Adobe FrameMaker 7.2 Application Pack for S1000D

TABLE OF CONTENTS

The Adobe® FrameMaker® 7.2 Application Pack for S1000D™ is a collection of XML applications that show how FrameMaker 7.2 can handle complex, industry standard XML documentation requirements. The Application Pack for S1000D uses the new XML schema and XSLT integration built into FrameMaker 7.2, which provides great flexibility for working with different document types.

The Adobe FrameMaker 7.2 Application Pack for S1000D is aimed at technical authors who create or edit S1000D content, and system developers who need to extend the Application Pack for S1000D to meet more advanced S1000D project requirements.

An introduction to S1000D

ASD S1000D is an international standard. Although it was first used by the European military aerospace industry, it has now been adopted by countries and industries around the world.

The S1000D specification is unusual because it specifies the requirements for a project's complete publication lifecycle. All stages of the publication lifecycle are accounted for, including initial project planning, business rules, management methods, configuration, writing rules, illustration rules, production methods, quality assurance, data storage, publishing, commenting, and revision cycles.

At the heart of any S1000D project you will find the Common Source DataBase, or CSDB. At its most basic, the CSDB is a repository for all of the components of a publication. It's more likely, however, that the CSDB will be used to manage the entire project, including:

- Project set-up
- Production workflow
- Quality assurance processes
- Lifecycle management and revision control
- Content storage for data modules and graphics
- Publication management for IETP or IETM and paper documents.

Modular publications

One of the S1000D specification's key features is the data module. A data module is an easy–to–manage document designed for reuse. A typical data module provides a small amount of content about a specific topic in a clearly defined context.

Data module types

S1000D is a standard for data interchange, and ensures that all data follows a common set of rules, thereby reducing the lifecycle costs for a project. S1000D defines a fixed set of data module types, which are:

DATA MODULE TYPE	PURPOSE
Business Rules Exchange	The BREX data module propagates the allowed customizations throughout a project. It is a configuration file that is not intended to be viewed by the end user. See Using BREX for more detail.
Crew	Provides structure for operational checklist procedures and related descriptive information. Typical use: aircrew flight reference cards and flight manuals.

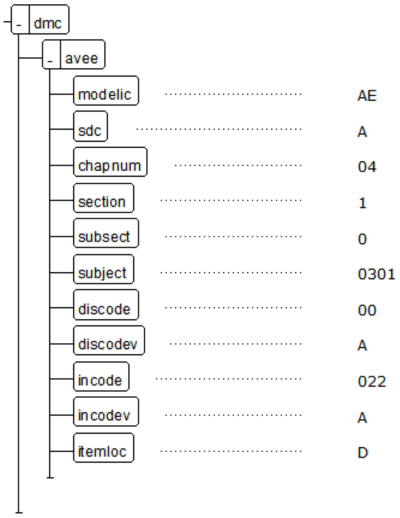
DATA MODULE TYPE	PURPOSE
Description	A general purpose data module for descriptive text. Also very useful for legacy data due to its flexible structure.
Fault	Dedicated structure for Fault Isolation procedures that may be represented as graphical fault isolation charts by some display systems.
Illustrated Parts Data	The basic building block for an Illustrated Parts Catalogue where each data module includes a labeled parts illustration with a corresponding parts list. These data modules are often generated by an ASD 2000M–compliant initial provisioning database system.
Procedural	Step-based procedural data. Includes Preliminary requirements, Maintenance function, and Close-up procedures.
Process	The Process data module is used by integrated systems to dynamically deliver maintenance information that is dependant on the status of the equipment. The majority of the content of a Process data module is a type of scripting language for data module delivery.
Schedule	A data module type that is used for maintenance planning information.
Wiring Data	The Wiring data module is used to store wiring data information including wire, harness, equipment, and standard parts data. This data could then be used to generate wiring and schematic diagrams.
Wiring Data Description	Each relevant element in a wiring data module requires a project–specific wiring element field description. This data module type, therefore, provides an interface between a company's wiring data production system and the wiring data module.

Managing data modules

A typical S1000D project makes use of thousands of data modules and graphics, or CSDB objects. Many of these CSDB objects are specific to the project, but some will be reused from other projects. So how do you manage all of this potentially fragmented, modular information?

S1000D's answer is to provide every CSDB object with a unique, but meaningful, identification code. For a data module this is called the Data Module Code, or DMC. (For example, the code for an illustration is the Illustration Control Number, or ICN.

The data module code has two forms AVEE and the deprecated AGE. The structure of a DMC and its meaning are shown below.



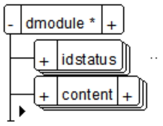
The data module code structure

DMC ELEMENT	DESCRIPTION
modelic - Model Identification Code	A globally unique code for each project. These codes are allocated by NAMSA. An up-to-date list of allocated codes can be found at: www.namsa.nato.int/s2000m/s2000m_moi_e.htm and www.namsa.nato.int/s2000m/s2000m_moi14_e.htm
sdc - System Difference Code	Identifies alternative systems that share identical SNS codes. This happens when, for example, subsystems supplied by different manufacturers perform an identical role.
chapnum - SNS Chapter Number	First part of the Standard Numbering System. Provides system–level breakdown of a product's structure. The use of "chapter" does not imply a page based publication breakdown.
section and subsect - SNS Section Number	Second part of the Standard Numbering System. Provides subsystem level breakdown of a product's structure.
subject - SNS Subject Number	Third part of the Standard Numbering System. Provides sub–subsystem level breakdown of a product's structure.
discode - Disassembly Code	For most types of data module, this element extends the possible system breakdown level. For the IPD data module, it holds the Figure number for the current SNS breakdown.
discodev - Disassembly Code Variant	Introduces a variant of the Disassembly code or IPD figure.
incode - Information Code	Identifies the purpose of the data module using a three digit code. Some examples are: • 041—Description of how it is made • 520—Remove procedures • 941—Illustrated Parts Data
incodev - Information Code Variant	Typically a way to define an alternative procedure that achieves the same results for a given incode.
itemloc - Item Location Code	Where the procedure is done (for example, "on the main equipment," or "on the workbench").

