

Integrating DITA Specialization with ADOBE® FRAMEMAKER® 9



© 2009 Adobe Systems Incorporated. All rights reserved.

Adobe, the Adobe logo, and FrameMaker are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries. All other trademarks are the property of their respective owners.

This Work is licensed under the Creative Commons Attribution Non-Commercial 3.0 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/3.0/>

Contents

DITA Specialization in FrameMaker

DITA specialization	1
Special handling of DITA elements in FrameMaker	3
Specializing DITA elements in FrameMaker	4
Create DITA DTDs for specialization	6
Create the specialized EDD from the DTD	10
Create the structured template	11
Create the read/write rule file	11
Update the structured application definition	11
Publishing specialized topics	12

DITA Specialization in FrameMaker

DITA specialization

About specialization

Specialization is the process of creating new designs based on existing designs. A specialization can reuse elements from higher-level designs. You can specialize DITA to create customized information models that meet your business requirements while retaining the benefits of the existing DITA architecture.

The DITA architecture provides for a general base topic and its three specialized variations: concept, task, and reference topic types. Each specialized topic type DTD declares elements that are specific and restricted to only that topic type. For example, the task topic DTD allows <step> and <choice> elements that are absent from other topic DTDs.

You can specialize task, concept, and reference DITA topic types when you require new structural types or new domains. For example, you can tweak your design to increase consistency or descriptiveness or to meet specific output needs.

Types of specialization

Specialization can be broadly categorized into two types:

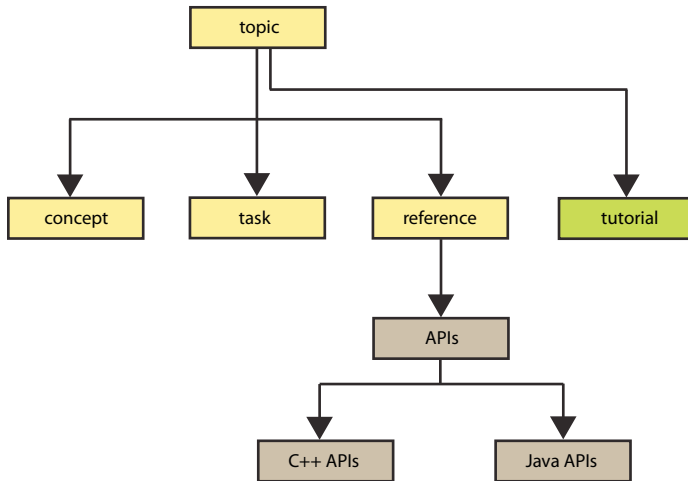
Structural specialization Defines new topic or map structures derived from base topics and maps, such as concept, task, or reference.

Domain specialization Defines markup for a specific information domain or subject area, such as programming or hardware.

To specialize an existing component, add the specialization statements to your DITA files, clearly identifying the ancestry or evolutionary path of your specialization. This way you can retain at least a minimum level of semantic structure during information reuse or interchange.

Structural specialization

Structural specialization defines new types of structured information, such as new topic types or new map types. Structural specialization allows you to create new topic types and yet maintain compatibility with existing style sheets, transforms, and processes.



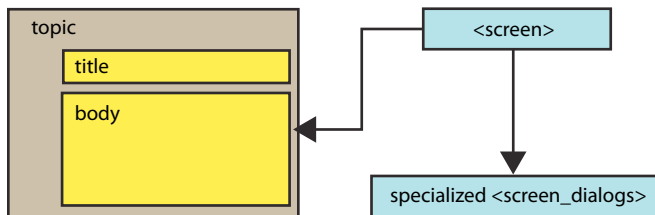
Shows the DITA base topics, concept, task, and reference, and specialized topics, tutorial and APIs. API topic is further specialized into Java APIs and C++ APIs.

Structural specializations declare new top-level topic types and map types. You use structural specializations to define entirely new document structures. With structural specializations, you can specialize any base element type, including topic and map, as well as elements in any topic or map specialization.

During structural specialization, you begin from the topmost level in the hierarchy, such as a topic or map. You then create a specialized version of the topic type and continue specializing elements as required.

Domain specialization

Domain specialization defines new types of elements, such as a new type of the <paragraph> element. These element specializations can be specific to a particular information domain or subject area, such as programming or hardware. For example, you can specialize the <screen> element from the user interface domain to catalog all dialog boxes.



