Concepts of the MSR application profile V2.x.x

MSR-MEDOC Taskforce DTD, Bernhard Weichel, Robert Bosch GmbH

Abstract

The datamodels defined for MEDOC are implemented as SGML/XML DTDs. All these MSR DTDs while implementing specific data models for their domain follow the same basic design principles. This document describes these design principles such that DTD authors can develop MSR compliant DTDs.

This document shall be used together with an elements and attribute description for more details, in particular for examples.

This document is intended for developers of MSR-MEDOC comliant DTDs.

Table of Contents

Introduction

Companies

MSR-MEDOC Taskforce DTD [MSR-MEDOC]

Name Roles	Departement	Address	Contact
Herbert Klein Author	XI-Works		
Bernhard Weichel	Robert Bosch GmbH		

Version Information

Document Part	Editor			
	Company	Version	State	Remarks
2 WD 30.3.2001 For details refer to nr. 1, Page 31	Bernhard Weichel			

1 What is the MSR-MEDOC SGML/XML Application Profile?

The MSR-MEDOC SGML/XML application profile is a set design principles, used within the MSR-MEDOC DTDs. Design Goals are:

- self explanatory names
- consistent design principles: do it the same way as much as possible
- Instances shall be balanced (in XML times: well formed)
- Sequence of elements according to the logical sequence of the contents[1]
- supporting standards such as OASIS-Open Exchange table model *in SGML Open Technical Memorandum TM 9502:1995. (http://www.oasis-open.org/html/a502.htm)*
- Make the DTD usable in SGML/XML editors directly
- clarity is more important than file size
- use the DTD as a ”checklist”
- apply the DTD in various use cases and processes

The MSR-MEDOC working group DTD apologizes if these principles are violated. Appropriate feedback is welcome to *Herbert Klein, Xiworks (mailto:herbert.klein@xiworks.de)*

1

2 Naming Conventions

The following **naming conventions** exist:

The number of elements increases dramatically, particularly when defining content-oriented DTDs. To provide the user with a little orientation help with regard to document acquisition, the element designations were given ”meaningful” names. This can only be achieved with a name length of more than 8 characters (max. 32 characters[2]).

All elements and attribute designations are fundamentally defined with lower-case letters. Furthermore, the frequency of usage of the elements was a factor in the length of the name. Elements occurring frequently received short designations.

Example paragraph —> **< p >**

index entry —> **< ie >**

Several terms were connected with a hyphen ”-” for elements whose semantics are not definable with just one term. However, abbreviations were used so that the element designations would not be too long.

Example: voltage mismatch test —> **< volt-mismatch-test >**

All name abbreviations shallb e used uniform throughout the entire DTD.

Table 1: list of abbreviations

abbreviation	meaning
ADD	additional
ADMIN	administrative
ARG	argument
AV	available
CALPRM	calibration parameter
CHAR	characteristic
CHARS	characteristics (plural)
COEFFS	wrapper for coefficient
COMPU	computation
CS	calibraion state
DEF	definition
DESC	description
DIFF	difference
DIR	directory
DOC	document or documentation
ENV	environment
FIELD	data fiield
IMPL	implementation
IMPLS	wrapper for multiple implementations
INFO	information
INFOS	wrapper for information

2

Table 1 (Cont.): list of abbreviations

abbreviation	meaning
INIT	initial, initialization
INV	inverseINV to INVERSE
LOCS	locations
MATH	mathematical expressoin
MAX	maximum
MC	measurement and calibration
MEM	memory
NAME	name of an object
NUMBER	used for amount or number of objects
OPER	operating
ORDER	order of a sequence, also used in term of ordering a product
ORIGIN	ursprung
PHYS	physical
PPM	parts per million
PRM	parameter used in the semantic of property
PROPS	properties
PTS	points - wurde in ASAP PTS genannt
REF	formal reference
REFERENCE	reference which is not expressed in SGML/XML (e.g. normative refererence)
REFS	wrapper for ref
SEG	segment
SET	a set of objects
SPEC	specification (this is more formal than description)
STD	standard
SW	software
TABLE	this is always a table in layout sense
TBD	to be defined
TBR	to be resolved
TOL	tolerance
TT	technical term
UNIT	measurement unit
VC	variant coding

3 design patterns

The MSR-MEDOC SGML/XML application profile follows certain design patterns in order to make it easier to understand the DTD and to utilize it in various use cases.

3.1 predefined structures

The automotive systems to be described with the help of a DTD may be described in various aspects which are all reflected in the DTD.

When a DTD is used this way as a checklist, it is appropriate, to use the DTD in ”strict” mode which turns many elements to be mandatory in its context. But nevertheless, the specification of a particular topic, e.g. ”acoustic characteristics” might not make sense or might only become necessary later on, depending on the project.

In this case the DTD supports an approach where the availability of the information can explicitly expressed. The content model then shall provide an alternative of

- The intended content
- one of **< na >**, **< tbd >**, **< tbr >**:

  < na > the intended content is **not applicable** for technical or organizational reasons. As an example, if a spark plug has no embedded software the software related elements are not applicable.

  < tbd > the intended content is **to be determined** (to be done). In this case the related information must be worked out. The responsible persons, intended schedule and description of the task can be noted in **< team-member-ref >**, **< schedule >**, **< desc >**.

```
<sw-architecture>
<tbd>
  <team-member-ref>weichel</team-member-ref
  <schedule>2001-10-01</schedule>
  <desc>
     The architecture of the software must be worked out respectively
     taken over from a previous project.
  </desc>
</tbd>
<sw-architecture>
```

Figure 1: Example for use of tbd

  < tbr > Tjhe information is available but must be resolved and imported into the SGML/XML document. The information can be described in a certain level of detail.

