry in Table 9.42 in the *PDF Reference*.	0.5
CreaseAngle	See the `CV` entry in Table 9.42 in the *PDF Reference*.	45

Rendition action

rendition_action_dictionary

```
rendition_action_dictionary =
  element MediaRendition { rendition_dictionary }?
  & element Screen { attribute ref { pdf_reference } }?
  & attribute Operation {  "PlayAndAssociate" |  "Stop" |  "Pause" |
  "Resume" |  "PlayAssociated" }?
  & element Script { javascript_action_js }?
rendition_action_dictionary &= common_action_dictionary
```

Description

A rendition action controls the playing of multimedia content. For more information, see *Rendition Actions* in section 8.5.3 in the *PDF Reference*.

Element or attribute	Description	Default Value
MediaRendition	See the `R` entry in Table 8.64 in the *PDF Reference*.	
Screen	See the `AN` entry in Table 8.64 in the *PDF Reference*.	
Operation	See the `OP` entry in Table 8.64 in the *PDF Reference*.	
Script	See the `JS` entry in Table 8.64 in the *PDF Reference*.	

Rendition

rendition_dictionary

```
rendition_dictionary =
  element Media { media_rendition_dictionary }
rendition_dictionary |=
  element Selector { selector_rendition_dictionary }
```

Description

There are two types of rendition objects:

- A media rendition is a basic media object that specifies what to play, how to play it, and where to play it.
- A selector rendition contains an ordered list of renditions. This list may include other selector renditions, resulting in a tree whose leaves are media renditions.

For more information, see section 9.1.2 in the *PDF Reference*.

Element or attribute	Description	Default Value
Media	A media rendition.	
Selector	A selector rendition.	

Renditions list

renditions_name_tree

```
renditions_name_tree =
   element MediaRendition {
   attribute Key { text }
   & rendition_dictionary }
```

Description

An XML element that maps name strings (which must have Unicode encoding) to rendition objects. For more information, see section 9.1.2, as well as the `Renditions` entry in Table 3.28 in the *PDF Reference*.

Reset-form action

resetform_action_dictionary

```
resetform_action_dictionary =
   element Fields { fields_array }?
   & attribute Flags {  ( list {  "Exclude"? } ) }?
resetform_action_dictionary &= common_action_dictionary
```

Description

A reset-form action resets selected interactive form fields to their default values. For more information, see *Reset-Form Actions* in section 8.6.4 in the *PDF Reference*.

Rich text string

rt_string_or_stream

```
rt_string_or_stream = pdf_text_string
rt_string_or_stream |=
   attribute src { string }
   & common_stream_dictionary
```

Description

A rich text string contains the text contents of a variable text form field, as well as markup annotations, and can include formatting (style) information. For more information, see *Rich Text Strings* in section 8.6.2 in the *PDF Reference*.

Rubber stamp annotation

stamp_annotation_dictionary

```
stamp_annotation_dictionary = common_annotation_dictionary
stamp_annotation_dictionary &= common_markup_annotation_dictionary
stamp_annotation_dictionary &=
   attribute IconName { name }?
   & attribute IconName_enc { token }?
```

Description

Displays text or graphics intended to look as if they were stamped on the page with a rubber stamp. For more information, see *Rubber Stamp Annotations* in section 8.4.5 in the *PDF Reference*.

Element or attribute	Description	Default Value
IconName	See the `Name` entry in Table 8.32 in the *PDF Reference*.	

Screen annotation

screen_annotation_dictionary

```
screen_annotation_dictionary = common_annotation_dictionary
screen_annotation_dictionary &=
   attribute Title { pdf_text_string }?
   & element AppearanceCharacteristics {
   appearance_characteristics_dictionary }?
   & element OnActivation { action_dictionary }?
screen_annotation_dictionary &= annotation_additional_actions_dictionary?
```

Description

Specifies a region of a page upon which media clips may be played. For more information, see *Screen Annotations* in section 8.4.5 in the *PDF Reference*.

Element or attribute	Description	Default Value
Title	See the `T` entry in Table 8.38 in the *PDF Reference*.	
AppearanceCharacteristics	See the `MK` entry in Table 8.38 in the *PDF Reference*.	
OnActivation	See the `A` entry in Table 8.38 in the *PDF Reference*.	

Selector renditions

selector_rendition_dictionary

```
selector_rendition_dictionary = common_rendition_dictionary
selector_rendition_dictionary &=
  element Renditions { array_of_rendition_dictionary }
```

array_of_rendition_dictionary

```
array_of_rendition_dictionary = element MediaRendition {
rendition_dictionary }*
```

Description

A selector rendition specifies an array of rendition objects. For more information, see *Selector Renditions* in section 9.1.2 in the *PDF Reference*.

Element or attribute	Description	Default Value
Renditions	See the `R` entry in Table 9.7 in the *PDF Reference*.	

Separation color space

separation_color_space

```
separation_color_space =
  attribute Colorant { name }
  & attribute Colorant_enc { token }?
  & element AlternateSpace { color_space }
  & element TintXform { function_dictionary }
```

Description

A Separation color space provides a means for specifying the use of additional colorants or for isolating the control of individual color components of a device color space for a subtractive device. When such a space is the current color space, the current color is a single-component value, called a *tint*, that controls the application of the given colorant or color components only. For more information, see `Separation Color Spaces` in section 4.5.5 in the *PDF Reference*.

Element or attribute	Description	Default Value
Colorant	A name object specifying the name of the colorant that this separation color space is intended to represent. For more information, see `Separation Color Spaces` in section 4.5.5 in the *PDF Reference*.	

Separation dictionary

separation_dictionary

```
separation_dictionary =
  element Pages { array_of_page_dictionary_reference }
  & element DeviceColorant { colorant_name_or_string }
  & element ColorSpace { color_space_array }?
```

Description

In a preseparated document, the separations for a page are described as separate page objects, each painting only a single colorant (usually specified in the DeviceGray color space). When this is done, a separation dictionary is used to represent additional information needed to identify the actual colorant associated with each separation and to group together the page objects representing all the separations for a given page. For more information, see section 10.10.3 in the *PDF Reference*.

Set-OCG-state action

setocgstate_action_dictionary

```
setocgstate_action_dictionary =
  element State { element Group {
  attribute ref { string }
  & attribute Mode {  "ON" |  "OFF" |  "Toggle" } }* }
setocgstate_action_dictionary &= attribute PreserveRB { boolean }?
setocgstate_action_dictionary &= common_action_dictionary
```

Description

A set-OCG-state action sets the state of one or more optional content groups. For more information, see *Set-OCG-State Actions* in section 8.5.3 in the *PDF Reference*.

Signature field

signature_field_dictionary_reference

```
signature_field_dictionary_reference = common_field_dictionary
```

Description

A form field that contains a digital signature. For more information, see *Signature Fields* in section 8.6.3 in the *PDF Reference*.

Soft mask image dictionary

soft_mask_image_dictionary

```
soft_mask_image_dictionary =
   element Matte { array_of_number }?
   & attribute Width { integer }
   & attribute Height { integer }
   & attribute BitsPerComponent { integer }
   & element ColorSpaceMap { color_space_map }?
   & attribute Interpolate { boolean }?
   & attribute src { string }
   & common_stream_dictionary
```

Description

A subsidiary image XObject specified in the `SMask` entry of the parent XObject's image dictionary. For more information, see *Soft-Mask Images* in section 7.5.4 in the *PDF Reference*.

Element or attribute	Description	Default Value
ColorSpaceMap	See the `Decode` entry in Table 7.11 in the *PDF Reference*.	