Shows domain specialization of the <screen> element to provide a consistent and specialized structure to dialog box documentation.

When implementing domain specialization, you define new elements for use within any topic or map type. Using domain specializations, you can specialize any base DITA element that is an allowed descendant of <topic> or <map> or any element in any other domain module.

For example, map-specific elements <topicref>, <topichead>, and <topicgroup> are defined as domain elements, even though you might expect them to be part of the base <map> type. Therefore, defining specializations of <topicref> or <topichead> is domain specialization, not map specialization.

Why and when to use specialization?

You use specialization to define new structural types or new information domains. Specialization provides a way to adapt your design for increased consistency or descriptiveness. You can also use specialization for specific output demands that the current data model does not meet.

All specializations can be processed by existing transforms and can also be transformed back to more general equivalents.

Benefits of specialization include the following:

- You can reuse the base vocabulary to define specializations and save the time it takes to create them.
- Changes to base elements automatically percolate to the specializations.
- You can easily plug in modules depending on requirements.
- You can revert specializations to their base types easily.
- Interoperability or mapping from specialized type documents to base type documents is guaranteed.

Rules of specialization

You cannot make the content models of specialized element types less restrictive than the content models of their base types.

When you specialize one element from another, the new element must obey certain rules to be valid.

- A specialized element must have a content model equivalent to or more restrictive than its base element.
- A specialized element must have attributes that are equivalent to or a subset of the attributes of its base element.
- The attributes of a specialized element must have values or value ranges that are equivalent to or a subset of the values or value ranges of the base element.
- A specialized element must have a properly formed class attribute.

Special handling of DITA elements in FrameMaker

For special elements such as `<table>`, `<image>`, `<indexterm>`, and `<footnote>`, the EDD contains special information denoting type. All specialized elements for such special elements must also have corresponding information in the EDD. Only then are the specialized elements interpreted correctly.

Following is the list of elements with special handling in Adobe® FrameMaker® 9. If you create specialized forms of any of these elements, include the corresponding information in their EDDs as well.

topicref Include the `<fm-topicreflabel>` as the valid first element in the general rule of its specialized `<topicref>`. Functionality, such as update reference, open all `<topicref>`, `<conref topic ref>`, `<navtitle>` becomes available for specialized `<topicref>` elements.

indexterm If index term import and export processing is enabled in the DITA Options dialog box, then nesting of `<indexterm>` and `<Index-see>`, `<index-see-also>`, `<index-sort-as>` is also available for specialized index term elements.

table/simpletable/retable/choicetable DITA table elements do not contain `<numCols>` and `<colWidth>` properties that you must set explicitly in `ditafm.ini` for `<retable>` and `<simpletable>` and for elements specialized from them. When specializing `<retable>` and `<simpletable>`, add elements parallel to `<fm-retablemeta>` and `<fm-chtheadrow>`, with a structure similar to the base elements in the EDD file. Then make similar declarations in the read/write rule file for the specialized elements. When you insert specialized `<table>`, `<simpletable>`, or `<retable>` elements, the names of the specialized elements appear in the Insert Table dialog box.

topic/map If the correct specialized `<topic>` or `<map>` application is set in DITA Options, the specialized `<topic>` or `<map>` name appears in the DITA > New DITA file submenu. Composite FrameMaker documents or book files with FrameMaker document functionality are available for the specialized topics as well.

image/alt Ensure that you declare the specialized image elements in the read/write rule file for the specialized image elements to work as FrameMaker graphic objects.

Note: FrameMaker 9 does not support specializations of the <alt> element.

xref/link When you specialize the <xref> element in the EDD file, ensure that the element <fm-xref> is available wherever the specialized <xref> element is available. When you insert specialized <xref> or <link> elements in a DITA document, the DITA Cross Reference dialog opens. The names of the specialized <xref> and <link> elements are available in this dialog box.

linktext Ensure that you add the <fm-linktext> element in the EDD, as a valid choice, at all occurrences of the specialized <linktext> element.

prolog/draft-comment The DITA Options dialog box has options for conditionalizing <prolog> or <draft-comment> elements when the file opens. If these options are selected, then any specializations of <prolog> and <draft-comment> elements are conditionalized as well.

fn You must declare the specialized <fn> element as a footnote in the read/write rules file as well.

Note: For all these elements, the empty class attribute is allowed for the base element.