An example of this design-pattern may be found in *sw-architecture.*

3.2 key Objects - mainly referrable objects

Referrable Objects mainly are key objects in the application domain such as part-types, variables, persons, projects and so on. These object usually establish a native object hierarchy. If appropriate, such objects shall be characterized by:

- a **short name** (**< short-name >**) providing a natural name in the application domain (e.g. a variable name: NL)
- a **long name** (**< long-name >**) providing a explanatory name (e.g. a varaible designator ”Idle speed”)

- a **concise description** (**< desc >**) providing one level more[3] of information than the long name
- an **introduction** (**< introduction >**) which may be used to provide e.g. managerial overview information such as description of contents, intended audience, hints how to read.
- a **name space** designator (fixed attribute **[f-namespace]**) indicating the scope for the short names descendant elements[4].
- a **class indicator** [5] (fixed attribute **[f-id-class]**) which describes the overall nature of this object. This indicator is used primarily to manage linking (see see Chapter 5 Linking p. 18) .
- a **identifier** (attribute **[id]**) implemented as an SGML/XML attribute to support SGML based linking
- a **category** (**< category >**) allowing to specify the particular nature of the object in question (see see Chapter 3.3 category p. 10). Categories can be defined by the user of a dtd. As a rule, categories may define subsets of elements with the same class indicator (**[f-id-class]**).
- **administrative data** (**< admin-data >**) providing information about intended document fragmentation, versioning, used languages, product data management (see see Chapter 3.4 admin-data p. 10).
- additional information (**< add-info >**) providing means to add information not supported by the particular content model.

These principles may be studied in *sw-variable.*

3.3 category

Categories (**< category >**) shall be used to classify objects in the application domain. In most cases, this categorization ends up in a subset of information which is appropriate for an object of the category in question. Therefore category may be used to provide suitable editor support (e.g. templates). Examples for categories are:

- Roles of an SGML/XML-file in the process (is it a requirement specifiation, a design document, or a particular data container such asl ASAM-2MCD-MC)
- kind of an calibration parameter such as single-value, curve, map
- kind of computation method such as (tablular, linear, scale)

Usually the category of an object could also be derived from its content, eg. by counting the number of axis of an calibration parameter. But for roubustness of processes, it is helpful to provide some sort of redundancy which can be used for check and support purposes. For example, a calibration parameter categorized as *map* must define exaclty two axis description (because a calibration map has two input and one value axis).

Examples may be studied in *sw-calprm*

3.4 admin-data

administrative data (**< admin-data >**) provide information about intended document fragmentation, versioning, used languages, product data management. Indicators for use of admin-data are:

3

4

5

- key objects of the application domain which are intended to be managed as an entity of its own.
- objects requiring specific company related information
- objects requiring particular versioning schemes
- objects requiring particular languages
- objects requiring paricular control of processing (controlled by **< formatter-ctrls >**

admin-data therefore shall be applied to key objects in the application domain (see see Chapter 3.2 key Objects - mainly referrable objects p. 9)

3.5 wrapper elements

Whenever an element occurs more than once, a wrapper element must be provided. This approach makes processing much easier, in particular for sequential (e.g. ”Simple API for XML” based processors). This principle shall be kept, even if file sizes is increased.

Hint:
If the wapper element is there it must at last be popoulated with one element. This makes it easier for sequenctial processors. Usually the wrapper element establishes the container (e.g. a table). If the wrapper is empty, then an malformed table would be the result. This principle is kept even we faced the experience that it is somewhat mor difficult for genrating tools to implement this. Such tools must first look if there are elements to process before they generate the wrapper.

Such wrapper elements shall be named as the plural of the repeated element as shown in see Figure 2 Example for wrapper elements p. 11.

```
<!ELEMENT SW-VARIABLES - - (SW-VARIABLE)+ >
```

Figure 2: Example for wrapper elements

For an example see *sw-variables*

In some cases, where the repeated element is the last one within its container, the wrapper may be omitted for easier handling as shown in see Figure 3 Omission of the wrapper element p. 11.

```
<!ELEMENT Behaviour-description   - - (System-behaviour, Function+) >
<!ELEMENT Function  - - (#PCDATA) >
```

Figure 3: Omission of the wrapper element

Both concepts are manageable with newer SGML tools (convertors, formatting).

3.6 Elements versus Attributes

It is not always easy to decide[6] if attributes or elements shall be used. The following policy shall be applied:

- use elements for the contents reflecting the application domain
- use attributes to implement features of processing such as namespaces, linking.

In spite of the fact, that attributes may provide value lists which may be utilized by an attribut dialogue in an editor, document content shall still be placed in elements. Support of value lists shall be provided by editor means, perhaps supported by DCI[7].

6

7

3.7 common attributes

The following common attributes shall be provided wherever appropriate:

- a container for a signature **[s]** which can be used to keep checksums in order to protect certain elements from occasionally being manipulated
- a container for a time stamp **[t]** which may be used to support change management on a time based baselining approach
- a container for a view indicator **[view]** which may be used to support multiple views within one SGML/XML instance. This attribute can be used to include/exclude the particular element within certain views (see see Chapter 3.8 View concepts p. 12)
- a container for semantic information **[si]** which may be used to indicate process specific semantics of an element. This attribute is inteded mainly for elements within the ”non content oriented information”. Examples for the usage of **[si]** are:
  - controlling the behavior of an interactive help system (*MSR-Hilfesystem (msrrep-help.chm)*
  - controlling the behavior of msr query (see see Chapter 4.1.4 MSR-QUERY: Importing data from other sources p. 16)
- attributes for
  - key objects (see)see Chapter 3.2 key Objects - mainly referrable objects p. 9
  - linking mechanisms (see see Chapter 5 Linking p. 18)

3.8 View concepts

Often it is required to support multiple views of a document being produced from one single source. Such requirements shall be fulfilled by

- utilizing category
- providing filters to create subsets being derived directly from the structure of the SGML/XML files utilizing the particular semantics of the elements
- utilizing **[view]** attribute which shall be added to any element which is a candidate to be hidden in particular document views. The semantics of the view attribute is as follows (see also see Figure 4 Example for using [view] attribute p. 12):
  - view takes a list of view names, in which the element in question shall appear. The element will be hidden in any other view.
  - if view is empty or not there it is always shown
  - if view starts with a ”-” the elment is hidden in all designated views.
  - the desired view must be selected by means of the processing tool

```
<p view=”print”>this paragraph appears in printed version only</p>
<p view=”html help”>this paragraph appears in html version as well as in the help file</p>
<p view=”- print”>this paragraph will not appear in the printed version</p>
<chapter>
  <long-name view=”print”>This headline will appear only in printed version<long-name>
  <p>This chapter will be invalide in non printed versions since its
        long-name will be hidden
  </p>
</chapter>
```

Figure 4: Example for using [view] attribute

Hint:
This approach can be implemented using a preprocessor which removes the hidden elements. But if the user does not use it correctly, it is possible to procduce invalid files, e.g. chapters without titles. This shall be handled by an appropriate editor support.