Software identifier dictionaries

array_of_software_identifier_dictionaries

```
array_of_software_identifier_dictionaries = element Viewer {
software_identifier_dictionary }*
```

software_identifier_dictionary

```
software_identifier_dictionary =
   attribute URI { pdf_ascii_string }
   & attribute MinVersion { version_array }?
   & attribute MinInclusive { boolean }?
   & attribute MaxVersion { version_array }?
   & attribute MaxInclusive { boolean }?
   & element OS { os_name_array }?
```

version_array

```
version_array =  ( list { integer } )
```

os_name_array

```
os_name_array = element OSName {
   attribute _enc { token }?
   & pdf_byte_string }*
```

Description

Software identifier dictionaries allow software to be identified by name, range of versions, and operating systems. For more information, see *Software Identifier Dictionary* in section 9.1.6 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
URI	See the `U` entry in Table 9.27 in the *PDF Reference*.	
MinVersion	See the `L` entry in Table 9.27 in the *PDF Reference*.	
MinInclusive	See the `LI` entry in Table 9.27 in the *PDF Reference*.	
MaxVersion	See the `H` entry in Table 9.27 in the *PDF Reference*.	
MaxInclusive	See the `HI` entry in Table 9.27 in the *PDF Reference*.	
OSName	See the `OS` entry in Table 9.27 in the *PDF Reference*.	

Sound action

sound_action_dictionary

```
sound_action_dictionary =
   element SoundFile { sound_stream_dictionary }
   & attribute Volume { number }?
   & attribute Synchronous { boolean }?
   & attribute Repeat { boolean }?
   & attribute Mix { boolean }?
sound_action_dictionary &= common_action_dictionary
```

Description

A sound action plays a sound through the computer's speakers. For more information, see *Sound Actions* in section 8.5.3 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
SoundFile	See the `Sound` entry in Table 8.58 in the *PDF Reference*.	

Sound annotation

sound_annotation_dictionary

```
sound_annotation_dictionary = common_annotation_dictionary
sound_annotation_dictionary &= common_markup_annotation_dictionary
sound_annotation_dictionary &=
   element SoundFile { sound_stream_dictionary }
   & attribute IconName { name }?
   & attribute IconName_enc { token }?
```

Description

A sound annotation is analogous to a text annotation except that instead of a text note, it contains sound recorded from the computer's microphone or imported from a file. A sound annotation is a markup annotation stored with its associated page. For more information, see *Sound Annotations* in section 8.4.5 in the *PDF Reference*.

Element or attribute	Description	Default Value
SoundFile	See the `Sound` entry in Table 8.36 in the *PDF Reference*.	
IconName	See the `Name` entry in Table 8.36 in the *PDF Reference*.	

Sound object

sound_stream_dictionary

```
sound_stream_dictionary =
   attribute SamplingRate { number }
   & attribute ChannelCount { integer }?
   & attribute BitsPerSample { integer }?
   & attribute Encoding { pdf_text_name }?
   & attribute Compression { name }?
   & attribute Compression_enc { token }?
   & attribute src { string }
   & common_stream_dictionary
```

Description

A sound object is a stream containing sample values that define a sound to be played through the computer's speakers. For more information, see section 9.2 in the *PDF Reference*.

Element or attribute	Description	Default Value
SamplingRate	See the `R` entry in Table 9.29 in the *PDF Reference*.	
ChannelCount	See the `C` entry in Table 9.29 in the *PDF Reference*.	

BitsPerSample	See the `B` entry in Table 9.29 in the *PDF Reference*.
Encoding	See the `E` entry in Table 9.29 in the *PDF Reference*.
Compression	See the `CO` entry in Table 9.29 in the *PDF Reference*.
Compression_enc	A PDFDocEncoded string representing the same information as the `Compression` attribute. For information on string encoding, see Section 3.8.1 of the *PDF Reference*.
src	

Source information dictionary

source_information_dictionary_or_array

```
source_information_dictionary_or_array =
  element Source { source_information_dictionary }
```

source_information_dictionary

```
source_information_dictionary =
  element Alias { string_or_url_alias_dictionary }
  & attribute Timestamp { date }?
  & attribute Expiration { date }?
  & attribute SubmissionType {  "NotSubmission" |  "Get" |  "Post" }?
  & element Command { web_capture_command_dictionary_reference }?
```

Description

A `Source` element describes the source or sources from which the objects belonging to the content set were created. For more information, see section 10.9.4 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
Alias	See the `AU` entry in Table 10.41 in the *PDF Reference*.	
Timestamp	See the `TS` entry in Table 10.41 in the *PDF Reference*.	
Expiration	See the `E` entry in Table 10.41 in the *PDF Reference*.	
SubmissionType	See the `S` entry in Table 10.41 in the *PDF Reference*.	0
Command	See the `C` entry in Table 10.41 in the *PDF Reference*.	

Square and circle annotations

shape_annotation_dictionary

```
shape_annotation_dictionary = common_annotation_dictionary
shape_annotation_dictionary &= common_markup_annotation_dictionary
shape_annotation_dictionary &=
  attribute InteriorColor { rgb_color_array }?
  & element BorderEffect { border_effect_dictionary }?
  & attribute Fringe { rectangle_list }?
```

Description

Square and circle annotations display, respectively, a rectangle or an ellipse on the page. For more information, see *Square and Circle Annotations* in section 8.4.5 in the *PDF Reference*.

Element or attribute	Description	Default Value
InteriorColor	See the `IC` entry in Table 8.28 in the *PDF Reference*.	
BorderEffect	See the `BE` entry in Table 8.28 in the *PDF Reference*.	
Fringe	See the `RD` entry in Table 8.28 in the *PDF Reference*.	

Stitching function array

array_of_function_dictionary

```
array_of_function_dictionary = element Function { function_dictionary }*
```

Description

An element which represents a stitching function. For more information, see the `Functions` entry in Table 3.37 in the *PDF Reference*.

Stream object

array_stream_object

```
array_stream_object =
  attribute src { string }
  & common_stream_dictionary
```

common_stream_dictionary

```
common_stream_dictionary =
  element File { file spec dictionary }?
  & element FileFilters { filter name or array }?
  & element FDecodeParms { parm dictionary or array }?
```

Description

A sequence of bytes of unlimited length.

Each object that is represented as a stream in PDF is placed in a separate file. This includes page contents, fonts, images, ICC color profiles, user file attachments, form data, and other objects.

For more information, see section 3.2.7 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
File	See the `F` entry in Table 3.4 in the *PDF Reference*.	
FileFilters	See the `FFilter` entry in Table 3.4 in the *PDF Reference*.	
FDecodeParms	See the `FDecodeParms` entry in Table 3.4 in the *PDF Reference*.	

String or annotation dictionary reference

string_or_annotation_dictionary_reference

```
string_or_annotation_dictionary_reference =
  element Field { attribute Pathname { pdf text string } }
string_or_annotation_dictionary_reference |=
  element Annotation { attribute ref { pdf reference } }
```

Description

Elements referencing annotations (`Annotation ref = ...`) or string referencing annotations (`Pathname`) to be hidden or shown in a Hide Action or fields to be included or excluded in a Submit Form action. For more information, see the `T` entry in Table 8.60 and the `Fields` entry in Table 8.85 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
Pathname	A string giving the fully qualified field name of an interactive form field.	
Annotation	An indirect reference to an annotation dictionary.	

Strings

string_or_array_of_string

```
string_or_array_of_string =
  element Val { pdf_text_string }
string_or_array_of_string |= element Val { pdf_text_string }*
```

Description

A string or series of string elements.

Structure attributes

attribute_object

```
attribute_object =
  attribute Owner { name }
  & attribute Owner_enc { token }?
  & element UserProperties { array_of_user_property_dictionaries }?
  & attribute Placement { name }?
  & attribute Placement_enc { token }?
  & attribute WritingMode { name }?
  & attribute WritingMode_enc { token }?
  & attribute BackgroundColor { rgb_color_array }?
  & attribute Color { rgb_color_array }?
  & attribute SpaceBefore { number }?
  & attribute SpaceAfter { number }?
  & attribute StartIndent { number }?
  & attribute EndIndent { number }?
  & attribute TextIndent { number }?
  & attribute TextAlign { name }?
  & attribute TextAlign_enc { token }?
  & attribute BlockAlign { name }?
  & attribute BlockAlign_enc { token }?
  & attribute IndentAlign { name }?
  & attribute IndentAlign_enc { token }?
  & attribute BaselineShift { number }?
  & attribute TextDecorationColor { rgb_color_array }?
  & attribute TextDecorationThickness { number }?
  & attribute TextDecorationType { name }?
  & attribute TextDecorationType_enc { token }?
  & attribute RubyAlign { name }?
  & attribute RubyAlign_enc { token }?
  & attribute RubyPosition { name }?
  & attribute RubyPosition_enc { token }?
  & attribute GlyphOrientationVerticaly { name }?
  & attribute GlyphOrientationVerticaly_enc { token }?
```

Description

An `attribute_object` is represented by a number of XML attributes or a stream that includes an `owner` entry identifying the application or plug-in that owns the attribute information. For more information, see section 10.6.4 in the *PDF Reference*.