The DMC is used as the data module's identifier for linking between data modules using the "refdm" element. The DMC is also used as the basis for the data module's file name (for example, DMC-AE-A-04-10-0301-00A-022A-A_001-02.XML.

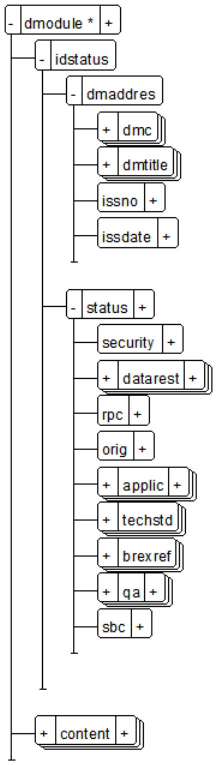
The Illustration Control Number is used as the entity name in a data module and the file name (for example, ICN-AE-A-321005-0-U8025-00503-A-02-1.CGM). See the S1000D specification, chapter 4.4, for more information about the ICN.

The structure of a data module



All data modules contain two main sections, the idstatus and the content.

The idstatus, or Identification and Status section, contains a comprehensive set of metadata elements that are used by the CSDB to manage the data module. The idstatus is not normally displayed to the user of the publication, and is often not editable by the author. The structure view below shows the idstatus elements. See the S1000D specification, chapter 3.9.5.1., for a complete explanation of the idstatus section.



The structure of the idstatus section

While the available element structure of the idstatus is identical for all data modules, the content section varies for each type of data module.

The content section is the part of the data module that the publication's user sees in the IETP or printed manual.

An S1000D project

Creating and managing a modular publication is unlike working on a traditional document, but the specification has guidance here as well.

Before a single data module is written or a legacy page converted, some important steps must be taken:

1. Define the project, and obtain a Model Identification Code.
2. Select the relevant Standard Numbering System for the equipment type, (for example, General surface vehicle, Navigation system, or even your project's own unique SNS).
3. Create the Data Module Requirements List. A DMRL is a list of all data modules that will be needed for the project. This stage should not be underestimated; it is a significant amount of work. However it is a vital part of the project setup. Once complete, the DMRL will aid resource planning and project costing. It will identify potential reuse cases. It also makes it possible to define refdm element links to data modules that do not yet exist.
4. If the project involves legacy conversion, the Illustration Control Numbers should be created while compiling the DMRL.
5. Create the project business rules and the BREX data module.

The publication module

S1000D gives instructions for page based and electronic publication. Both methods are controlled by the publication module, a structured list of references to data modules.

Further information

This introduction can only give an overview of the S1000D specification. For more information, see the S1000D website at www.s1000d.org, where the complete specification is available for download in Adobe PDF format. You will also find most issues of the DTDs and XML schemas if you want to extend the application pack to support additional data module types or previous issues.

What's included with the Application Pack for S1000D

The Adobe FrameMaker 7.2 Application Pack for S1000D includes four schema–based XML applications:

- Business Rules Exchange
- Description
- Illustrated Parts Data
- Procedural

All applications include the issue 2.2.1 schemas.

It is Adobe's intention to support S1000D issue 2.3 when it is published in late 2006. This new issue of the specification will include the ATA requirements for civil aircraft maintenance documentation as well as other enhancements.

Components

The Application Pack's components are described in the following table.

ITEM	DESCRIPTION
attr221.00.mif attr221.at.mif attr221.cc.mif attr221.cm.mif attr221.co.mif attr221.cp.mif attr221.cs.mif attr221.cv.mif attr221.dt.mif attr221.em.mif attr221.lt.mif attr221.pf.mif attr221.rt.mif attr221.sk.mif attr221.sl.mif attr221.th.mif	Generated MIF fragments for BREX attribute configuration
attr221.exp.xsl	Attribute value sets for postprocessing
attr221.imp.xsl	Attribute value sets for preprocessing
brex221.edd.fm	The element definition document for the Business Rules EXchange data module application. Minimal formatting provided because a BREX module is not designed for printed output.
brex221.exp.xsl	XSLT postprocessing transform that reinstates the original attribute values from the referenced BREX data module. See Using BREX for more information.
brex221.fmx.dtd	This XML DTD is initially created by FrameMaker when importing the BREX XSD schema and is used by FrameMaker for validation during XML import. It also defines the structure of the BREX EDD.
brex221.imp.xsl	XSLT preprocessing transform that merges configuration data from the referenced BREX data module to redefine various attribute values. It also writes out the associated MIF fragments. See Using BREX for more information.
brex221.rwr.fm	Read/write rules for the BREX application
brex221.tpl.fm	A normal structured FrameMaker template for the BREX application
cals221.exp.xsl	XSLT postprocessing transform that reinstates the S1000D/CALS table structure on export