3.9 long-name versus label

Sometimes an element establishes a title but cannot be used as a link target. Such objects shall be characterided by

- **< label >** instead of **< long-name >**
- **< short-label >** instead of **< short-name >**

3.10 Additional specification and information

It is recommended to establish semantically meaningful structures whereever appropriate. But nevertheless, it is not possible to define the whole semantic of the desired object. Therefore additional means must be there to provide specifications and information.

- additional specifiation (**< add-spec >**) shall be provided to specify further subjects, mostly represented as further chapters in printed documents. The attribute **[si]** may then be used to denote a specific semantic of such specifications.

  This element shall be used to provde specifiation for which no particular content model exists.

  Example my be studied in *sw-system*

- additional information (**< add-info >**) shall bve provided to give further detail to the subject being currently described. In printed documents this information is mostly given as further paragraphs within a given chapter.

  This element shall be used to provide further information which is not supported by the current content model.

  Example my be studied in *sw-variable*

3.11 building subsets of content models (Architectures)

In some cases it is necessary to define subsets of content models. If working only with SGML this could be achieved by exclusion exceptions. These exceptions have serious drwabacks (in particular they are not supported in XML). Therefore a different approach shall be followed:

- Define the full featured content mode as base model and name it somehow (e.g. **< topic >**).
- Derive restricted content models from the base model and add a numerical suffix to the element name (such as **< topic-1 >**). As a rule, the higher the suffix is, the more restrictions are applied.

This approach makes it straightforward for processors and users to recognize similar content models.

Hint:
In the context of customizability (in particular multilinugal versions) of a dtd, it appears that elements have different restriction levels but still the same content model. In this case the restrictions concern features being turned off in the customization. One example is given by **< long-name >** and **< long-name-1 >**

3.12 Semantic Tables

A common problem when using content-oriented DTDs is that the visual representation is not obvious in an SGML/XML editor.

Therefore it is recommended that such objects are structured as canonical as possible in order to easily identify rows and columns for tabular display.

It may also be appropriate to use **< msr-query >** to provide means to keep a rendition oriented representation of the data. This approach was followed in MSRNET.DTD as well as in MSRFMEA.DTD [8]

[8]

4 Generally useful elements

4.1 Non content oriented information

In cases where no particular content model is possible or required, the DTD shall provide means to describe the subject in an informal manner. Within MSR this is called ” **n** on **c** ontent **o** riented **i** nformation” (**< ncoi >**). This approach provides generic word processing structures such as inline elements paragraphs, lists, tables, figures, chapters. Details may be found at *Benutzerhandbuch MSRREP (msrrep-ug-de.chm)* .

4.1.1 Captions

Element such as figures, formula and tables may have captions. In this case the may also serve as a link target. To support this approach, an optional caption element (e.g. **< figure-caption >**) shall be provided. this element gets the ID as well as the title.

```
<!ELEMENT figure - - (figure-caption? (graphic?, verbatim?) >
<!ELEMENT figure-caption - - (long-name?, short-name?)>
<!ATTLIST figure-caption
            ID ID #implied>
```

Figure 5: Example of a caption

4.1.2 CALS Table (Oasis-open exchange table model)

The CALS table model is used for user definable tables in most SGML DTDs. Since this model is also supported in most SGML tools (with graphic table editors, for example), **this model was integrated in the MSR DTDs** . For details see *9502:1995 CALS Table Model DTD (http://www.oasis-open.org/html/a502.html)* .

However, since this model on principle allows two-stage interlacing of ”tables” (with the help of the **< entrytbl >** element) with one another, it was modified in this area in accordance with the less demanding requirements.

Table 2: Deviations from CALS table model

CALS Table Element	MSR Modifications in the CALS Table Element Content
< table >	The **< title >** and **< shortitle >** elements are replaced by the **< long-name >** and the **< short-name >** element. None of the three exclusions (table, figure, chart) are used. The following attributes are omitted: inschlvl, delchlvl, label, hcp, sssn, unit, module, lru, assem, subassem, s-subassm, compon, partno, refdes, texttype, applictype, applicrefid, skilltrk, contype, assocfig, assoctab, security, restrict, release, codeword, scilevel, diglyph.
< tgroup >	The following attributes are omitted: security, restrict, release, codeword, scilevel, diglyph.
< colspec >	No modifications.
< spanspec >	No modifications.
< thead >	The following attributes are omitted: security, restrict, release, codeword, scilevel, diglyph.
< tfoot >	The following attributes are omitted: security, restrict, release, codeword, scilevel, diglyph.

Table 2 (Cont.): Deviations from CALS table model

< tbody >	The following attributes are omitted: security, restrict, release, codeword, scilevel, diglyph.
< row >	The **< entrytbl >** element is omitted. The following attributes are omitted: security, restrict, release, codeword, scilevel, diglyph.
< entry >	The content model of this element is completely changed. The following attributes are omitted: security, restrict, release, codeword, scilevel, diglyph.
< entrytbl >	Not used.

4.1.3 Integration of Graphics

The Integration of graphics is done with the element **< figure >**. The element **< figure >** has an optionas **< figure-caption >** with **< long-name >** for the title of the graphic, an optional **< short-name >** and a element **< graphic >** which is used io include the graphic. Therefore **< graphic >** provides the attributes **[filename]** which defines the file and **[notation]** which defines the format. MSR DTDs provide 2 ways of defining graphic files:

1. File name is user-definable when creating instances. Therfore **[filename]** is a CDATA attribute .
2. File names for all graphics are defined with entities in the SGML editor. Therefore the attribute **[filename]** has to be configured to be an ENTITIY in the dtd configuration.