Specializing DITA elements in FrameMaker

The major tasks in specialization are planning the list of required specialized elements and defining where they fit in the existing DITA hierarchy. Avoid overspecialized elements and crude specializations. Don't create specialized elements when existing elements suffice.

Follow this workflow when specializing topics:

- 1 Create a new set of DTDs defining the new set of elements derived from existing element types.
- 2 Combine the specialized DTDs into a base DTD.
- 3 Create a new read/write rule file using the existing standard Topic/Map read/write rule file. Add element mappings for specialized elements derived from DITA elements, which already have some mappings defined.

Often a specialized element derives from a base element that has a declaration statement in the read/write rule file. In this case, include the same declaration in the read/write rule file for the specialized element.

For example, suppose you map the DITA <image> element to the FrameMaker <graphic> element. In this case, the DITA <image> element and its specialization must include the same declaration in their read/write rule files.

Similarly, if there is an unwrap statement for a base element, then elements derived from it should contain the unwrap statement as well.

- 4 Set the class attributes of all the specialized elements in EDD.

The correct element hierarchy should be created from the base element to the specialized one.

- 5 Import the base DTD as EDD in FrameMaker using the read/write rule file and DTDs created in steps 1 to 3.

- 6 Make FrameMaker-specific changes in the EDD file. More specifically, make these topic specializations:

- a Copy all the elements starting with FM- from the standard topic EDD to the new EDD. These are FrameMaker-specific elements declared to handle special objects, such as <table>, <crossref>, and <image>.
- b Change the content model of the element properties to "fm-propheading?, fm-propertybody+".
- c Change the content model <choicetable> to "chhead, fm-chbody" and of <simpletable> to "sthead, fm-stbody".

- d Change the content model of <sthead> to “fm-stheadrow” and of <chhead> to “fm-chheadrow”.
 - e Add an <fm-xref> element to the content model of <xref>, <syntaxdiagram>, <synblk>, <groupseq>, <groupchoice>, <groupcomp>, <fragment>, and <fig> elements and all other elements where <xref> is valid.
 - f Hide the elements <fm-graphic>, <alt>, <index-base>, <index-see>, <index-see-also>, and <index-see-also> using conditional tags.
- 7 Make these map specializations in the EDD:
- a Copy all the elements starting with FM- from the standard topic EDD to the new EDD. These are FrameMaker-specific elements declared to handle special objects, such as <table>, <crossref>, and <image>.
 - b Change the content model of the element properties to “fm-propheading?, fm-propertybody+”.
 - c Change the content model <choicetable> to “chhead, fm-chbody” and of <simpletable> to “sthead, fm-stbody”.
 - d Change the content model of <sthead> to “fm-stheadrow” and of <chhead> to “fm-chheadrow”.
 - e Add an <fm-xref> element to the content model of <xref>, <syntaxdiagram>, <synblk>, <groupseq>, <groupchoice>, <groupcomp>, <fragment>, and <fig> elements and all other elements where <xref> is valid.
 - f Hide the elements <fm-graphic>, <alt>, <index-base>, <index-see>, <index-see-also>, and <index-see-also> using conditional tags.
 - g Change the content model of reltable to <fm-reltablemeta>?, <relheader>?, <fm-reltablebody>.
 - h Change the content model of relheader to (fm-relheaderrow)
 - i Add the following elements to Map EDD :

Element (Table Row): fm-relheaderrow
General rule: relcolspec+

Element (Table Body): fm-reltablebody
General rule: relrow+

- j Add fm-topicreflabel to content model of <topicref>.
- 8 Copy the element format of the standard DITA elements from the default DITA topic or map EDD to the new EDD. Also define the formatting rules for the new specialized elements in the EDD.
- 9 Import the EDD into a new template file. Then import paragraph and character formats from the standard DITA template into this new template. Create a new structured application for specialized elements using the template, read/write rules, and the integrated DTD file created in the previous steps.
- 10 Select the name of the new application from the DITA Topic Application or DITA Map Application list boxes in the DITA Options dialog box. Click Save.

Your specialized Topic/Map now appears in the DITA > New DITA file submenu. Start authoring.

Create DITA DTDs for specialization

DITA DTDs are divided into smaller modules that reflect the base elements hierarchy (Topic and Map) and their respective domain and structural specializations like Task, Concept, BookMap, UIDomain, Programming Domain, and so on. The fixed set of changes that you make in DTDs is defined below.