ITEM	DESCRIPTION
cals221.imp.xsl	XSLT preprocessing transform that restuctures the CALS tables so that FrameMaker's table titles can be used
copy221.gen.xsl	XSLT identity transform stylesheet that is used to pass through XML content without changes
desc221.edd.fm	The element definition document for the Descriptive data module application. The formatting follows the requirements of S1000D chapter 6.2.2 as closely as possible.
desc221.eddAPL.fm	Alphabetical list of elements from the description EDD. This is an aid to development and is not used as part of the application.
desc221.exp.xsl	XSLT transform for Descriptive data module postprocessing. References the following stylesheets using xsl:import or xsl:include: • copy221.gen.xsl • attr221.exp.xsl • brex221.exp.xsl • rdfm221.exp.xsl • figs221.exp.xsl • refs221.exp.xsl • xref221.exp.xsl • cals221.exp.xsl
desc221.fmx.dtd	This XML DTD is initially created by FrameMaker when importing the Description XSD schema, and is used by FrameMaker for validation during XML import. It also defines the structure of the Descriptive data module EDD.
desc221.imp.xsl	XSLT transform for Descriptive data module preprocessing. References the following stylesheets using xsl:import or xsl:include: • copy221.gen.xsl • attr221.imp.xsl • figs221.imp.xsl • brex221.imp.xsl • refs221.imp.xsl • tocs221.imp.xsl • frpi221.imp.xsl • cals221.imp.xsl • xref221.imp.xsl
desc221.rwr.fm	Read/write rules for the Descriptive data module application
desc221.tpl.mif	A special structured template in MIF format. This template references MIF fragment files that are dynamically updated each time FrameMaker opens a data module.
figs221.exp.xsl	XSLT postprocessing transform that reinstates the S1000D structure for the <figure> element. See XSLT stylesheets for more information.
figs221.imp.xsl	XSLT preprocessing transform that builds a FrameMaker compatible structure for the <figure> element. See XSLT stylesheets for more information.
frpi221.exp.xsl	XSLT postprocessing transform that outputs processing instructions as required by the project configuration
frpi221.imp.xsl	This XSLT preprocessing transform handles foreign application's processing instructions. It is designed to be extended as required. In its basic configuration it will transform <?pub _newline?> processing instructions into <newline> elements in FrameMaker.
ipdm221.edd.fm	The element definition document for the Illustrated parts data module application. The formatting follows the requirements of S1000D chapters 6.2.2 and 5.3.1.4 exactly.
ipdm221.eddAPL.fm	Alphabetical list of elements from the IPD EDD. This is a development aid, and is not used as part of the application.

ITEM	DESCRIPTION
ipdm221.exp.xsl	XSLT transform for Procedural data module postprocessing. References the following stylesheets using xsl:import or xsl:include: • copy221.gen.xsl • attr221.exp.xsl • brex221.exp.xsl • rdfm221.exp.xsl • figs221.exp.xsl • xref221.exp.xsl
ipdm221.fmx.dtd	This XML DTD is initially created by FrameMaker when importing the IPD XSD schema, and is used by FrameMaker for validation during XML import. It also defines the structure of the IPD EDD.
ipdm221.imp.xsl	XSLT transform for Procedural data module preprocessing. References the following stylesheets using xsl:include: • copy221.gen.xsl • figs221.imp.xsl • brex221.imp.xsl • attr221.imp.xsl
ipdm221.rwr.fm	Read/write rules for the Illustrated parts data module application
ipdm221.tpl.mif	A special structured template in MIF format. This template references MIF fragment files that are dynamically updated each time FrameMaker opens a data module.
prel221.exp.xsl	XSLT postprocessing transform that re-creates the Schema defined structure for each Preliminary requirements section
prel221.imp.xsl	XSLT preprocessing transform that constructs a table structure for each Preliminary requirements section.
prod221.edd.fm	The element definition document for the Procedural data module application. The formatting follows the requirements of S1000D chapter 6.2.2 as closely as possible although the specification is a little ambiguous in places.
prod221.exp.xsl	XSLT transform for Procedural data module postprocessing. It references the following stylesheets using xsl:include: • copy221.gen.xsl • attr221.exp.xsl • brex221.exp.xsl • rdfm221.exp.xsl • figs221.exp.xsl • refs221.exp.xsl • prel221.exp.xsl • xref221.exp.xsl • cals221.exp.xsl
prod221.fmx.dtd	This XML DTD is initially created by FrameMaker when importing the Procedural XSD schema, and is used by FrameMaker for validation during XML import. It also defines the structure of the Procedural EDD.
prod221.imp.xsl	XSLT transform for Procedural data module preprocessing. It references the following stylesheets using xsl:include: • copy221.gen.xsl • attr221.imp.xsl • figs221.imp.xsl • brex221.imp.xsl • refs221.imp.xsl • tocs221.imp.xsl • prel221.imp.xsl • cals221.imp.xsl • xref221.imp.xsl
prod221.rwr.fm	Read/write rules for the Procedural data module application
prod221.tpl.mif	A special structured template in MIF format. This template references MIF fragment files that are dynamically updated each time FrameMaker opens a data module.

ITEM	DESCRIPTION
rdfm221.exp.xsl	XSLT postprocessing transform that builds the rdf:Description metadata structure using current values from the <idstatus>
refs221.exp.xsl	XSLT postprocessing transform that builds the <refs> list of <refdm> and <reftp> elements. The list is built in document order with duplicates removed.
refs221.imp.xsl	XSLT preprocessing transform that builds the <refs> list of <refdm> and <reftp> elements. The list is built in document order with duplicates removed. Table structure is added for formatting purposes.
rule221.rwr.fm	This generic read/write rules file is included by each S1000D application's main read/write rules file
UsingS1000DAppPack.pdf	This technical paper
stuctapps.s1000d.fm	This structapps fragment file is used when installing the Adobe FrameMaker 7.2 Application Pack for S1000D.
tocs221.imp.xsl	XSLT preprocessing transform that builds the table of contents, list of tables and list of figures. Each TOC entry is an active hypertext link.
xref221.exp.xsl	This XSLT postprocessing transform returns the FrameMaker <xref> structure to the S1000D standard structure
xref221.imp.xsl	XSLT preprocessing transform that wraps the FrameMaker cross reference object element so that <xref> content can be handled

The Application Pack for S1000D also includes the complete set of S1000D XML modular schema components, which are also available from www.s1000d.org along with other XML and SGML resources.

Installation

To install the Adobe FrameMaker 7.2 Application Pack for S1000D:

Windows

1 Unzip the file S1000D221.zip into your chosen location. (For example, C:\Program Files\Adobe\S1000D).

2 Move the S1000D_dm folder from the S1000D folder into C:\Program Files\Adobe\FrameMaker7.2\Templates. This folder contains the XML templates for new data modules.