4.1.4 MSR-QUERY: Importing data from other sources

< ncoi > also provides means to import data from other sources into e.g. into **< add-spec >**. This means is called ”MSR-Query (**< msr-query >**). In general such an import is characterized by:

- The properties of the data to be imported. This is specified by the name of the import algorithm (**< msr-query-name >**) , the arguments (**< msr-query-arg >**) specifying more details and a **< comment >** which describes the intended algorithm in human readable manner.
- the result section (**< msr-query-result >**) which is mainly **< ncoi >**.

Such imports may occur on any level such as chapters, paragraphs, and inline-elements. Therefore there are

- **< msr-query-chapter >** for results appearing as chapters
- **< msr-query-topic-1 >** for results appearing als topics within chapters
- **< msr-query-p >** for results appearing as block (or paragraph) level elements
- **< msr-query-text >** for results appearing as simple text perhaps embellished by inline elements.

In order to support msrquery, the DTD must provide **< msr-processing-logs >** as child of the root element. This will be used by query processors to log the query processing.

4.2 Project Data

If the DTD is used in context of a particular project, it is recommended to provide **< general-project-data >** as a container for project related information.

4.3 special-data

It is highly recommended to provide means to capture data which is not directly supported by the DTD. This approach removes the requirement to split the data to multiple files and to define a new file format as soon as the desired information set is not supported by the DTD.

In order to clearly identify such ”backdoors”, one particular subtree (**< special-data >**) shall be provided as child of the root element.

5 Linking

System requirements and product specifications contain many relationships between objects, for example, when signals are assigned to ports, or when a part is described several times and is simply referenced each time. It is also neccesary to describe relationships between object of different documents. E.g. to express the connection between a **< net-node-port >** in a *MSRNET* document and a **< net-port >** of an electronic control unit in a *MSRSYS* document.

Table 3: Supported linking approaches

	short name hierarchy	**HyTime and ID-IDREF**
formal link (e.g. **< sw-variable-ref >**)	supported	supported
informal link (**< xref >**)	not fully supported	supported
free link	not supported	partially supported by HyTime resp. by **< msr-query >**

5.1 Basic concepts

There are different types of links in the MSR DTDs.

formal links
: Formal links can reference to all addressable objects of a certain link class by name. E.g. the element **< company-ref >** references allways a **< company >**. The objects of a link class don't have to be neccessarily within the same document. E.g. a **< net-port-ref >** in a *MSRNET* document references a **< net-port >** in a *MSRSYS* document.

informal links
: Informal links can reference to all addressable objects of all link classes by name.

free links
: Free links are able to link any adressable portion of information. E.g. words, sentences, paragraphs or graphics. Resources participating in a free link can be addressed by name and/or postion. That is, the documents have to be read-only because changes in the documents may also change the postion of the participating resource. Examples for this are the attachment of annotations on existing documents, managment summaries and predefined reading paths (guided tours).

As of application profile 2 only formal links and informal links are used in the MSR DTDs.[9]. Resources participating in these links can be addressed by

- its natural name, specified by its **< short-name >** (see see Chapter 5.2 Short Name Hierarchy p. 19)
- HYTIME and CLINK (see see Chapter 5.3 Hytime CLink and Nameloc p. 20) which may be turned optional by DTD configuration (see Chapter 6.2 Configuring the DTDs p. 23)

The following conventions apply:

- The Short Name Hierarchy and the Hytime mechanisms can be used to define links in a MSR SGML Instance. However, at the interchange of the instances both mechanisms have to be defined in a consistent way.
- Linking within a single instance can't be defined with Hytime "nameloc". These links have to be defined either with pure ID/IDREF(clinks).
- Natural locators are not CASE sensitive.

9

5.2 Short Name Hierarchy

This linking mechanism is based on the idea of linking with names which the author already knows, e.g. names of ports, interfaces, signals, sw-variables etc.

5.2.1 Structure and building of natural locators

Each resource in a MSR document is unique within a particular namespace, e.g. a **< port >** is unique within a certain **< interface >**. All MSR DTDs have namespaces for all their resources. These namespaces are indicated by the attribute **[f-namespace]**. The attribute **[f-namespace]** contains all element names of which the element is a namespace. In the example below **< port-group >** is a namespace of **< port >** and **< interface >** is a namespace of **< port >**, **< port-group >** and **< module >**. Each which is a namespace for other elements has a **< short-name >** which can be used to identify this object. To build a unique natural locator the **< short-name >**s of the namespaces separated by "/"-characters are used to build a path to address the resource.