Note: If you have FrameMaker 9 installed on your machine, you can access DITA DTDs or .mod files from: `<installation_directory>/Adobe/FrameMaker9/Structure/xml/DITA/app/dtd` You can also download all the sample files used in the steps for structural specialization from: https://share.acrobat.com/DITA_Specialization.

Modify DTDs for structural specialization

Implementing structural specialization is a two-step procedure:

- 1 Create a .mod file containing the definitions of the specialized elements.
- 2 Integrate the .mod file with the existing DITA DTDs in database.dtd.

To make your specialized elements types work with the existing `<topic>` hierarchies, add your specialization to database.dtd. Alternatively, you can create a separate DTD.

For `<map>` specialization, modify and use map.dtd or bookmap.dtd.

The following example defines a new specialized object element with only specialized `<xref>` and `<footnote>` elements as the content model.

Create the .mod file for structural specialization for `<topic>`

- 1 Copy any existing .mod file and rename it. For example, copy reference.mod and save it as objectsp.mod.
- 2 Open the new .mod file, objectsp.mod. In the section Specialization Of Declared Elements, change the info-type declaration to the new specialized structure type. The specialized structure type is required for integrating the specialized modules with existing ones. For example, replace the line:

```
<!-- ===== -->
<!--           SPECIALIZATION OF DECLARED ELEMENTS           -->
<!-- ===== -->

<!ENTITY % reference-info-types "%info-types;" >
```

with this one:

```
<!-- ===== -->
<!--           SPECIALIZATION OF DECLARED ELEMENTS           -->
<!-- ===== -->

<!ENTITY % objectsp-info-types "%info-types;" >
```

Note: Similarly in the following three steps, remove the existing declarations in the specified sections and replace them with the information for the new elements. This way you retain the formatted structure of the existing DTDs and map parallel information when declaring new elements.

3 Declare the new entities for the specialized elements required, up to the top of the hierarchy.

```
<!-- ===== -->
<!--          ELEMENT NAME ENTITIES          -->
<!-- ===== -->
<!ENTITY % myobjecttype "myobjecttype" >
<!ENTITY % mybody       "mybody" >
<!ENTITY % myp          "myp" >
<!ENTITY % myobject     "myobject" >
<!ENTITY % myxref       "myxref" >
<!ENTITY % myfootnote   "myfootnote" >
<!-- ===== -->
```

4 Declare the new specialized elements, and so on, for other elements.

```
<!-- ===== -->
<!--          ELEMENT DECLARATIONS          -->
<!-- ===== -->
<!--          LONG NAME: myobject           -->
<!--          ----- -->
<!ELEMENT myobject      (({%myxref;}*, {%myFootnote;}*) >
<!ATTLIST myobject
  declare      (declare)          #IMPLIED
  classid      CDATA              #IMPLIED
  codebase     CDATA              #IMPLIED
  data         CDATA              #IMPLIED
  type         CDATA              #IMPLIED
  codetype     CDATA              #IMPLIED
  archive      CDATA              #IMPLIED
  standby      CDATA              #IMPLIED
  height       NMTOKEN            #IMPLIED
  width        NMTOKEN            #IMPLIED
  usemap       CDATA              #IMPLIED
  name         CDATA              #IMPLIED
  tabindex     NMTOKEN            #IMPLIED
  longdescref  CDATA              #IMPLIED
  %univ-atts;
  outputclass  CDATA              #IMPLIED
  longdescre   CDATA              #IMPLIED >
```

5 In the Specialization Attribute Declarations section, declare the element from which the specialized element is derived. You must declare the hierarchy down to the base <topic> or <map> type (starting with a “-” for structural specialization). For example, if the specialized element is derived from a reference element, include the complete hierarchy:

- topic/reference/refbody specialtopic/specialbody

Add the following lines to the Specialization Attribute Declarations section:

```
<!-- ===== -->
<!--          SPECIALIZATION ATTRIBUTE DECLARATIONS          -->
<!-- ===== -->
<!ATTLIST myobjecttype  %global-atts; class CDATA "- topic/topic myobjecttype/myobjecttype ">
<!ATTLIST mybody        %global-atts; class CDATA "- topic/body myobjecttype/mybody " >
<!ATTLIST myp           %global-atts; class CDATA "- topic/p myobjecttype/myp " >
<!ATTLIST myxref        %global-atts; class CDATA "- topic/xref myobjecttype/myxref " >
<!ATTLIST myobject      %global-atts; class CDATA "- topic/object myobjecttype/myobject " >
<!ATTLIST myfootnote    %global-atts; class CDATA "- topic/fn myobjecttype/myfootnote " >
```

Update ditabase.dtd

Integrate the new .mod file with the existing ones by modifying ditabase.dtd.