3 In FrameMaker 7.2, open the Structured Application Definition Document by choosing File > Structure Tools > Edit Application Definitions.

4 Open the file stuctapps.s1000d.fm in theApplication Pack for S1000D folder.

5 Copy the four XML applications and paste them into the Structured Application Definition Document.

6 Double-click the schema path in the S1000D2.21BREX application. The Variable dialog box opens.

7 Edit the variable "S1000D.Path" so that its definition matches the chosen installation path (for example, C:\\Program Files\\Adobe\\S1000D\\).

8 Choose File > Structure Tools > Read Application Definitions.

9 Save the Structured Application Definition Document. The installation is complete.

What you can do with the Adobe FrameMaker 7.2 Application Pack for S1000D

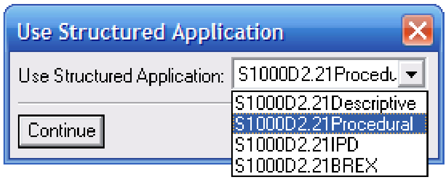
Depending upon the complexity and business rules of your S1000D project, you may be able to use the Application Pack for S1000D unchanged for all data modules. A more likely scenario is that you'll use the Adobe FrameMaker 7.2 Application Pack for S1000D as the basis for further developments.

Perhaps you need to create alternative formatting for the existing data module types. Or, you may need to add a currently unsupported data module type.

The following sections introduce the features of the Application Pack for S1000D and give examples of how to use them.

Getting started

The Adobe FrameMaker 7.2 Application Pack for S1000D includes a small set of example data modules in the Examples folder. Because all data modules share the same root element name of <dmodule>, you must choose the correct structured application.



Use Structured Application: S1000D2.21Procedu

Choose a Structured Application

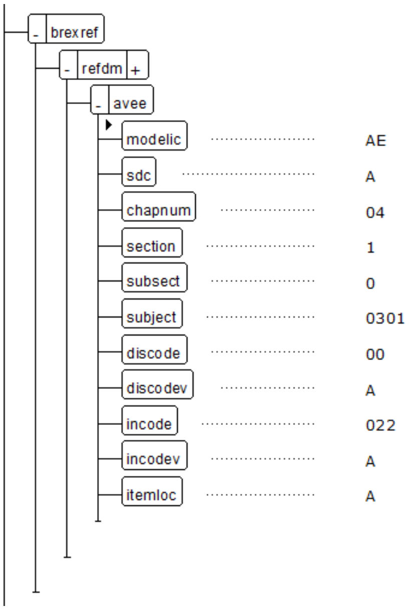
The data module type may be indicated by the DMC information code, (see incode - Information Code. You'll find a full list in chapter 8.4 of the S1000D specification. Some examples are:

- 040, 041, 042—use S1000D2.21Descriptive
- 121, 151, 330, 663—use S1000D2.21Procedural
- 022—use S1000D2.21BREX
- 941—use S1000D2.21IPD

Using BREX

Business Rules Exchange (BREX), the S1000D method for controlling project configuration or customization, is fully integrated with the Description, Procedural, and IPD applications.

Each data module includes a <brexref> element, which is a reference to a BREX data module that includes project specific settings. This data module code is used as part of the file name by the XSLT transform stylesheet when it merges the information into each data module. In the screenshot on the next page, the refdm maps to a data module with the filename "DMC-AE-04-10-0301-00-A-022A-A.xml." The path to the BREX data module is defined for each XML application in the Structured Application Definition Document.



modelic	AE
sdc	A
chapnum	04
section	1
subsect	0
subject	0301
discode	00
discodev	A
incode	022
incodev	A
itemloc	A

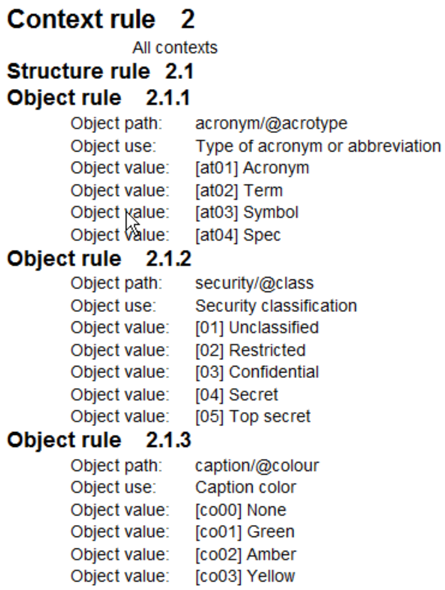
The <brexref> structure

Setting up the BREX file

In this exercise, you will make a small change to the BREX data module then, observe the effect it has on a data module that references it.

1 In FrameMaker, open the supplied BREX data module <$S1000Dpath>/Examples/DMC-S1000DBIKE-AAA-00-00-00-0000-022A-D_001-00.xml.

2 In Structure rule 2.1, locate Object rule 2.1.2. This rule lists the security settings that the project will use.



Context rule 2

All contexts

Structure rule 2.1

Object rule 2.1.1

Object path: acronym/@acrotype
Object use: Type of acronym or abbreviation
Object value: [at01] Acronym
Object value: [at02] Term
Object value: [at03] Symbol
Object value: [at04] Spec

Object rule 2.1.2

Object path: security/@class
Object use: Security classification
Object value: [01] Unclassified
Object value: [02] Restricted
Object value: [03] Confidential
Object value: [04] Secret
Object value: [05] Top secret

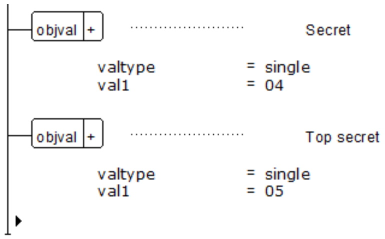
Object rule 2.1.3

Object path: caption/@colour
Object use: Caption color
Object value: [co00] None
Object value: [co01] Green
Object value: [co02] Amber
Object value: [co03] Yellow