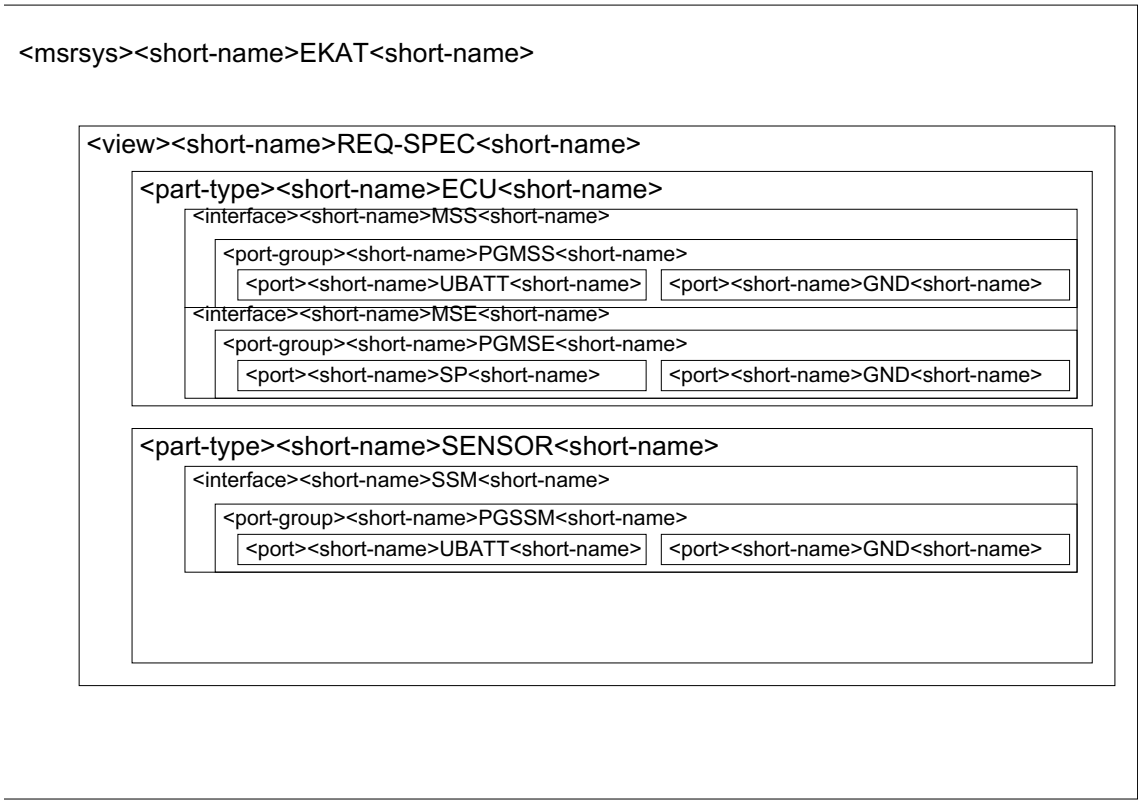


Figure 6: Example of a short name hierarchy

The full natural locator for the **< port >** "GND" within the **< interface >** "MSS" in the example above is "/EKAT/REQ-SPEC/ECU/MSS/GND". Full natural locators start always with a slash ("/"). The full locator is only necessary for links in other documents. Natural locators which only identify resources within the same document can be defined as short natural locators. In the example above the short natural locator of the **< port >** "GND" within the **< interface >** "MSS" is "MSS/GND". That is the short natural locator uses only the necessary namespace **< short-name >**s to identify a resource. The following table shows a list of the short natural locators of resources in the example above (see Figure 6 Example of a short name hierarchy p. 19).

Table 4: Example of resources

short-name	resource type [f-id-class]	unique short natural locator
GND	port	MSS/PGMSS/GND
GND	port	MSE/PGMSE/GND
GND	port	SSM/PGSSM/GND
UBATT	port	MSS/PGMSS/UBATT
UBATT	port	SSM/PGSSM/UBATT
SP	port	SP
PGMSS	port-group	PGMSS
PGMSE	port-group	PGMSE
PGSSM	port-group	PGSSM
MSS	interface	MSS
MSE	interface	MSE
SSM	interface	SSM
ECU	part-type	ECU
SENSOR	part-type	SENSOR

5.2.1.1 Link construction

All linking elements in the MSR DTDs (**< ...ref >**) are also used for links with short name hierarchies. The natural locator is described in the content of the linking element.

```
<port- ref>MSS/GND</port- ref>
```

If the DTD is configured with empty linking elements (dtd switch ”linking element content” see see Topic 6.2 Configuring the DTDs p. 23), the dtd provides an additional attribute **[natloc]** where the natural locator can be described.

```
<port- ref NATLOC=”MSS/GND”>
```

5.3 Hytime CLink and Nameloc

CLink and *NameLoc* are *HyTime* architectural forms(see *[/ Standard: Hypermedia/Time-based Structuring Language (HyTime) [ISO/IEC 10744] / Standard: Hypermedia/Time-based Structuring Language (HyTime) [ISO/IEC 10744] / Relevant Position: Hyperlinks module]*). These forms are assigned by using a predefined attribute name with predefined attribute values. The architectual form attribute name is *HyTime* . The following attribute values of the **[HyTime]** attribute are used in MSR DTDs:

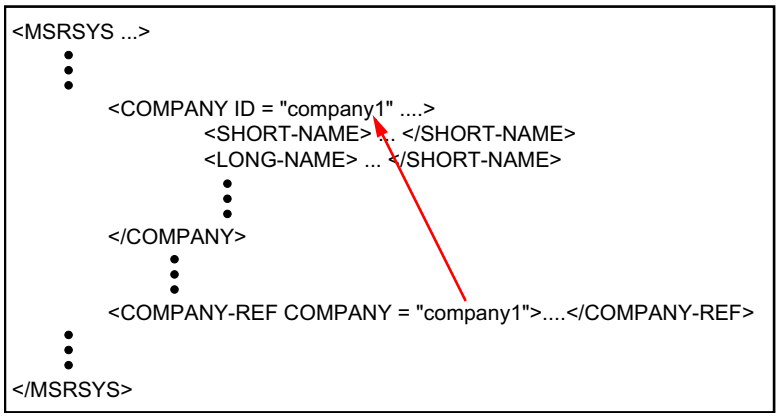
- *clink*
- *namloc*

5.3.1 CLink

CLink uses ID/IDREF to locate link target by name. If the CLink architectural form is used, the locator attribute has by default the name **[linkend]**. But *HyTime* gives the opportunity to change the name of the locator attribute. The *HyTime* attribute **[HyNames]** maps the name of the real locator attribute to **[linkend]**. The value of the *HyNames* attribute is the string ”linkend ” followed by the name of the locator attribute. If the name of the locator attribute is ”company”, then the value of the HyNames attribute has to be ”linkend company”. The locator attribute **[company]** is then

used to store the ID of the resource, to which the link is assigned. In MSR DTDs, the attributes **[HyTime]** and **[HyNames]** are fixed. Therefore they don't occur within an instance. The graphic below shows an example of a clink for linking within a single instance.The values of the **[HyTime]** attributes are as following:

- HyTime = "clink"
- HyNames = "linkend company"



clink__nohd.eps

Figure 7: Linking with CLink

5.3.2 NameLoc

The "NameLoc" architectural form is used to link to resources which are not within the same instance, i.e. to external resources. To be conform to the SGML ID/IDREF concept, it is necessary that the link targets are located within the same document. Hytime **< nameloc >** provides a virtual link target which represents an external resource within the document. So a standard SGML parser won't get any errors, because **< nameloc >** has an ID. Therefore **< nameloc >** has also a **< long-name >** and a **< short-name >** where the **< long-name >** and the **< short-name >** of the external resource can be copied. To reference to the external resource **< nameloc >** has an additional element named **< nmlist >**. The element **< nmlist >** has an attribute named **[docorsub]** and an attribute named **[nametype]**. The function of these attributes depends of the value of **[nametype]**. The values of **[nametype]** are either "element" or "entity".