Note: To avoid overwriting the original `database.dtd`, you can rename it to `databaseObjectsp.dtd` for this example.

- 1 In the Topic Nesting Override section, add the declaration for the specialized topic type. Ensure you include the base type information in the declaration.

```
<!-- ===== -->
<!--          TOPIC NESTING OVERRIDE          -->
<!-- ===== -->
<!ENTITY % info-types "topic | concept | task | reference |
                    glossentry | myobjecttype" >
```

- 2 To import the new `.mod` file, add an entry in the Topic Element Integration of the `database.dtd`.

```
<!-- ===== -->
<!--          TOPIC ELEMENT INTEGRATION          -->
<!-- ===== -->
<!--          Embed topic to get generic elements          -->
<!ENTITY % topic-type PUBLIC "-//OASIS//ELEMENTS DITA Topic//EN"
"topic.mod" >
%topic-type;

<!ENTITY % objectsp-type PUBLIC "-//OASIS//ELEMENTS DITA Topic//EN"
"objectsp.mod" >
%objectsp-type;
```

Note: To restrict multiple topic types in a single topic type, create an integration file but don't integrate topic types together as shown in this example.

Modify DTDs for domain specialization

DITA domains are implemented with two files:

- A **.mod** file that declares the elements for the domain.
- A **.ent** file that declares the entities for the domain.

For domain specialization, create both files. In the `.mod` file, declare the specialized elements; in the `.ent` file, declare the entities for integration-related information. The `.ent` file is required because domain specialized elements must be available wherever their base elements are.

After creating these files, update the `database.dtd` for implementing domain specialization for `<topic>`. The steps in the following sections define three new domain specialized elements for `<image>`, `<prolog>`, and `<link>` for `<topic>`.

For implementing domain specialization for `<map>`, follow the same procedure but edit the following files:

- `MapGroup.mod` for declaring the elements for the domain.
- `MapGroup.ent` for declaring the entities for the domain.
- `Map.dtd` or `BookMap.dtd` for integrating the `.mod` and `.ent` files.

Create the `.mod` file

- 1 Copy any existing `.mod` file and rename it. For example, copy `utilitiesDomain.mod` and save it as `domainsp.mod`.

Note: In the following three steps, remove the existing declarations in the specified sections and replace them with the information for the new elements. This way you retain the formatted structure of the existing DTDs and map parallel information when declaring new elements.

- Open the new mod file, domainsp.mod. In the section Element Name Entities, declare the new entities for the specialized elements.

```
<!-- ===== -->
<!--          ELEMENT NAME ENTITIES          -->
<!-- ===== -->
<!ENTITY % Dlink      "Dlink"                >
<!ENTITY % Dprolog    "Dprolog"              >
<!ENTITY % Dimage     "Dimage"                >
<!-- ===== -->
```

- Declare the new specialized elements. Copy the following lines for the specialized element, <Dimage>.

```
<!-- ===== -->
<!--          ELEMENT DECLARATIONS          -->
<!-- ===== -->
<!--          LONG NAME: Dimage              -->
<!ELEMENT Dimage      ({%alt;})              >
<!ATTLIST Dimage
  href          CDATA          #REQUIRED
  keyref        NMTOKEN       #IMPLIED
  alt           CDATA          #IMPLIED
  longdescref   CDATA          #IMPLIED
  height        NMTOKEN       #IMPLIED
  width         NMTOKEN       #IMPLIED
  align         CDATA          #IMPLIED
  scale         NMTOKEN       #IMPLIED
  placement (inline | break | -dita-use-conref-target) "inline"
  %univ-atts;
  outputclass   CDATA          #IMPLIED >
<!-- ===== -->
```