Context rules in the BREX data module

Note: *The first element in each object rule is the object path which uses an Xpath expression to identify the context for the rule. In this case 'security/@class' means that the object rule relates to the class attribute on any security elements.*

3 Add a new <objval> element after [05] Top secret.



objval Secret
valtype = single
val1 = 04

objval Top secret
valtype = single
val1 = 05

<objval> attributes in the structure view

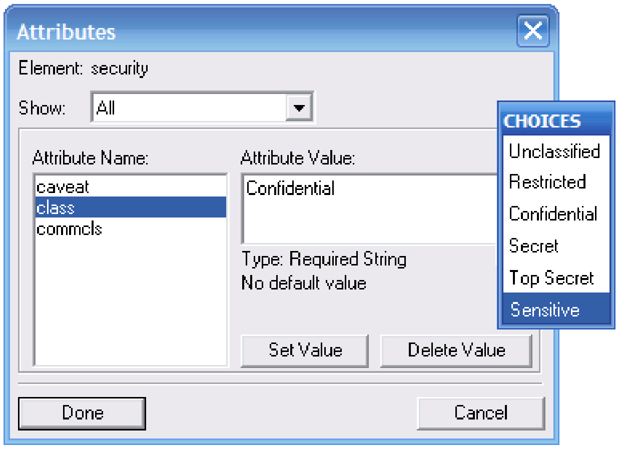
4 When prompted for attribute values, set valtype to single and val1 to 51. Click Insert Element, and then type in the meaning for the new <objval>, for example, Sensitive.

Note: *The Val1 attribute stores the value that is saved to XML in every data module. Without BREX integration the security level shown in the header and footer would be a meaningless number. Also the job of editing a data module would be much more difficult if you have to remember that em64 is an emphasis setting of red text in a monospaced font. The value 51 is chosen because configurable attributes may have values from 00 to 99 where 00 to 50 are reserved for S1000D specification definition.*

5 Save and close the BREX data module.

6 Open any data module from the examples folder. Find and select the <security> element.

7 Open the Attributes dialog box, then select class. Click the Choices button to confirm that your new security setting is available as an option.



Element: security

Show: All

Attribute Name: class

Attribute Value: Confidential

Type: Required String
No default value

BREX integration in the Attributes dialog

How BREX integration works

When a data module is opened, the XSLT transform uses the xalan:write extension to output a MIF fragment that contains the new attribute choices. Each structured template is a MIF file that uses several include statements to pull in the generated MIF fragments.

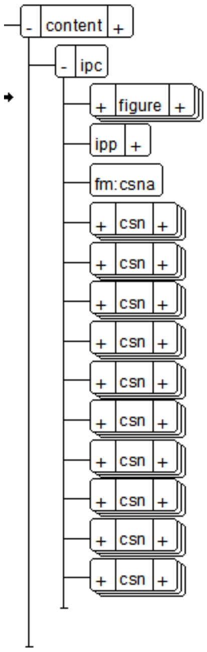
Manual IPD compilation

The IPD data module was designed to be generated as output from a 2000M provisioning database. In reality, the IPD is often compiled manually. With the FrameMaker Application Pack for S1000D, you can work either way. The formatting follows the S1000D requirements exacly, while the transformed structure makes sense of a very confusing source schema structure.

This section explains how the Application Pack for S1000D helps users avoid the hidden complexities of IPD compilation.

IPD content structure

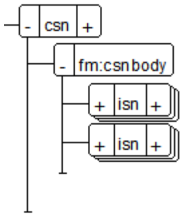
The IPD content section is arranged as shown below.



IPD content structure

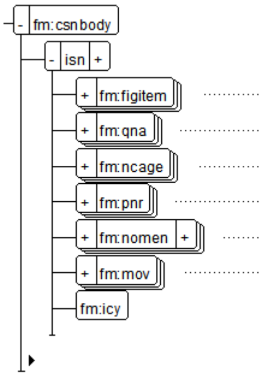
The <ipc> element contains the following descendants:

- A single **<figure>** element that may include a single graphic or multiple sheets. See [Figure transformation]() for more information.
- A single **<ipp>** element which stores information about the Initial Provisioning Project.
- A single **<fm:csna>** element. This is simply an anchor element for the following <csn> table structures. Its purpose is to ensure that the parts list starts at the top of the page following the graphic.
- One or more **<csn>** elements. The Catalogue Sequence Number is the parent element of each line entry in the parts list. Each <csn> contains the elements shown below.



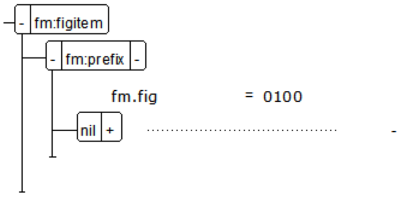
<csn> structure

- Each <csn> is a FrameMaker table element, the child **<fm:csnbody>** is a table body element.
- Each **<isn>** (or Item Sequence Number) is a FrameMaker table row element. There must be at least one <isn> for each <fm:csnbody>. The reason for having more than one <isn> is to show variant parts data or, alternative parts that perform the same function.
- Within the <isn>, seven generated FrameMaker table cell elements contain the parts data elements.



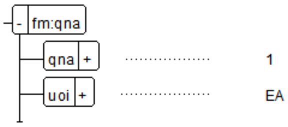
<fm:csnbody> structure

- **<fm:figitem>** Figure number and item number cell. Contains the generated **<fm:prefix>** element, which displays the figure number at the top of the parts list as well as the item number from the <csn> element. When an item is not illustrated, add a <nil> element as a child of <fm:prefix>. Enter a single "-" as its content.