If **[nametype]** = "element" the attribute **[docorsub]** contains the name of the external entity which contains the resource. And the content of the element **< nmlist >** contain the ID of this resource.

If **[nametype]** = "entity" the attribute **[docorsub]** is empty and the element **< nmlist >** contains the name of the entity which is refered.

Depending of the value of the attribute **[nametype]** the element **< nmlist >** refers either to an external document or to an element within an external document. The graphic below shows an example with **[nametype]** set to "element".

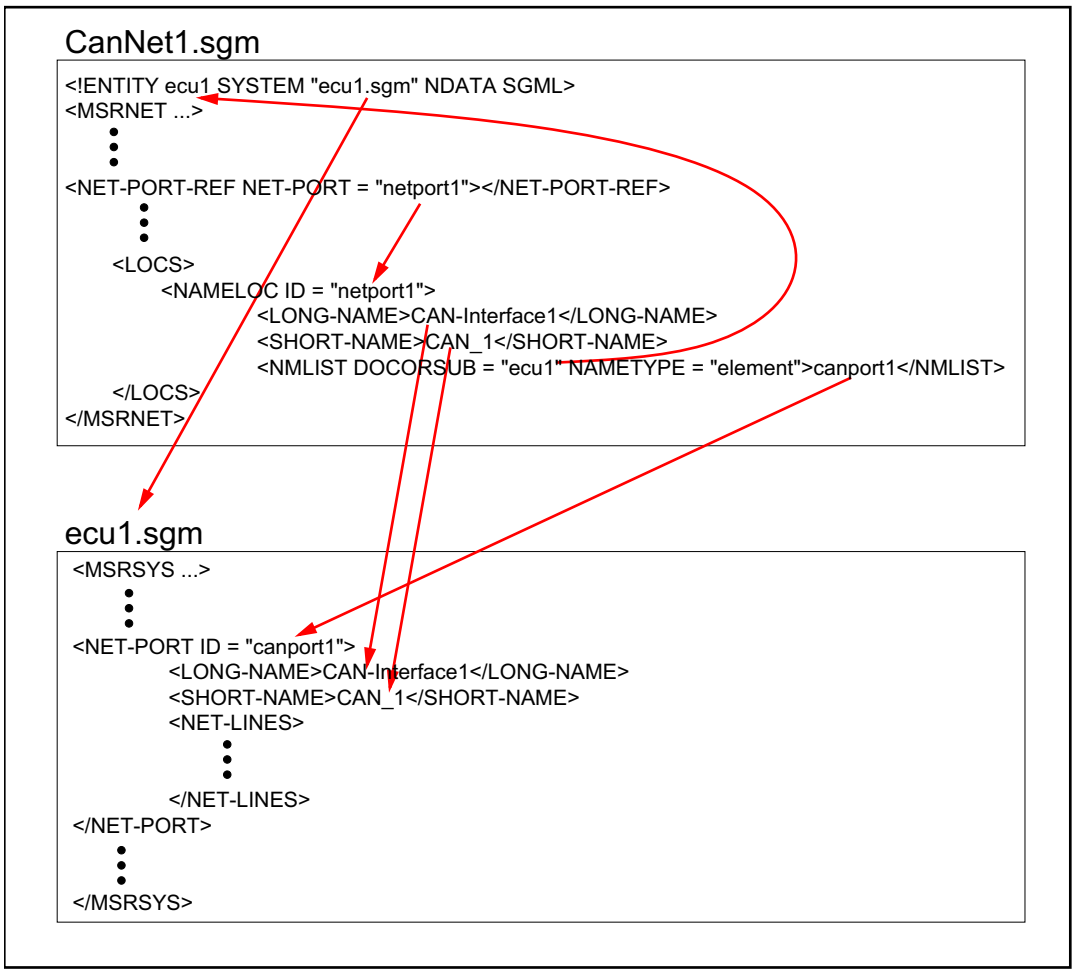


Figure 8: Linking with the NameLoc

The element **< net-port-ref >** refers to the element **< nameloc >** by refering to the ID of the element **< nameloc >**. The attribute **[docorsub]** of the element **< nmlist >** refers to the external entity ”ecu1”. This entity defines the system identifier (filename) of the entity. The content of the element **< nmlist >** refers to the ID of the resource within the external document. The **< long-name >** and the **< short-name >** of the **< nameloc >** element represents the **< long-name >** and the **< short-name >** of the element **< net-port >** of the resource.

6 SGML and XML support

In order to support both SGML and XML the following principles must be followed:

- no inclusion or exclusion exceptions (see see Chapter 3.11 building subsets of content models (Architectures) p. 13)
- Elements in DTD shall be uppercase (relevant only for XML DTS)
- And connector may not be used
- no minimization
- Support of iso-8859-1 encoding
- no marked sections in the instance
- no elements being declared empty. Such element lead to non wellformed instances which cannot be used as SGML and as XML files simultaneously.

6.1 Modularity - utilizing parameter entities

The individual partners can break down a complex interchange DTD into fragments. In order to simplify input or further processing, parts of these fragments are then removed and/or the company's own internal information (acquisition DTDs) is added.

It is recommended to set up particularly complex DTDs modularly, i.e. each module is one file. This way, these modules can easily be reused for further DTD definitions (for example, for defining an acquisition DTD). Maintenance of module elements occurs in just one place, however. Besides that, the grouping of elements, attributes and entities can be supported through the creation of modules.