- In the Specialization Attribute Declarations section, declare the element from which the specialized element is derived. Declare the hierarchy down to the base <topic> or <map> type (starting with a "+" for domain specialization). For example, if the specialized element is derived from another utility domain element, define the complete hierarchy from specialized element to utilities domain to topic. (The utilities domain is specialized from <topic>.)

```
<!-- ===== -->
<!--          SPECIALIZATION ATTRIBUTE DECLARATIONS          -->
<!-- ===== -->
<!ATTLIST Dprolog %global-atts; class CDATA "+ topic/prolog domainsp-d/Dprolog " >
<!ATTLIST Dlink %global-atts; class CDATA "+ topic/link domainsp-d/Dlink " >
<!ATTLIST Dimage %global-atts; class CDATA "+ topic/image domainsp-d/Dimage " >
```

Create the .ent file

- Create the .ent file with the filename domainsp.ent.

The information in this file allows the elements to be substituted instead of aggregated. That is, wherever the base element is allowed, its specialized element is also allowed.

Note: As with the .mod files, you can rename an existing .ent file and replace the declaration statements as required.

- Open the .ent file and declare the entities for integration of new elements with the existing ones (using domain extensions).

```
<!-- ===== -->
<!--          ELEMENT EXTENSION ENTITY DECLARATIONS          -->
<!-- ===== -->
<!ENTITY % domainsp-d-image "Dimage"          >
<!ENTITY % domainsp-d-link  "Dlink"           >
<!ENTITY % domainsp-d-prolog "Dprolog"        >
```

- 3 Declare the domain attribute entity to define the ancestry down to the root from which the elements are derived. If you are specializing any element from some domain extension, then you need to declare up to the top.

```
<!-- ===== -->
<!--          DOMAIN ENTITY DECLARATION          -->
<!-- ===== -->
<!ENTITY domainsp-d-att "(topic ank-d)">
```

Update ditabase.dtd

Integrate the specialized .mod file with the existing ones by modifying ditabase.dtd. For domain specialization, specify both the .mod and .ent files in the ditabase.dtd as follows:

- 1 Define the new domain in the vocabulary section in the Domain Entity Declarations section.

```
<!-- ===== -->
<!--          DOMAIN ENTITY DECLARATION          -->
<!-- ===== -->
<!ENTITY % domainsp-d-dec PUBLIC "-//domainsp//ENTITIES DITA domainsp Domain//EN" "domainsp.ent" >
%domainsp-d-dec;
<!------- and the other existing ones ----->
```

- 2 Define the vocabulary substitution for the specialized elements. Include the elements from which the domain specialized elements extend.

```
<!-- ===== -->
<!--          DOMAIN EXTENSIONS          -->
<!-- ===== -->
<!ENTITY % image "image | %domainsp-d-image;" >
<!ENTITY % prolog "prolog | %domainsp-d-prolog;" >
<!ENTITY % link "link | %domainsp-d-link;" >
<!------- and the other existing ones ----->
```

- 3 Add the vocabulary attribute declaration statements.

```
<!-- ===== -->
<!--          DOMAINS ATTRIBUTE OVERRIDE          -->
<!-- ===== -->
<!ENTITY included-domains "%ui-d-att; %hi-d-att; %pr-d-att; %sw-d-att;
%ut-d-att; %indexing-d-att; %domainsp-d-att;" >
```

- 4 Specify the vocabulary definition and include the .mod file for domain element integration. This entry includes all the specialized elements declared in the .mod file.

```
<!--===== -->
<!--          DOMAIN ELEMENT INTEGRATION          -->
<!--===== -->
<!ENTITY % domainsp-doctype PUBLIC "-//domainsp//ELEMENTS DITA User Interface Domain//EN" "domainsp.mod " >
%domainsp-doctype;
```

Create the specialized EDD from the DTD

You now create the specialized EDD from the specialized DTDs. You can do this by copying the topic.edd.fm and renaming it. For example, you can give it the new name ObjectSp.edd.fm.

Note: You can locate the topic EDD file from

<installation_directory>/Adobe/FrameMaker9/Structure/xml/DITA/app/DITA-Topic-FM. Copy this to a new folder, <installation_directory>/Adobe/FrameMaker9/Structure/xml/DITA/app/ObjectSp.

- 1 Copy topic.edd.fm and save it under the name ObjectSp.edd.fm. Open the ObjectSp.edd.fm file.
- 2 Select StructureTools > Import DTD and select the specialized DTD you created previously.
- 3 Save the specialized EDD file.