Example

```
<Attribute SpaceAfter="0" TextAlign="Start" Owner="Layout"
  WritingMode="LrTb" SpaceBefore="0" EndIndent="0" StartIndent="0"
  TextIndent="0" />
```

Element or attribute	Description	Default Value
Owner	See the `O` entry in Table 10.15 in the *PDF Reference*.	
UserProperties	See the `P` entry in Table 10.15 in the *PDF Reference*.	

Structure file element

structure_file_element

```
structure_file_element = structure_tree_element_dictionary
```

structure_tree_element_dictionary

```
structure_tree_element_dictionary =
  attribute Tag { name }
  & attribute Role { name }?
  & attribute Path { pdf_text_string }?
structure_tree_element_dictionary &=
  attribute ID { pdf_byte_string }?
  & attribute ID_enc { token }?
  & attribute Page_src { pdf_reference }?
structure_tree_element_dictionary &= structure_class?
structure_tree_element_dictionary &=
  attribute Revision { integer }?
  & attribute Title { pdf_text_string }?
  & attribute Lang { pdf_text_string }?
  & attribute Alt { pdf_text_string }?
  & attribute ActualText { pdf_text_string }?
  & attribute ExpandedName { pdf_text_string }?
structure_tree_element_dictionary &= structure_tree_a?
```

structure_tree_a

```
structure_tree_a =
  element Attributes { structure_tree_a_array }
structure_tree_a |=
  element Attribute { attribute_object }
```

structure_tree_a_array

```
structure_tree_a_array = attribute_object_or_rev_number*
```

Description

A structure file element is a child of the structure tree root that may refer to another structure file element or a content item.

In the page structure file there is a series of entries for interior nodes of the structure tree. These nodes are parents of content items on that page. Each `Struct` element describes one node in the structure tree. The `Role` and `BaseRole` indicate the *tag* and role-mapped tag of the content subtree.

For more information, see section 10.6.1 in the *PDF Reference*.

Element or attribute	Description	Default Value
Path	The path to the node.	
Role	The tag of the content subtree.	
Attribute	The attribute values of the node.	
Revision	See the `R` entry in Table 10.10 in the *PDF Reference*.	
Title	See the `T` entry in Table 10.10 in the *PDF Reference*.	
ExpandedName	See the `E` entry in Table 10.10 in the *PDF Reference*.	

Structure tree cache file

cache

```
cache =
   element Cache { cache_header cache_body }
```

cache_header

```
cache_header =
   element Query {
   element Files { element Pattern { attribute Value { text } }+ }?
   & attribute XPath { text }?
   & attribute Translate { text }? }?
   & attribute Identifier { text }?
   & element Timestamp {
   attribute Type { text }?
   & attribute Time { text }?
   & attribute DocumentID { pdf_base64_string }?
   & attribute InstanceID { pdf_base64_string }? }?
```

cache_body

```
cache_body =
   element Data { cache_data }
```

cache_data

```
cache_data = structure_cache_contents*
```

structure_cache_contents

```
structure_cache_contents =
   element MCR { marked_content_reference }
   & element Elem { structure_file_element }
```

structure_file_element

```
structure_file_element =
   attribute Path { pdf_text_string }
   & attribute ref { string }
```

marked_content_reference_dictionary

marked_content_reference

```
marked_content_reference =
   attribute ref { string }
   & attribute Path { pdf_text_string }
```

Description

The shape of the structure tree is cached and stored in the document structure-tree cache file. The cache contains an entry for each interior structure tree node indicating which page contains information about it. Since more than one page may contain (the same) information about a node, any such page can be used in the cache entry. The cache itself lists all of the structure tree nodes and their path names; this is sufficient information to allow a traversal of the tree. Individual nodes can then be filled in on demand.

The cache consists of a header and data sections. The header section provides enough information to allow the data section of the cache to be regenerated. The data section contains the structure tree. The Mars plug-in does not use the structure tree cache header to create cache data.

The data section is a sequence of `Path` elements followed by `MCR` elements which represent the interior nodes of the structure tree. For each node, the full path is included along with a reference to the page information file for the page containing the full information for the node.

Element or attribute	Description	Default Value
Pattern	Provides a regular expression that identifies the files to be searched for structure elements.	

XPath	`XPath` or `Translate` or both are applied to those files selected for caching data. The resulting nodes are concatenated and placed into the cache.
Translate	`XPath` or `Translate` or both are applied to those files selected for caching data. The resulting nodes are concatenated and placed into the cache.
Identifier	
Timestamp	

Structure tree root

structure_tree_root_dictionary

```
structure_tree_root_dictionary =
   element RoleMap { role_map_dictionary }?
   & element ClassMap { class_map_dictionary }?
```

role_map_dictionary

```
role_map_dictionary = element Role { attribute Name { text }
   attribute Name_enc { token }? role_name }+
```

role_name

```
role_name =
   attribute MapTo { name }
   & attribute MapTo_enc { token }?
```

class_map_dictionary

```
class_map_dictionary = element Class { attribute Name { text }
   attribute Name_enc { token }? structure_tree_a_orig }+
```

structure_tree_a_orig

```
structure_tree_a_orig =
   element Attributes { structure_tree_a_array }
structure_tree_a_orig |=
   element Attribute { attribute_object }
```

attribute_object_or_rev_number

```
attribute_object_or_rev_number =
   element Attribute { attribute_object }
```

Description

The logical structure of a document is described by a hierarchy of objects called the structure hierarchy or structure tree. At the root of the hierarchy is a dictionary object called the structure tree root, located by means of the `StructureTypes` entry in the document catalog.

In Mars, the structure tree is not represented as a single tree, but is broken into separate pieces of information that are stored in different locations in the document package. The structure tree is broken down as follows: leaf nodes are represented as part of markup corresponding to marked content containers nested within the SVG page content. Information about interior structure nodes that are parents of the content on a page is stored in a separate file associated with that page. Finally, information indicating which pages have the structure information for which interior nodes is stored in a cache file.

The Role Map and Class Map are stored in the document backbone. The Role Map maps author-defined structure tags to standard structure tags. The Class Map defines default attribute values for particular classes of attributes. The Role Map and Class Map are described in the document backbone schema.

For more information, see section 10.6.1 in the *PDF Reference*. For information specific to Mars, see Specifying Marked Content, Logical Structure and Tagging and Adding Foreign Data, both in the guide *Developing Applications Using Mars*.

Example

```
<StructureTypes>
  <ClassMap>
    <Class name="CM1">
      <Attribute Owner="Layout" TextAlign="Center"/>
    </Class>
    <Class name="CM2">
      <Attribute Owner="Layout" TextAlign="None"/>
    </Class>
  </ClassMap>
</StructureTypes>
```

Submit-form action

submitform_action_dictionary

```
submitform_action_dictionary =
  element File { file_spec_dictionary }
  & element Fields { fields_array }?
  & attribute Flags {  ( list {  "Exclude"? "IncludeNoValueFields"?
  "ExportFormat"? "GetMethod"? "SubmitCoordinates"? "XFDF"?
  "IncludeAppendSaves"? "IncludeAnnotations"? "SubmitPDF"?
  "CanonicalFormat"? "ExclNonUserAnnots"? "ExclFKey"? "EmbedForm"? } ) }?
submitform_action_dictionary &= common_action_dictionary
```

Description

A submit-form action transmits the names and values of selected interactive form fields to a specified uniform resource locator (URL), presumably the address of a web server that will process them and send back a response. For more information, see *Submit-Form Actions* in section 8.6.4 in the *PDF Reference*.

Element or attribute	Description	Default Value
File	See the `F` entry in Table 8.85 in the *PDF Reference*.	