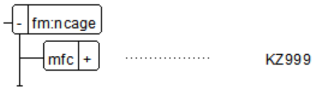
<fm:figitem> structure

- **<fm:qna>** Quantity per next higher assembly cell. Must contain a single **<qna>** element that gives the quantity of this item in its parent assembly. For the first or top level item this is normally "Ref." Zero or more **<uoi>** elements are used to define the Unit of Issue, (for example, EA for "each," mm for millimetres). Allowable codes are defined in S2000M.



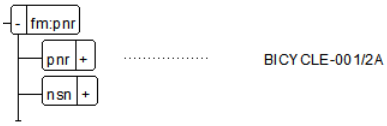
<fm:qna> structure

- **<fm:ncage>** NATO Commercial And Government Entity cell. Must contain a single **<mfc>** element. The terms NCAGE and MFC (Manufacturer's Code) are interchangeable. This code is unique to the manufacturer of each part.



<fm:ncage> structure

- **<fm:pnr>** Part Number cell. A single mandatory **<pnr>** element holds the item's part number. An optional **<nsn>** element is used if the NATO Stock Number is required.



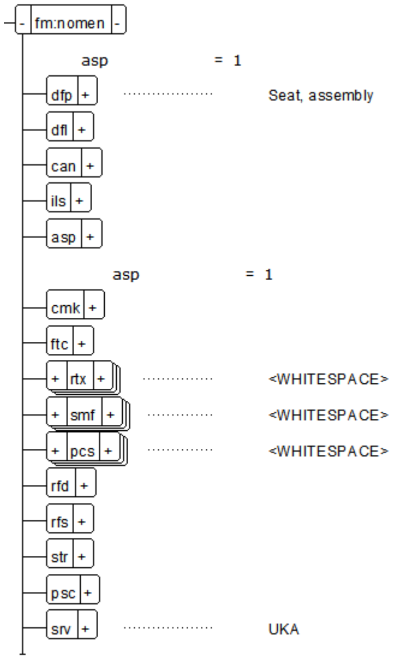
<fm:pnr> structure

- **<fm:nomen>** Nomenclature cell. Holds all of the descriptive elements for an item. The relationship between parts is indicated by the indenture level as defined by the ind attribute on the ancestor <csn> element. Indenture is shown by the number of bullets preceding the <dfp> element.

The asp attribute is used for formatting purposes. Its value is copied from the optional **<asp>** element during import. When asp =”1”, the preceding bullets are replaced with the same number of asterisks, which shows that this is an attaching part. You may change this value manually if required, but it will always use the values from the <asp> element when imported from XML.

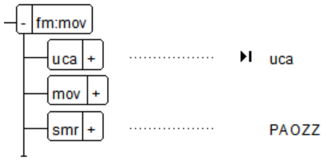
- **<dfp>** or Description For Part element is mandatory and must use the unique name that is allocated to the part. Because of a potentially confusing section of the underlying IPD schema structure, it is possible to use more than one <dfp> element. However this is not recommended as it will have unwanted side effects when saving XML.
- **<dfl>** – Description For Location is optional, it is used for a context specific description of the item.
- **<asp>** – Attaching, Storage, or Shipping Part. The asp attribute may have the value 1, 2, or 3. The value is copied to the asp attribute on the <fm:nomen> element on import from XML.

For descriptions for other elements, see chapter 5.3.1.4 of the S1000D specification.



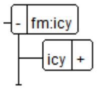
<fm:nomen> structure

- **<fm:mov>** Model Version, Usable on Code cell. Contains optional elements **<uca>** (Usable on Code for the Assembly) or **<uce>** (Usable on Code for the Equipment). The optional, repeatable **<mov>** element displays the Model version information for the current item. The **<smr>** (Source, Maintenance, and Recoverability) element is required and repeatable. See the S2000M specification for an explanation of its use.



<fm:mov> structure

- **<fm:icy>** Interchangeability cell, which can contain a single **<icy>** element. See the S2000M specification for an explanation of its use.



<fm:icy> structure

Description and Procedural data modules

Description and Procedural data module types are more conventional in their structure. The best way to learn how to use them is to open any of the provided examples and see what can be done by trial and error. The S1000D specification provides guidance for the usage of all elements for these modules.