Each MSR DTDs thus consists of several modules.

Each application must provide a tool specific between public identifier and the file name to facilitate parsing of the DTD, see the catalog file in the dtd distribution.

6.2 Configuring the DTDs

All MSR DTDs can be configured for different operation modes respectively for the requirements of different SGML-editors and SGML-browsers. The configuration mechanism is based on marked sections which switch on respectively switch off parts of the DTD structure. These marked sections are controled by entities which contain the marked section status keyword. The specifications of these entities are located in pairs within the msr *xxx* .dtd files. Each of this pairs is a dtd switch. To configure the DTD invert the terms of the pairs of entities of the corresponding switches in the msrsys.dtd, msrsw.dtd, msrnet.dtd or the msrrep.dtd to "IGNORE" respectively "INCLUDE".

Table 5: DTD Switches

DTD Switch Name	Description	Switch Entities	Default Value
linking element content	Configuration of the content of the linking elements(**< ...ref >**) The content is either "#PCDATA"(Default) or "EMPTY" This switch can be used for SGML-Tools which require the EMPTY linking elements.	linking-element-content-pcdata	INCLUDE
		linking-element-content-empty	IGNORE

Table 5 (Cont.): DTD Switches

DTD Switch Name	Description	Switch Entities	Default Value
linking element, locator attribute: REQUIRED or IMPLIED	Configuration of the locator attribute(IDREF) default value. The default value is either ”REQUIRED” (default) or ”IMPLIED” This switch can be used if a SGML Tool doesn’t use the standard ID/IDREF link mechanisms and the content of locator attribute will be generated by a separate process.	loc-attr-required	INCLUDE
		loc-attr-implied	IGNORE
resource element, ID attribute: REQUIRED or IMPLIED	Configuration of the ID-attribute default value.The default value is either ”REQUIRED” (default) or ”IMPLIED” This switch can be used if a SGML Tool doesn’t use the standard ID/IDREF link mechanisms and the content of ID-attribute will be generated by a separate process. This switch is also useful for SGML-Editors with link managers which create the IDs not at the moment of the element insertion, e.g. if the ID creation is a process which has to be started manually.	id-attr-required	INCLUDE
		id-attr-implied	IGNORE
linking element locator attribute type: IDREF or CDATA resource element, ID attribute type: IDREF or CDATA	Configuration of the ”declared value” of the linking element locator attributes and the resource element, ID-attributes. This switch can be used to configure a DTD for parsing instances with ID/IDRREF errors.	id-idref	INCLUDE
		no-id-idref	IGNORE
XML-Simple-Link at xref: YES or NO	This switch inserts additional attributes for xml-simple links at **< xref >**. This is only a configuration for testing purposes.	no-xml-link	INCLUDE
		xml-link	IGNORE
graphic filename ENTITY or CDATA	This switch configures the ”declared value” of the attribute **[filename]** of the element **< graphic >**. The value can be ”CDATA” (default) or ”ENTITY”. Some SGML-Tools can manage graphics only as entities.	graphic-filename-cdata	INCLUDE
		graphic-filename-entity	IGNORE
multilinuality : YES or NO	This switch can be used to configure the dtd for multilingual operation	not-multilingual	INCLUDE
		multilingual	IGNORE

Table 5 (Cont.): DTD Switches

DTD Switch Name	Description	Switch Entities	Default Value
viewer-info : YES or NO	This switch inserts additonal **< viewer-info >** elements in elements at chapter level respectively at topic level which have no **< long-name >** or **< label >** element. These elements can be used for the construction of a ”table of content” navigator in some SGML-Browsers.	no-viewer-info	INCLUDE
		viewer-info	IGNORE
desc,language,doc-revisions,param-detail-spec optional : YES or NO	This switch sets the occurance of the elements **< desc >**,**< language >**,**< doc-revisions >**,**< param-detail-spec >** to ”?” ([0..1] = optional). This switch can be used to parse instances which lack of these elements.	no-optional-elements	INCLUDE
		optional-elements	IGNORE
Softquad Author/ Editor Tables	This switch inserts additional elements within **< table >**. This switch is used to configure a DTD for SoftQuad Author/Editor SGML editor.	no-sqae	INCLUDE
		sqae	IGNORE

The MSR DTDs have standard configurations. Basis for the document interchange between companies are the corresponding standard DTD configurations.

6.3 SGML Declaration

The following **SGML features** of declaration were taken into account:

- FORMAL: The use of this mechanism makes sense, since the SGML parser is then able to check the structure of the ”public identifier” defined in the DTD.
- OMITTAG: The ”end-tag” is optional only for empty elements in the DTD. Other tag minimizations are not allowed in the interchange instance. This ensures that only normalized instances are interchanged.
- DATATAG, RANK, LINK, SUBDOC, SHORTTAG, CONCUR:

  Are not used since these features are only supported by a few tools.

The SGML declaration mainly serves to define the document character set and a mapping of the abstract syntax (defined in ISO 8879 Norm as ”reference concrete syntax”) onto the concrete syntax of the current SGML document and its DTD. That means that syntax specifications, memory requirements for the parser, base character sets and particular SGML features can be adapted according to the respective SGML application.

For MSR applications, an SGML declaration was defined which deviates from the reference values of Norm ISO 8879 in the following items:

Table 6: Reference value deviations in the SGML declaration

Set	Par	Reference Value (in Character)	MSR Value (in Character)
Capacity Set	TOTALCAP: Grand total of individual capacity points.	35000	175000
	GRPCAP: Token at any level in a content model.	35000	70000
	ATTCAP: Attribute defined.	35000	50000
Quantity Set	LITLEN: Length of a literal or delimited attribute value.	240	2048
	NAMELEN: Length of a name, name token, number, etc.	8	32
	ATTCNT: Number of attribute names and name tokens in an element's attribute definition.	40	80
	GRPCNT: Number of tokens in a group.	32	80
Delimiter Set	SHORTREF: Additional delimiter to the delimiter set.	SGMLREF	NONE

6.4 Character sets

SGML assumes that one relatively small **character set** will be used within an application. Any characters required in the document that are not in the basic character set are represented using combinations of those basic character set in constructs called character entities. So the SGML documents that are produced can be exchanged regardless of country or platform, the character entities must be entered in place of the current character code in the instance for character sets (ASCII-Code > 127). The conversion can be carried out during text input or in a subsequent conversion procedure.