Target dictionary

target_dictionary

```
target_dictionary =
  attribute Relationship {  "Parent" |  "Child" }
  & attribute Filename { pdf_byte_string }?
  & attribute Filename_enc { token }?
target_dictionary &= destination_or_page?
target_dictionary &= annotation_name_or_number?
target_dictionary &= element Target { target_dictionary }?
```

destination_or_page

```
destination_or_page =
  attribute Dest { pdf_byte_string }
  & attribute Dest_enc { token }?
destination_or_page |=
  attribute PageNum { nonNegativeInteger }
```

annotation_name_or_number

```
annotation_name_or_number =
  attribute AnnotName { pdf_text_string }
annotation_name_or_number |=
  attribute AnnotNum { nonNegativeInteger }
```

Description

A target dictionary, specified by the `Target` element in an Embedded go-to action, locates the target in relation to the source, in much the same way that a relative path describes the physical relationship between two files in a file system. For more information, see *Embedded Go-To Actions* in section 8.5.3 in the *PDF Reference*.

Element or attribute	Description	Default Value
Relationship	See the `R` entry in Table 8.52 in the *PDF Reference*.	
Filename	See the `N` entry in Table 8.52 in the *PDF Reference*.	

Filename_enc	A PDFDocEncoded string representing the same information as the `Filename` attribute. For information on string encoding, see Section 3.8.1 of the *PDF Reference*.
Dest	Specifies a named destination in the current document that provides the page number of the file attachment annotation. See the `P` entry in Table 8.52 in the *PDF Reference*.
PageNum	Specifies the zero-based page number in the current document containing the file attachment annotation. See the `P` entry in Table 8.52 in the *PDF Reference*.
AnnotName	Specifies the value of `NM` in the annotation dictionary. See the `A` entry in Table 8.52 in the *PDF Reference*.
AnnotNum	Specifies the zero-based index of the annotation in the `Annotations` array of the page specified by `PageNum`. See the `A` entry in Table 8.52 in the *PDF Reference*.

Template page name tree

template_page_name_tree

```
template_page_name_tree = element Page { attribute ref { pdf_reference } }?
```

Description

An element containing a series of elements that maps name strings to invisible (template) pages for use in interactive forms. For more information, see section 8.6.5, as well as the `Templates` entry in Table 3.28 in the *PDF Reference*.

Text annotation

text_annotation_dictionary

```
text_annotation_dictionary = common_annotation_dictionary
text_annotation_dictionary &= common_markup_annotation_dictionary
text_annotation_dictionary &=
   attribute Open { boolean }?
   & attribute IconName { name }?
   & attribute IconName_enc { token }?
   & attribute State { pdf_text_string }?
   & attribute StateModel { pdf_text_string }?
```

Description

A text annotation represents a *sticky note* attached to a point in the document. For more information, see *Text Annotations* in section 8.4.5 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
IconName	See the `Name` entry in Table 8.23 in the *PDF Reference*.	

Text field

text_field_dictionary_reference

```
text_field_dictionary_reference =
  element DefaultValue { pdf_text_string }?
  & attribute MaxLen { integer }?
text_field_dictionary_reference &= common_field_dictionary
```

Description

A box or space in which the user can enter text from the keyboard. Unlike PDF, the form field values are represented in a separate file rather than in this field dictionary. That separate file follows the XFDF schema for form fields; it does not support XFDF-defined annotations. The file must be named /form/form_data.xfdf. For more information, see *Text Fields* in section 8.6.3 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
DefaultValue	See the `DV` entry in Table 8.69 in the *PDF Reference*.	
MaxLen	See the `MaxLen` entry in Table 8.78 in the *PDF Reference*.	

Text markup annotation

text_markup_annotation_dictionary

```
text_markup_annotation_dictionary = common_annotation_dictionary
text_markup_annotation_dictionary &= common_markup_annotation_dictionary
text_markup_annotation_dictionary &=
  attribute Coords { quadpoints_array }
```

quadpoints_array

```
quadpoints_array =  ( list { number } )
```

Description

Text markup annotations appear as highlights, underlines, strikeouts, or jagged (*squiggly*) underlines in the text of a document. For more information, see *Text Markup Annotations* in section 8.4.5 in the *PDF Reference*.

Element or attribute	Description	Default Value
Coords	See the `QuadPoints` entry in Table 8.30 in the *PDF Reference*.	

Text string

pdf_text_string

```
pdf_text_string = string
```

Description

A string containing information that is intended to be human-readable. For more information, see section 3.8.1 in the *PDF Reference*.

Thread action

thread_action_dictionary

```
thread_action_dictionary =
  element File { file_spec_dictionary }?
  & element ArticleThread { thread_action_thread_reference }
  & element Bead { thread_action_bead_reference }?
thread_action_dictionary &= common_action_dictionary
```

Description

A thread action causes a viewing application to display a specified bead on an article thread in either the current document or a different one. For more information, see *Thread Actions* in section 8.5.3 in the *PDF Reference*.

Element or attribute	Description	Default Value
File	See the `F` entry in Table 8.55 in the *PDF Reference*.	
Thread	See the `D` entry in Table 8.55 in the *PDF Reference*.	
Bead	See the `B` entry in Table 8.55 in the *PDF Reference*.	

Thread action bead reference

thread_action_bead_reference

```
thread_action_bead_reference =
  attribute Index { integer }
thread_action_bead_reference |=
  attribute ref { pdf_reference }
```

Description

The bead in the destination thread. For more information, see the `B` entry in Table 8.55 in the *PDF Reference*.

Element or attribute	Description	Default Value
Index	The index of the bead within its thread.	
BeadRef	An indirect reference to a bead dictionary.	

Thread action thread reference

thread_action_thread_reference

```
thread_action_thread_reference =
  attribute Index { integer }
thread_action_thread_reference |=
  attribute Title { pdf_text_string }
thread_action_thread_reference |=
  attribute ref { pdf_reference }
```

Description

The destination thread of a thread action. For more information, see the `D` entry in Table 8.55 in the *PDF Reference*.

Element or attribute	Description	Default Value
Index	The index of the thread within the `Threads` array of its document's catalog.	
Title	The title of the thread as specified in its thread information dictionary.	

Top-level element

root

```
root =
  element PDF { catalog_dictionary }
```

Description

The top level element is a Mars file.

Transformation matrix

coordinate_map_array

```
coordinate_map_array =  ( list { number number number number number number }
)
```

Description

A matrix used in transformations between coordinate systems. For more information, see section 4.2.3 in the *PDF Reference*.

Transition action

trans_action_dictionary

```
trans_action_dictionary =
   element Trans { transition_dictionary }
trans_action_dictionary &= common_action_dictionary
```

Description

A transition action can be used to control drawing during a sequence of actions. For more information, see *Transition Actions* in section 8.5.3 in the *PDF Reference*.

Transition dictionary

transition_dictionary

```
transition_dictionary =
   attribute Duration { number }?
   & attribute Style {  "Split" |  "Blinds" |  "Box" |  "Wipe" |  "Dissolve" |
   "Glitter" |  "R" |  "Fly" |  "Push" |  "Cover" |  "Uncover" |  "Fade" }?
   & attribute Dimension {  "Horizontal" |  "Vertical" }?
   & attribute Motion {  "Inward" |  "Outward" }?
   & attribute Direction { number }?
   & attribute FlyScale { number }?
   & attribute FlyRectangular { boolean }?
```

Description

A transition dictionary describes the style and duration of the visual transition to use when moving from another page to the given page during a presentation. For more information, see section 8.3.3 and Table 8.13 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
Duration	See the `D` entry in Table 8.13 in the *PDF Reference*.	R
Style	See the `S` entry in Table 8.13 in the *PDF Reference*.	
Direction	See the `Di` entry in Table 8.13 in the *PDF Reference*.	0
Motion	See the `M` entry in Table 8.13 in the *PDF Reference*.	
Dimension	See the `Dm` entry in Table 8.13 in the *PDF Reference*.	H
FlyScale	See the `SS` entry in Table 8.13 in the *PDF Reference*.	
FlyRectangular	See the `B` entry in Table 8.13 in the *PDF Reference*.	