Create the structured template

- 1 Copy the dita-topic.template.fm file and rename it to objectsp.template.fm.
- 2 Select File -> Import -> Element Definitions and import the element definitions from the specialized EDD into the template.
- 3 Save and close the template file.

Create the read/write rule file

Usually, you update the read/write rules file only if the specialized elements have a modified import or export behavior.

- ❖ Copy the topic.rules.txt file and save it as ObjectSptopic.rules.txt.

Update the structured application definition

To use new specialized DTDs, you must specify the structured application definition.

- 1 Select StructureTools > Edit Application Definitions.

Note: You can locate the structapps.fm file from <installation_directory>/Adobe/FrameMaker9/Structure folder. For this example, you can rename the structapps to structappsObjectSp.fm.

- 2 In the structure view, insert an <XMLApplication> element and name it ObjectSp. Add the following elements to the <XMLApplication> element.
- 3 Add the <Template> element and specify the path to the specialized template, objectsp.template.fm.
- 4 Add the <DTD> element and specify the path to the DTD, ditabaseObjectsp.dtd.
- 5 Add the <ReadWriteRules> element and specify the path to the rules file, ObjectSptopic.rules.txt.

Application name:	ObjectSp
Template:	\$STRUCTDIR\xml\dita\app\DITA-Topic-FM\objectsp\template.fm
DTD:	\$STRUCTDIR\xml\dita\app\dt\d\databaseObjectsp.dtd
Read/write rules:	\$STRUCTDIR\xml\dita\app\DITA-Topic-FM\ObjectSp- topic.rules.txt
DOCTYPE:	topic task concept reference glossentry dita
	myobjecttype
Conditional Text:	
Output Text PI:	OutputAllTextWithPIs
Entity locations	
Public ID:	--OASIS//DTD DITA Topic//EN
Filename:	\$STRUCTDIR\xml\dita\app\dt\d\topic.dtd
Public ID:	--OASIS//DTD DITA Task//EN
Filename:	\$STRUCTDIR\xml\dita\app\dt\d\task.dtd
Public ID:	--OASIS//DTD DITA Concept//EN
Filename:	\$STRUCTDIR\xml\dita\app\dt\d\concept.dtd
Public ID:	--OASIS//DTD DITA Reference//EN
Filename:	\$STRUCTDIR\xml\dita\app\dt\d\reference.dtd
Public ID:	--OASIS//DTD DITA Glossary//EN
Filename:	\$STRUCTDIR\xml\dita\app\dt\d\glossary.dtd
Public ID:	--OASIS//DTD DITA Composite//EN
Filename:	\$STRUCTDIR\xml\dita\app\dt\d\databaseObjectsp.dtd
Public ID:	--IBM//DTD DITA Topic//EN
Filename:	\$STRUCTDIR\xml\dita\app\dt\d\topic.dtd
Public ID:	--IBM//DTD DITA Task//EN
Filename:	\$STRUCTDIR\xml\dita\app\dt\d\task.dtd
Public ID:	--IBM//DTD DITA Concept//EN
Filename:	\$STRUCTDIR\xml\dita\app\dt\d\concept.dtd
Public ID:	--IBM//DTD DITA Reference//EN
Filename:	\$STRUCTDIR\xml\dita\app\dt\d\reference.dtd
Public ID:	--IBM//DTD DITA Glossary//EN
Filename:	\$STRUCTDIR\xml\dita\app\dt\d\glossary.dtd
Public ID:	--IBM//DTD DITA Composite//EN
Filename:	\$STRUCTDIR\xml\dita\app\dt\d\databaseObjectsp.dtd
Use API client:	dita_fm_app

Structured application specification for ObjectSp

- 6 Save and close the file.
- 7 Close and restart FrameMaker.
- 8 Select DITA -> DITA Options and choose ObjectSp from the DITA Topic Applications list. You will find the new topic type appearing under the DITA -> New DITA File submenu.
- 9 Create a topic from the specialized file to begin authoring.

Publishing specialized topics

In DITA specialization you have the advantage of processing specialized content with unspecialized, general tools. However, these tools process the elements according to the general content model from which the specialization is derived. For example, specialized forms of <paragraph> are still formatted as paragraphs.

To fine-tune or deviate from the base formats, you can modify EDDs, XSLT stylesheets, templates, and read/write rules within the FrameMaker environment. After making these changes, you can publish Adobe PDFs with the new format definitions from FrameMaker.