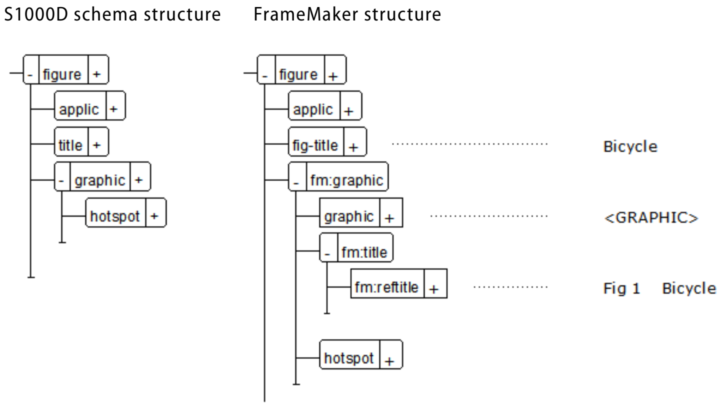
fm:display options

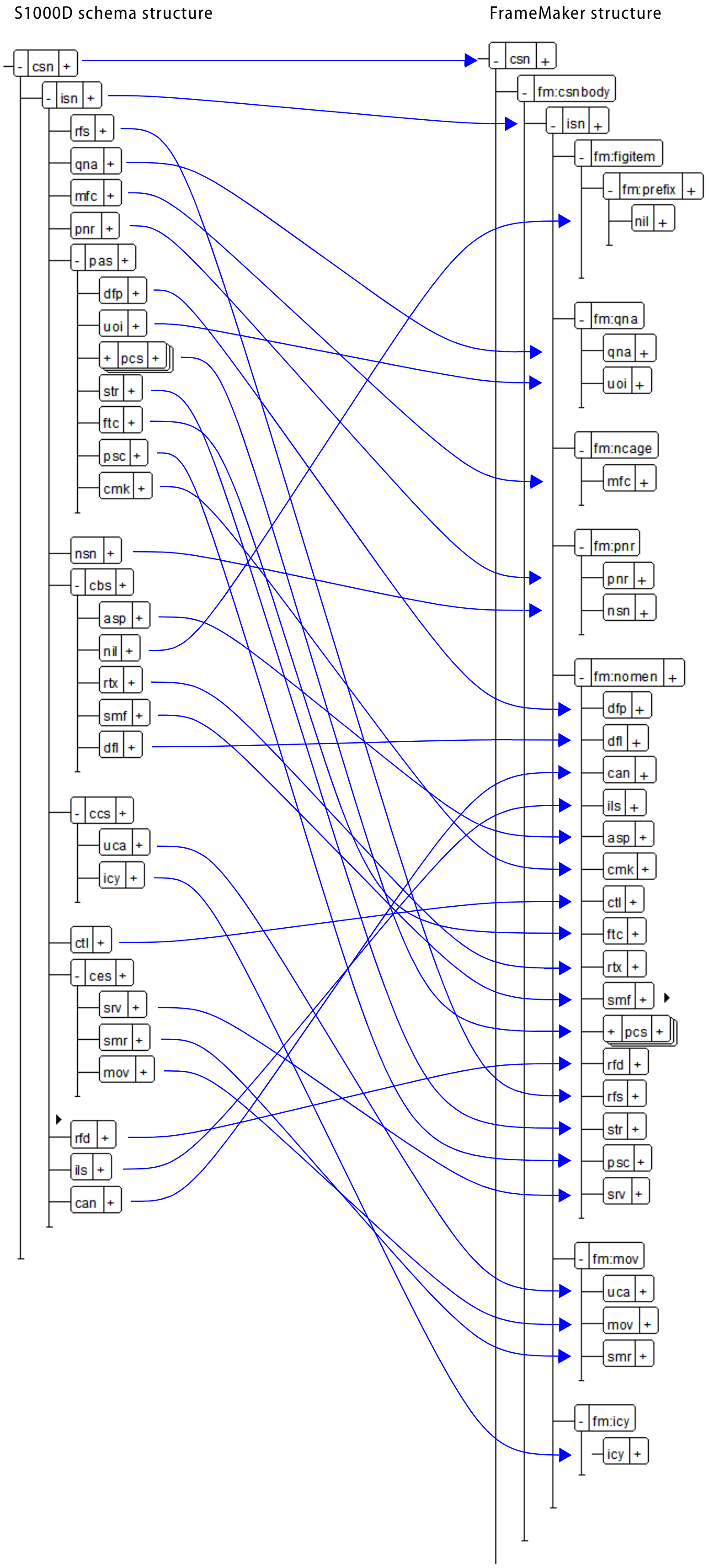
All four S1000D applications may be configured to display for edit or publish. The top level <dmodule> element includes the <fm:display> attribute, which has values of either "edit" or "publish." The attribute's setting affects the way some elements are displayed. In Edit mode, the <idstatus> and <fig-title> are visible. In Publish mode, these items are hidden. Elements are hidden by their formatting, rather than by using conditional text, so that the data module remains valid regardless of the <fm:display> setting.

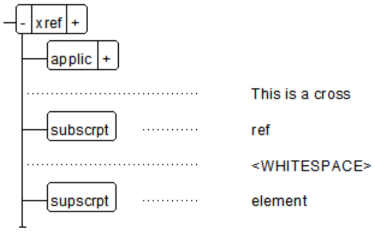
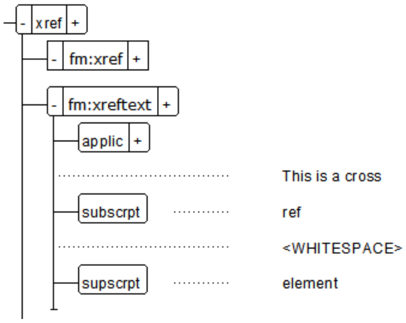
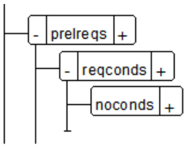
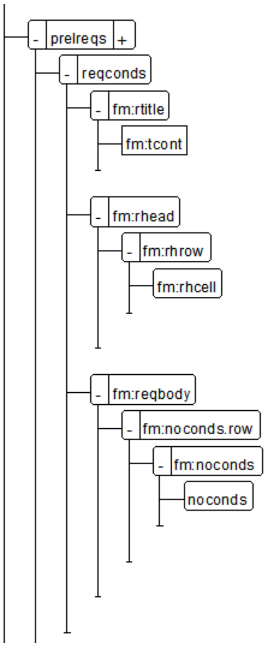
You can choose whether the data module opens in Edit or Publish mode by setting the value of the "start.mode" stylesheet parameter to "edit" or "publish" as required. The stylesheet parameters are defined in the Structured Application Definition Document.

XSLT stylesheets

The XSLT stylesheets for import and export were designed to enhance the use of structure in FrameMaker. The following table describes the main features of the XSLT processing.

XSLT FEATURE	DESCRIPTION
Figure transformation	The figure transformation performs several functions: • Moves child <hotspot> elements from the <graphic> object element. • Adds the <fm:graphic> element as a container for <graphic>, <fm:title>, <hotspot> and <rfa> elements. • Creates new <fm:title> and child <fm:reftitle> elements below each graphic that cross-reference to the renamed <fig-title> element above the graphic. • The figure's <title> element has been renamed to <fig-title> because an id attribute has been added. Do not make any cross references to this element because the id is dropped on export to XML. • All transformations are reversed for export to XML.  S1000D schema structure FrameMaker structure figure applic title graphic hotspot figure applic fig-title fm:graphic graphic fm:title fm:reftitle hotspot Bicycle <GRAPHIC> Fig 1 Bicycle • Writes out the graphic entities on export.
BREX handling	The BREX handling performs the following functions: Concatenates the file name of the current data module's referenced BREX file. • Merges the BREX file with the current data module and exchanges the source alphanumeric attribute values for the full names from the BREX. • Writes out a MIF file fragment for each attribute type to the location defined by the $mifpath parameter that's set in the Structured Application Definition Document. The XML application template is dynamically configured to match the BREX.