Trap network annotation

trap_network_annotation_dictionary

```
trap_network_annotation_dictionary =
  element Version { trap_array_of_object_references }?
  & element AnnotStates { array_of_name }?
  & element FontFauxing { trap_array_of_object_references }?
trap_network_annotation_dictionary &= common_annotation_dictionary
```

Description

A complete set of traps generated for a given page under a specified set of trapping instructions is called a trap network. For more information, see *MTrap Network Annotations* in section 10.10.5 in the *PDF Reference*.

Tristimulus value

tristimulus_value

```
tristimulus_value =  ( list { number number number } )
```

Description

X, Y, and Z components of the CIE 1931 XYZ space. For more information, see section 4.5.4 in the *PDF Reference*.

TrueType font

truetype_font_dictionary

```
truetype_font_dictionary = common_font_dictionary
```

Description

A TrueType font dictionary. For more information, see section 5.5.2 in the *PDF Reference*.

Type 0 (sampled) function

sampled_function_dictionary

```
sampled_function_dictionary =
   attribute Domain { array_of_number }
   & attribute Range { array_of_number }
   & attribute Size { array_of_integer }
   & attribute BitsPerSample { integer }
   & attribute Order { number }?
   & attribute Encode { array_of_number }?
   & attribute Decode { array_of_number }?
   & attribute src { string }
   & common_stream_dictionary
```

Description

Type 0 functions use a sequence of sample values to provide an approximation for functions whose domains and ranges are bounded. For more information, see section 3.9.1 in the *PDF Reference*.

Type 0 font

type0_font_dictionary

```
type0_font_dictionary = element DescendantFonts {
array_of_1_CIDFontDictionaries }?
type0_font_dictionary &= common_font_dictionary
```

array_of_1_CIDFontDictionaries

```
array_of_1_CIDFontDictionaries =
   element CIDFont { CIDFont_dictionary }
```

Description

A composite font whose glyphs are obtained from a font-like object called a CIDFont. For more information, see section 5.6 in the *PDF Reference*.

Type 1 font

type1_font_dictionary

```
type1_font_dictionary = common_font_dictionary
```

Description

A stylized PostScript program that describes glyph shapes. For more information, see section 5.5.1 in the *PDF Reference*.

Type 2 (exponential interpolation) function

interpolated_function_dictionary

```
interpolated_function_dictionary =
  attribute Domain { array_of_number }
  & attribute Range { array_of_number }
  & attribute C0 { array_of_number }?
  & attribute C1 { array_of_number }?
  & attribute Exponent { number }
```

Description

Type 2 functions include a set of parameters that define an exponential interpolation of one input value and n output values. For more information, see section 3.9.2 in the *PDF Reference*.

Element or attribute	Description	Default Value
Exponent	See the `N` entry in Table 3.36 in the *PDF Reference*.	

Type 3 (stitching) function

stitching_function_dictionary

```
stitching_function_dictionary =
  attribute Domain { array_of_number }
  & attribute Range { array_of_number }
  & element StitchingFunctions { array_of_function_dictionary }
  & attribute Bounds { array_of_number }
  & attribute Encode { array_of_number }
```

Description

Type 3 functions define a stitching of the subdomains of several 1-input functions to produce a single new 1-input function. For more information, see section 3.9.3 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
StitchingFunctions	See the `Functions` entry in Table 3.37 in the *PDF Reference*.	

Type 4 (PostScript calculator) function

postscript_function_dictionary

```
postscript_function_dictionary =
   attribute Domain { array_of_number }
   & attribute Range { array_of_number }
   & attribute src { string }
   & common_stream_dictionary
```

Description

A type 4 function, also called a PostScript calculator function, is represented as a stream containing code written in a small subset of the PostScript language. For more information, see section 3.9.4 in the *PDF Reference*.

U3D path

string_or_u3d_path

```
string_or_u3d_path = element PathSegment { pdf_text_string }*
string_or_u3d_path |=
   element PathSegment { pdf_text_string }
```

Description

A sequence of one or more strings used to access a view node within the U3D artwork. For more information, see the `U3DPath` entry in Table 9.37 in the *PDF Reference*.

URI action

uri_action_dictionary

```
uri_action_dictionary =
   attribute URI { pdf_ascii_string }
   & attribute IsMap { boolean }?
uri_action_dictionary &= common_action_dictionary
```

Description

A URI action causes a URI to be resolved. For more information, see *URI Actions* in section 8.5.3 in the *PDF Reference*.

URI dictionary

uri_base_dictionary

```
uri_base_dictionary =
   attribute Base { pdf_ascii_string }
```

Description

A URI dictionary containing document-level information for URI (uniform resource identifier) actions. For more information, see the `URI` entry in Table 3.25, as well as Table 8.57, in the *PDF Reference*.

Example

```
<URI URI="http://www.adobe.com"/>
```

URL alias dictionaries

string_or_url_alias_dictionary

```
string_or_url_alias_dictionary = pdf_ascii_string
string_or_url_alias_dictionary |= url_alias_dictionary
```

url_alias_dictionary

```
url_alias_dictionary =
   attribute Target { pdf_ascii_string }
   & element Aliases { array_of_aliases }?
```

array_of_aliases

```
array_of_aliases = element Alias { pdf_ascii_string }*
```

Description

When a URL is accessed via HTTP, a response header may be returned indicating that the requested data is at a different URL. This redirection process may be repeated in turn at the new URL and can potentially continue indefinitely. It is not uncommon to find multiple URLs that all lead eventually to the same destination through one or more redirections. A URL alias dictionary represents such a set of URL chains leading to a common destination. For more information, see *URL Alias Dictionaries* in section 10.9.4 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
Target	See the `U` entry in Table 10.42 in the *PDF Reference*.	
Aliases	See the `C` entry in Table 10.42 in the *PDF Reference*.	

Usage application dictionaries

usage_application_dictionary_array

```
usage_application_dictionary_array = element UsageApplication {
oc_usage_application_dictionary }*
```

oc_usage_application_dictionary

```
oc_usage_application_dictionary =
   attribute Event { name }
   & attribute Event_enc { token }?
   & element Groups { oc_group_dictionary_reference_array }?
   & element Usages { usage_name_array }
```

usage_name_array

```
usage_name_array = element Usage {
   attribute _enc { token }?
   & name }*
```

Description

A usage application dictionary specifies the rules for which usage entries should be used by viewer applications to automatically manipulate the state of optional content groups, which groups should be affected, and under which circumstances. For more information, see the `AS` entry in Table 4.51, and *Usage and Usage Application Dictionaries* in section 4.10.3 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
Groups	See the `OCGs` entry in Table 4.53 in the *PDF Reference*.	
Usages	See the `Category` entry in Table 4.53 in the *PDF Reference*.	

User properties

array_of_user_property_dictionaries

```
array_of_user_property_dictionaries = element UserProperty {
user_property_dictionary }*
```

user_property_dictionary

```
user_property_dictionary =
   attribute Name { pdf_text_string }
   & attribute FormattedValue { pdf_text_string }?
   & attribute Hidden { boolean }?
user_property_dictionary &= default_dict_element
```

Description

User properties, contained in page structure attributes, that can be used to contain information that is reflected in an element's appearance. For more information, see *User Properties* in section 10.6.4 in the *PDF Reference*.

Element or attribute	Description	Default Value
Name	See the `N` entry in Table 10.16 in the *PDF Reference*.	
FormattedValue	See the `F` entry in Table 10.16 in the *PDF Reference*.	
Hidden	See the `H` entry in Table 10.16 in the *PDF Reference*.	false

Variable text

variable_text_dictionary_fields

```
variable_text_dictionary_fields =
  element DefaultAppearance { attribute Font { text }
    attribute TextSize { number }?
    attribute FillColorSpace { "Gray" | "RGB" | "CMYK" }?
    attribute FillColor { number | ( list { number number numbernumber? }
    )}?
    attribute StrokeColorSpace { "Gray" | "RGB" | "CMYK" }?
    attribute StrokeColor { number | ( list { number number numbernumber? }
    ) }?
    attribute CharacterSpacing { number }?
    attribute TextLeading { number }?
    attribute TextMatrix { ( list { number number number number number
    number } ) }?
    attribute RenderMode { number }?
    attribute TextRise { number }?
    attribute WordSpacing { number }?
    attribute HorizontalScaling { number }?
  }
variable_text_dictionary_fields &=
  attribute Justification { "Left" | "Centered" | "Right" }?
  & attribute DefaultStyle { pdf_text_string }?
  & element DefaultRichTextValue { rt_string_or_stream }?
```

Description

A field that may contain text whose value is not known until viewing time. For more information, see *Variable Text* in section 8.6.2 in the *PDF Reference*.

Element or attribute	Description	Default Value
DefaultAppearance	See the `DA` entry in Table 8.71 in the *PDF Reference*.	

Justification	See the `Q` entry in Table 8.71 in the *PDF Reference*.
DefaultStyle	See the `DS` entry in Table 8.71 in the *PDF Reference*.
DefaultRichTextValue	See the `RV` entry in Table 8.71 in the *PDF Reference*.

Version 1.3 OPI dictionary

opi_13_dictionary

```
opi_13_dictionary =
  element File { file_spec_dictionary }
  & attribute ID { pdf_byte_string }?
  & attribute ID_enc { token }?
  & attribute Comments { pdf_text_string }?
  & element Size { opi_size_array }
  & element CropRect { opi_crop_rect_array }
  & element CropFixed { opi_crop_rect_array }?
  & element Position { opi_position_array }
  & element Resolution { opi_resolution_array }?
  & attribute ColorType { pdf_text_name }?
  & element Color { opi_cmyk_color_array }?
  & attribute Tint { number }?
  & element Overprint { boolean }?
  & element ImageType { opi_image_type_array }?
  & element GrayMap { opi_graymap_array }?
  & attribute Transparency { boolean }?
  & element Tags { opi_tags_array }?
```

opi_tags_array

```
opi_tags_array = default_array_element*
```

opi_graymap_array

```
opi_graymap_array =  ( list { integer } )
```

Description

In OPI 1.3, a proxy consisting of a single image, with no changes in the graphics state, may be represented as an image XObject; otherwise it must be a form XObject. For more information, see section 10.10.6 and Table 10.55 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
File	See the `F` entry in Table 10.55 in the *PDF Reference*.	

Version 2.0 OPI dictionary

opi_20_dictionary

```
opi_20_dictionary =
   element File { file_spec_dictionary }
   & element MainImage {
   attribute _enc { token }?
   & pdf_byte_string }?
   & element Tags { opi_tags_array }?
   & element Size { opi_size_array }?
   & element CropRect { opi_crop_rect_array }?
   & element Overprint { boolean }?
   & element Inks { opi_ink_info }?
   & element Dimensions { opi_size_array }?
   & element IncludedImageQuality { number }?
```

opi_tags_array

```
opi_tags_array = default_array_element*
```

opi_ink_info

```
opi_ink_info = name
opi_ink_info |= default_array_element*
```

Description

In OPI 2.0, the proxy always entails changes in the graphics state and hence must be represented as a form XObject. For more information, see section 10.10.6 and Table 10.56 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
File	See the `F` entry in Table 10.56 in the *PDF Reference*.	
Dimensions	See the `IncludedImageDimensions` entry in Table 10.56 in the *PDF Reference*.	

View parameter name tree

view_parameter_name_tree

```
view_parameter_name_tree =
   element Resource {
   attribute Key { text }
   & default_dictionary }
```

Description

An XML structure used to map strings to objects that can be used by applications or scripts to modify the default view of the 3D artwork. For more information, see the `Resources` entry in Table 9.35 in the *PDF Reference*.

Element or attribute	Description	Default Value
Key	Name associated with the view.	

Viewer preferences dictionary

viewer_preferences_dictionary

```
viewer_preferences_dictionary =
  attribute HideToolbar { boolean }?
  & attribute HideMenubar { boolean }?
  & attribute HideWindowUI { boolean }?
  & attribute FitWindow { boolean }?
  & attribute CenterWindow { boolean }?
  & attribute DisplayDocTitle { boolean }?
  & attribute NonFullScreenPageMode { name }?
  & attribute NonFullScreenPageMode_enc { token }?
  & attribute Direction { pdf_text_name }?
  & attribute ViewArea { name }?
  & attribute ViewArea_enc { token }?
  & attribute ViewClip { name }?
  & attribute ViewClip_enc { token }?
  & attribute PrintArea { name }?
  & attribute PrintArea_enc { token }?
  & attribute PrintClip { name }?
  & attribute PrintClip_enc { token }?
  & attribute PrintScaling { pdf_text_name }?
  & attribute Duplex { pdf_ascii_string }?
  & attribute PickTrayByPDFSize { boolean }?
  & attribute PrintPageRange { array_of_integer }?
  & attribute NumCopies { integer }?
```

Description

An element controlling the way the document is to be presented on the screen or in print.

Viewer preferences in PDF documents control view settings when the file is opened and other viewer behaviors. The `ViewerPreferences` element is a child of the PDF element in the Mars backbone.

For more information, see the `ViewerPreferences` entry in Table 3.25, and section 8.1 in the *PDF Reference*.

Example

```
<ViewerPreferences
  CenterWindow = "true"
  DisplayDocTitle = "true"
```

```
    HideToolbar = "false"
    NonFullScreenPageMode = "UseOC"
/>
```

Viewport dictionaries

viewport_dictionary_array

```
viewport_dictionary_array = element ViewPort { viewport_dictionary }*
```

viewport_dictionary

```
viewport_dictionary =
   element BBox { rectangle_blank_sep }
   & attribute Name { pdf_text_string }?
   & element Measure { measure_dictionary }?
```

Description

A viewport is a rectangular region of a page. Viewports allow different measurement scales (specified by the `Measure` entry) to be used in different areas of a page, if necessary. For more information, see section 8.8 in the *PDF Reference*.

Watermark annotation

watermark_annotation_dictionary

```
watermark_annotation_dictionary = element FixedPrint {
fixed_print_dictionary }?
watermark_annotation_dictionary &= common_annotation_dictionary
```

Description

An annotation used to represent graphics that are expected to be printed at a fixed size and position on a page, regardless of the dimensions of the printed page. For more information, see *Watermark Annotations* in section 8.4.5 in the *PDF Reference*.

Web Capture command dictionary

wc_string_or_stream

```
wc_string_or_stream = pdf_byte_string
wc_string_or_stream |=
   attribute src { string }
   & common_stream_dictionary
```

web_capture_command_dictionary

```
web_capture_command_dictionary =
  attribute URL { pdf_ascii_string }
  & attribute Levels { integer }?
  & attribute Flags {  ( list {  "SameSite"? "SamePath"? "Submit"? } ) }?
  & element Post { wc_string_or_stream }?
  & attribute ContentType { pdf_ascii_string }?
  & element HTTPHeaders {
  attribute _enc { token }?
  & pdf_byte_string }?
  & element Settings { web_capture_command_settings_dictionary }?
```

web_capture_command_settings_dictionary

```
web_capture_command_settings_dictionary =
  element Global { web_capture_engine_settings_dictionary }?
  & element Engines { web_capture_engine_class_settings_dictionary }?
```

web_capture_engine_class_settings_dictionary

```
web_capture_engine_class_settings_dictionary = element ConversionSettings {
attribute Name { text }
attribute Name_enc { token }? web_capture_engine_settings_dictionary }+
```

web_capture_engine_settings_dictionary

```
web_capture_engine_settings_dictionary = default_dictionary
```

Description

Represents a command executed by Web Capture to retrieve one or more pieces of source data that were used to create new pages or modify existing pages. For more information, see *Command Dictionaries* in section 10.9.4 in the *PDF Reference*.

Element or attribute	Description	Default Value
URL	See the `URL` entry in Table 10.43 in the *PDF Reference*.	
Levels	See the `L` entry in Table 10.43 in the *PDF Reference*.	1
Flags	See the `F` entry in Table 10.43 in the *PDF Reference*.	0
Post	See the `P` entry in Table 10.43 in the *PDF Reference*.	
ContentType	See the `CT` entry in Table 10.43 in the *PDF Reference*.	

HTTPHeaders	See the `H` entry in Table 10.43 in the *PDF Reference*.	
Settings	See the `S` entry in Table 10.43 in the *PDF Reference*.	