XSLT FEATURE	DESCRIPTION
IPD transformation	The IPD transform completely rearranges the IPD content structure as shown below.  S1000D schema structure FrameMaker structure csn isn rfs qna mfc pnr pas dfp uoi pcs str ftc psc cmk nsn cbs asp nil rtx smf dfl ccs uca icy ctl ces srv smr mov rfd ils can csn fm:csnbody isn fm:figitem fm:prefix nil fm:qna qna uoi fm:ncage mfc fm:pnr pnr nsn fm:nomen dfp dfl can ils asp cmk ctl ftc rtx smf pcs rfd rfs str psc srv fm:mov uca mov smr fm:icy icy

XSLT FEATURE	DESCRIPTION
Cross reference transformation	The cross reference transformation moves all child elements from the <xref> element. The existing xref element becomes a container for the optional <fm:xreftext> element, which now contains the source xref's text content (if any). The active FrameMaker cross reference object element is <fm:xref>. The complexity of the structure is because of the restrictions on mixed content for XML. S1000D schema structure  xref applic This is a cross subscrpt ref <WHITESPACE> supscrpt element FrameMaker structure  xref fm:xref fm:xreftext applic This is a cross subscrpt ref <WHITESPACE> supscrpt element
ToC generation	Generates the internal ToC entries on import for Descriptive and Procedural data modules.
Refs generation	Generates and sorts the reference data module list tables on import for Descriptive and Procedural data modules.
Preliminary requirements tables	In the procedural data module, the preliminary requirements information must be formatted as tables although the source XML does not conform to any existing table structure. **Required conditions** S1000D schema structure  FrameMaker structure  prelreqs reqconds fm:rtitle fm:tcont fm:rhead fm:rhrow fm:rhcell fm:reqbody fm:noconds.row fm:noconds noconds
rdf metadata generation	Generates the rdf:Description section on save to XML, ensuring that the rdf metadata is always correct when saved while removing unnecessary clutter from the FrameMaker document.

The fm namespace

The XSLT transformations often add structure to match FrameMaker's requirements. These added elements are created in the fm namespace to make them easy to identify.

The only place where this rule cannot be used is in some added attributes elements. Although FrameMaker supports namespaces in attribute names, it does not recognize the namespace prefix for some EDD formatting rules.

Extending the Application Pack for S1000D

The Adobe FrameMaker 7.2 Application Pack for S1000D has been designed with extensibility in mind. The XSLT stylesheets are modular so that you can choose which to use for your own S1000D applications. To create a new S1000D application, follow these steps (they assume the new application is for the Schedule data module):

1 Open the Structured Application Definition Document.

2 Copy the description template (.fm file, not MIF) and rename it to schd221.tpl.fm.

3 Copy the description read/write rules file and rename it to schd221.rwr.fm.

4 Add any new schedule specific rules to schd221.rwr.fm.

5 Copy the S1000D2.21Description XML application then rename the copy as required (for example S1000D221Schedule).

6 Alter the file name references to suit the data module type.

7 Open the XML schema for the data module type. A new EDD is created with an equivalent DTD. This DTD may be edited to create FrameMaker specific structures. Common stuctures such as <xref> and <tocs> can be copied from one of the existing DTDs. To ensure that the DTD and EDD have equivalent structures, always make the changes in the DTD then import the DTD into the EDD.

8 Create new primary XSLT stylesheets for import and export by copying, renaming, and editing desc221.imp.xsl and desc221.exp.xsl. Whenever you alter the DTD, you will probably need to create XSLT transformations for import and export. Whenever possible keep your new XSLT in separate files. These files must then be included in the primary stylesheet using xsl:include.

9 When you have fully tested the application, save the template as a MIF file. Open the MIF file in a text editor and insert each BREX MIF file include statement as shown below:

```
<EDAttrDef
    <EDAttrName `class'>
    <EDAttrRequired No>
    <EDAttrReadOnly No>
    <EDAttrHidden No>
    include (attr221.00.mif)
   > # end of EDAttrDef
```

Creating XSLT stylesheets for FrameMaker

FrameMaker's read/write rules share some abilities with XSLT. It, therefore, makes sense to avoid functional overlap. Limit the read/write rules for operations that cannot be done with XSLT, plus non-essential element renaming.

S1000D resources

The official resource for S1000D information is the S1000D website at www.s1000d.org. You can download all versions of the specification starting from issue 2.0 as Adobe PDF files. You can also download the DTDs, schemas, and example files for versions 1.7 and later.

Glossary

TERM	MEANING
ASD	The AeroSpace and Defence Industries Association of Europe. ASD represents the aeronautics, space, defence, and security industries in Europe.
ASD 2000M or S2000M	ASD Specification 2000M (S2000M) is a standard that specifies the information exchange requirements for materiel management functions that support international projects. S2000M is based on a business model agreed upon between military customers and industry suppliers.
IETP or IETM	Interactive Electronic Technical Publication (or Technical Manual). Sometimes qualified by a type such as IETP-X for XML based IETP.
MIF	Maker interchange Format. FrameMaker's text-based file format that allows exchange between different versions of FrameMaker.
NAMSA	NATO Maintenance and Supply Agency

**Adobe Systems Incorporated • 345 Park Avenue, San Jose, CA
95110-2704 USA • *www.adobe.com***