Web Capture content set

web_capture_content_set_dictionary

```
web_capture_content_set_dictionary =
  attribute Subtype { pdf_text_name }
  & attribute WebCaptureId { pdf_base64_string }
  & element ContentObjects { array_of_object_reference }
  & element Sources { source_information_dictionary_or_array }
  & attribute MimeType { pdf_ascii_string }?
  & attribute Timestamp { date }?
  & attribute SetTitle { pdf_text_string }?
  & attribute PageId { pdf_base64_string }?
  & element RefCounts { ref_integer_or_array }
```

Description

A dictionary describing a set of PDF objects generated from the same source data. For more information, see section 10.9.3 in the *PDF Reference*.

Element or attribute	**Description**	**Default Value**
Subtype	See the `S` entry in Table 10.38 in the *PDF Reference*.	
WebCaptureId	See the `ID` entry in Table 10.38 in the *PDF Reference*.	
ContentObjects	See the `O` entry in Table 10.38 in the *PDF Reference*.	
Sources	See the `SI` entry in Table 10.38 in the *PDF Reference*.	
MimeType	See the `CT` entry in Table 10.38 in the *PDF Reference*.	
Timestamp	See the `TS` entry in Table 10.38 in the *PDF Reference*.	
SetTitle	See the `T` entry in Table 10.39 in the *PDF Reference*.	
PageId	See the `TID` entry in Table 10.39 in the *PDF Reference*.	
RefCounts	See the `R` entry in Table 10.40 in the *PDF Reference*.	

Web Capture content set name tree

web_capture_content_set_name_tree

```
web_capture_content_set_name_tree =
  element Url {
  attribute Key { text }
  & web_capture_content_set_dictionary }
```

Description

An element that maps URLs to Web Capture content sets. For more information, see section 10.9.3, as well as the `URLS` entry in Table 3.28 in the *PDF Reference*.

Web Capture identifiers name tree

web_capture_ids_name_tree

```
web_capture_ids_name_tree =
  element WebId {
  attribute Key { text }
  & web_capture_content_set_dictionary }
```

Description

An element that maps digital identifiers to Web Capture content sets. For more information, see section 10.9.3, as well as the `IDS` entry in Table 3.28 in the *PDF Reference*.

Web Capture information dictionary

web_capture_information_dictionary

```
web_capture_information_dictionary =
  attribute WebCaptureVersion { number }
  & element Commands { array_of_web_capture_command_dictionary_reference }?
```

array_of_web_capture_command_dictionary_reference

```
array_of_web_capture_command_dictionary_reference =
  element WebCaptureCommand { attribute Name { text }
    attribute Name_enc { token }?
    web_capture_command_dictionary_reference }+
```

web_capture_command_dictionary_reference

```
web_capture_command_dictionary_reference = web_capture_command_dictionary
```

Description

Contains document-level information related to Web Capture. For more information, see section 10.9.1 in the *PDF Reference*.

Element or attribute	Description	Default Value
WebCaptureVersion	See the `V` entry in Table 10.37 in the *PDF Reference*.	
Commands	See the `C` entry in Table 10.37 in the *PDF Reference*.	

Widget annotation

widget_annotation_dictionary

```
widget_annotation_dictionary = common_annotation_dictionary
widget_annotation_dictionary &=
   attribute Highlight {  "None" |  "Invert" |  "Outline" |  "Push" |
   "Toggle" }?
   & element AppearanceCharacteristics {
   appearance_characteristics_dictionary }?
   & element OnActivation { action_dictionary }?
widget_annotation_dictionary &=
widget_annotation_additional_actions_dictionary?
```

Description

Interactive forms use widget annotations to represent the appearance of fields and to manage user interactions. For more information, see *Widget Annotations* in section 8.4.5 in the *PDF Reference.*

Element or attribute	Description	Default Value
Highlight	See the `H` entry in Table 8.39 in the *PDF Reference*.	
AppearanceCharacteristics	See the `MK` entry in Table 8.39 in the *PDF Reference*.	
OnActivation	See the `A` entry in Table 8.39 in the *PDF Reference*.	

Windows launch parameter dictionary

windows_launch_dictionary

```
windows_launch_dictionary =
  attribute Name { pdf_byte_string }
  & attribute Name_enc { token }?
  & attribute Dir { pdf_byte_string }?
  & attribute Dir_enc { token }?
  & attribute Action { pdf_ascii_string }?
  & attribute Params { pdf_byte_string }?
  & attribute Params_enc { token }?
```

Description

A dictionary containing Windows-specific parameters for launching the designated application. For more information, see *Launch Actions* in section 8.5.3 in the *PDF Reference*.

Element or attribute	Description	Default Value
Name	See the `F` entry in Table 8.54 in the *PDF Reference*.	
Dir	See the `D` entry in Table 8.54 in the *PDF Reference*.	
Action	See the `O` entry in Table 8.54 in the *PDF Reference*.	
Params	See the `P` entry in Table 8.54 in the *PDF Reference*.	

XFA/XML form objects

stream_or_array_of_XFA

```
stream_or_array_of_XFA =
  element XDP { XDP_stream }
stream_or_array_of_XFA |=
  element FormPart { attribute src { text } attribute Name { text } }+
```

XDP_stream

```
XDP_stream = common_stream_dictionary
XDP_stream &=
  attribute src { string }
```

Description

The XFA/XML form objects reference files that contain the parts that comprise XML forms. These parts correspond to the parts that can appear in an Adobe XML Data Package (XDP).

The first and last entry in an `XFA` element reference incomplete XML files that contain the opening and closing parts of the root element in an XDP file. The other entries reference files that contain other form parts.

Example

```
<AcroForm >
  <XFA>
    <FormPart Name="xdp:xdp" src="/form/xfa_1914032634-1.xml"/>
    <FormPart Name="config" src="/form/config.xml"/>
    <FormPart Name="template" src="/form/template.xft"/>
    <FormPart Name="datasets" src="/form/datasets.xml"/>
    <FormPart Name="localeSet" src="/form/localeSet"/>
    <FormPart Name="PDFSecurity" src="/form/PDFSecurity"/>
    <FormPart Name="form" src="/form/form"/>
    <FormPart Name="/xdp:xdp" src="/form/xfa_1914032634-2.xml"/>
  </XFA>
</AcroForm>
```

Element or attribute	**Description**	**Default Value**
Name	Name of the XML form part. The string names can be arbitrary, with these exceptions: ● Opening XDP entry must be identified by the string `xdp:xdp`. ● XFA template must be identified by the string `template`. ● XFA datasets must be identified y the string `datasets`. ● XFA configuration must be identified by the string `"config"`. ● Closing XDP entry must be identified by the string `/xdp:xdp`. For more information, see section 8.6.7 in the PDF Reference for more information.	
src	Location of the file containing the XML form part.	

A Universal Container Format

The files that comprise a Mars document must be packaged as specified by Universal Container Format (UCF).

Overview

This section introduces this specification and the conventions it uses.

Purpose and Scope

This specification defines the Universal Container Format. UCF is a general-purpose container technology. It is based on the packaging principles of OCF, the OEBPS 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. The goal of this specification 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 specification borrows heavily from the OCF specification. Where possible, the same language is used, with the permission of the IDPF.

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).

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

Mars

An XML representation of PDF.

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/business_and_developers/developer/appnote).

Relationship to other specifications

This specification 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/business_and_developers/developer/appnote)
- 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)

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.

Conforming Containers

The term “Conforming UCF Abstract Container” indicates an UCF Abstract Container (See “Abstract Container vs. Physical Container” on page 224) 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 (“Abstract Container vs. Physical Container” on page 224) 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).

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.

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.

UCF Overview

This section introduces conceptual issues related to UCF.

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.

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. This specification 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. "File Names" on page 228 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 "Zip Container" on page 233.

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.

Examples

This section is informative.

This section includes an example of a OEBPS and Mars files.

Example of an OEBPS file

To illustrate the concepts from the previous section, let's assume we have a single OEBPS 1.0 document of Dickens' "Great Expectations" which consists of an OEBPS 1.2 package file ("Great Expectations.opf") and a large number of HTML files, one for the cover page (e.g., "cover.html") and one for each chapter (e.g., "chapter01.html"). The contents of the publication might be as follows:

```
OEBPS 1.2 Publication:
Great Expectations.opf
cover.html
chapters/
    chapter01.html
    chapter02.html
    … other HTML files for the remaining chapters …
```

The contents of the Abstract Container include all of the assets from the Publication, plus a small number of files defined by UCF within the META-INF directory. Note that container.xml is REQUIRED by OCF even though it is not required by UCF. See "META-INF" on page 229 for descriptions of the files within the META-INF directory.

```
Abstract Container:
mimetype
META-INF/
   container.xml
   [manifest.xml]
   [metadata.xml]
   [signatures.xml]
   [encryption.xml]
   [rights.xml]
OEBPS/
   Great Expectations.opf
   cover.html
   chapters/
   chapter01.html
   chapter02.html
   … other HTML files for the remaining chapters …
```

When the above abstract container is mapped to a File System Container, the directory structure within the file system exactly matches the UCF's Abstract Container directory structure shown above:

```
File System Container:
…some directory within the file system…/
mimetype
META-INF/
    container.xml
    [manifest.xml]
    [metadata.xml]
    [signatures.xml]
    [encryption.xml]
    [rights.xml]
OEBPS/
   Great Expectations.opf
    cover.html
    chapters/
   chapter01.html
   chapter02.html
   … other HTML files for the remaining chapters …
```

When the above Abstract Container is stored within a Zip container, the contents of the Zip archive will match the directory structure shown above. When the Zip archive is created, the mimetype file must be the first file in the archive. [See "Container media type identification " on page 229.]

```
Zip Container:
mimetype
META-INF/
   container.xml
   [manifest.xml]
   [metadata.xml]
   [signatures.xml]
   [encryption.xml]
   [rights.xml]
OEBPS/
   Great Expectations.opf
   cover.html
   chapters/
   chapter01.html
   chapter02.html
   … other HTML files for the remaining chapters …
```

The mimetype file contains the string "application/epub+zip."

The corresponding META-INF/container.xml file might appear as follows:

```
<?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>
```

Note: N.B. The use of the specific namespace string "urn:oasis:names:tc:opendocument:xmlns:container" should be considered provisional until approved by an OASIS technical committee.

Example of a Mars file

A Mars file, unlike an OEBPS file, does not require container.xml to include a rootfiles element. The root file is always backbone.xml. The example file does not include digital signatures, encrypted data, or DRM restrictions and therefore does not include signatures.xml, encryption.xml, or rights.xml.

```
Abstract Container:
mimetype
META-INF/
    container.xml
   [manifest.xml]
   [metadata.xml]
backbone.xml
page/0/
   info.xml
```

```
  pg.svg
  im_1.png
  f_1.fd
fonts/
  f_1._sfnt
```

The corresponding META-INF/container.xml file might appear as follows:

```
<?xml version="1.0"?>
<container version="1.0"
  xmlns="urn:oasis:names:tc:opendocument:xmlns:container">
  <relationships xmlns:pdf="http://ns.adobe.com/pdf/2006">
  <relationship type="metadata" target="$path.xmp"/>
  <relationship type="pdf:annotation" target="$path.ann"/>
  </relationships>
</container>
```

A Mars container.xml file includes relationship elements for metadata and annotations.

UCF Container Contents

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 "Zip Container" on page 233. 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 of this Specification.

Relative IRIs for referencing other components

Files within the Abstract Container reference 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" references 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:

```
<img src="image1.jpg" alt="…" …/>
```

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 "Example of an OEBPS file" on page 225, the IRI "/OEBPS/cover.html" will refer to the file OEBPS/cover.html no matter what file the IRI is found in.

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. This 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 referenced by URI from within the content.

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 “Zip Container” on page 233 for more detail on this mechanism.

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.

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 "Relationships (Optional)" on page 231.

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 "RELAX NG UCF Schema" on page 234.

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.

This example is informative.

The following example shows a sample container.xml for an UCF container inside of which is an OEBPS Publication with the root file "OEBPS/My%20Crazy%20Life.opf" (the OEBPS package file):

```
<?xml version="1.0"?>
<container version="1.0"
   xmlns="urn:oasis:names:tc:opendocument:xmlns:container">
   <rootfiles>
      <rootfile full-path="OEBPS/My%20Crazy%20Life.opf"
       media-type="application/oebps-package+xml" />
   </rootfiles>
</container>
```

This example is informative.

The following example adds an alternate PDF version of the Publication:

```
<?xml version="1.0"?>
<container version="1.0"
   xmlns="urn:oasis:names:tc:opendocument:xmlns:container">
   <rootfiles>
      <rootfile full-path="OEBPS/My%20Crazy%20Life.opf"
       media-type="application/oebps-package+xml" />
      <rootfile full-path="PDF/My%20Crazy%20Life.pdf"
       media-type="application/pdf" />
   </rootfiles>
</container>
```

This is normative.

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.

This example is informative.

For example, the following container.xml file is conformant.

```
<?xml version="1.0"?>
<container version="1.0"
  xmlns="urn:oasis:names:tc:opendocument:xmlns:container"
  foo:xmlns=" … "
  foozle:xmlns=" … " />
  <foo:bar />
  <rootfiles foozle:identifier="bar">
  …
  </rootfiles>
</container>
```

But the following container.xml file is non-conformant due to the non-namespace-qualified use of the <foo> element.

```
<?xml version="1.0"?>
<container version="1.0"
  xmlns="urn:oasis:names:tc:opendocument:xmlns:container">
  <foo />
  <rootfiles>
   …
  </rootfiles>
</container>
```

Likewise, the following container.xml file is also non-conformant due to the non-namespace-qualified use of the "identifier" attribute on the <rootfiles> element.

```
<?xml version="1.0"?>
<container version="1.0"
  xmlns="urn:oasis:names:tc:opendocument:xmlns:container">
  <rootfiles identifier="bar">
   …
  </rootfiles>
</container>
```

Relationships (Optional)

The container.xml file MAY include information that identifies implicit relationships between files in a container. Usually, one file explicitly references 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`	Full path to the resource	/a/b/c.d
`dir`	Directory (without the filename)	/a/b
`filename`	Path without the directory	c.d
`basename`	Filename without the extension	c
`ext`	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.

For example:

```
<?xml version="1.0"?>
<container version="1.0"
  xmlns="urn:oasis:names:tc:opendocument:xmlns:container">
  <relationships xmlns:pdf="http://ns.adobe.com/pdf/2006">
    <relationship type="metadata" target="$path.xmp"/>
    <relationship type="pdf:annotation" target="$path.ann"/>
  </relationships>
</container>
```

The first relationship specifies that any file may have associated metadata. This metadata is found in a file that has the same name as the original with ".xmp" added as an extension. For example, /page/0/pg.svg may have metadata in /page/0/pg.svg.xmp. The second relationship specifies that any file may have associated annotations. The annotations are found in a file that has ".ann" added as an 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 “Digital Signatures” on page 240 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 “Encryption” on page 242 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 this 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.

Zip Container

UCF’s Zip Container supports the Zip format as specified by the application note at http://www.pkware.com/business_and_developers/developer/appnote/, 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 “Encryption” on page 242. 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

RELAX NG UCF Schema

Here is a RELAX NG schema that defines the grammar of a `container` element.

```
<?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>

</choice>
```

Example

The following example demonstrates the use of the 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>j6lwx3rvEPO0vKtMup4NbeVu8nk=</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>

  <!-- 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: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>
   ...
    <img src="images/cover.png"
      alt="Cover image: a picture of the Bard of Avon" />
    ...
  </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.

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 defined by this specification 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, this 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. For example, Mars always uses backbone.xml. This eliminates the need to find, read, and process an extra file when opening a Mars file, which can have significant cost when viewing Mars files on the Web.
- 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. Mars (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.

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 signatures.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 "RELAX NG UCF Schema" on page 234.

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 referenced 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 Syntax and Processing") 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>j6lwx3rvEPO0vKtMup4NbeVu8nk=</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>
```

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 “RELAX NG UCF Schema” on page 234.

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>
```

Index