To support this, the required character sets shall be defined in the DTD.

Since every SGML application must support the character sets defined in the MSR DTDs, only the most essential characters were used and defined as an MSR subset of the following international SGML character entity sets:

- ISOnum: PUBLIC "ISO 8879-1986//ENTITIES Numeric and Special Graphic//EN"
- ISOlat1: PUBLIC "ISO 8879-1986//ENTITIES Added Latin 1//EN"
- ISOtech: PUBLIC "ISO 8879-1986//ENTITIES General Technical//EN"
- ISOgrk3: PUBLIC "ISO 8879-1986//ENTITIES Greek Symbols//EN"

The MSR sdata entity file (*[External FILE: SDATA Entities / URL: msrsdata.ent]*) contains the exact definitions of the respective subsets.

7 Versioning the DTD

Each MSR DTD has a version label, which consists of the DTD name, 3 figures and an optional letter. The version label is composed as follows:

<DTD Name> V<Application Profile Number>.<Version Number>.<Revision Number>[<Patch Level>]

E.g. MSRSW V1.1.0b .

<

<DTD Name> The name of the DTD, e.g. MSRSYS

<Application Profile Number> Number of the MSR Application Profile. [0-9]+

Changes at elements and attributes, that belong to the MSR Application Profile lead to the increase of the application profile number.

<Version Number> Number of the version. [0-9]+

The version number will increased by changes, which are not compatible to old instances. I.e. old instances have to be converted.

The version number does not start again when the Application profile is increased. This may cause gaps in the numbering scheme:

MSRSW 1.1.0 -> MSRSW 2.2.0[10].

<Revision Number> Number of the revision [0-9]+

Changes, which are compatible to old instances lead to the increase of the revision number. I.e. old documents don't have to be converted.

<Patch Level> Letter of the patch level [a-z]

Changes, which are fully compatible to old instances and have no effect to document instances or changes, which serve to fix implementation bugs in the current revision, lead to an optional patch level label.

10

App. A Linking: General Definitions for linking

The following definitions apply to this document . Some definitions are the same as described in *XML Linking Language (XLink) ()*

link An explicit relationship between two or more data objects or portions of data objects.

resource An addressable unit of information that participates in a link. Concretely, anything reachable by the use of a locator in some linking element.

linking element An element that asserts the existence and describes the characteristics of a link. At MSR this is the origin of the link.

locator Data, provided as part of a link, which identifies a resource. Concretely, a character string in a linking element (e.g. URL, ID, query, concatenated pointer). MSR distinguishes between artificial locators and natural locators. An artificial locator identifies a resource by an ID which is unique in the document. The natural locator identifies a resource by giving information over the resource's properties (**< short-name >**). Examples for this are:

Identification of a control device plug pin by the device type part number, the plug name and the pin number.

Identification of a parameter value by the component number and the parameter short name.

Identification of a conversion formula by the formula short name.

traversal The action of using a link; that is, of accessing a resource. Traversal may be initiated by a user action (for example, clicking on the displayed content of a linking element or the displayed name of a target) or occur under program control.

link class A group of resources which can be linked together. E.g. the link class "prm" consists of a lot of parameters (**[f-id-class = "prm"]**) and the linking elements **< prm-ref >** and **< xref id-class = "prm" >**.

inline link Abstractly, a link which serves as one of its own resources. Concretely, a link where the content of the linking element serves as a participating resource. MSR XREF, HTML A, HyTime clink, and TEI XREF are all examples of inline links.

out-of-line link A link whose content does not serve as one of the link's participating resources . Such links presuppose a notion like extended link groups, which indicate to application software where to look for links. Out-of-line links are generally required for supporting multidirectional traversal and for allowing read-only resources to have outgoing links.

addressing Identification of a resource by specifying a locator.

App. B

MSRREP User's guide (msrrep-ug-de.chm)

MSRSW-tr-EADOC (msrsw-tr-eadoc.chm)

Documentadministration

Table : team members

Name	Company	
Herbert Klein	MSR-MEDOC Taskforce DTD	Department: XI-Works
Bernhard Weichel	MSR-MEDOC Taskforce DTD	Department: Robert Bosch GmbH

Table : version overview

Version	Date	Publisher	State
2	30.3.2001	Bernhard Weichel	WD

Table : modifications

Version	Change	Related to
2	Adopted to Version 2.x.x Reason: ASAM-2-MCD workshop	Content

Table : modifications included

Date	Chapter	Change	Related to
Nr. 1, 30.3.2001	Gesamt	Adopted to Version 2.x.x Reason: ASAM-2-MCD workshop	Content

References

Standards

Designation:	**[ISO/IEC 10744]:** Hypermedia/Time-based Structuring Language (HyTime)	
URL:		
Relevant Position:	Hyperlinks module	20

External Documents

Designation:	9502:1995 CALS Table Model DTD	15
URL:	http://www.oasis-open.org/html/a502.html	

Designation:	Benutzerhandbuch MSRREP	15
URL:	msrrep-ug-de.chm	

Designation:	Herbert Klein, Xiworks	6
URL:	mailto:herbert.klein@xiworks.de	

Designation:	in SGML Open Technical Memorandum TM 9502:1995.	6
URL:	http://www.oasis-open.org/html/a502.htm	

Designation:	MSR-Hilfesystem	12
URL:	msrrep-help.chm	

Designation:	MSRREP User's guide	30
URL:	msrrep-ug-de.chm	

Designation:	MSRSW-tr-EADOC	30
URL:	msrsw-tr-eadoc.chm	

Designation:	XML Linking Language (XLink)	
Document number:	WD-xlink-19980303	
State:	Draft	
Date:	3-March-1998	
Publisher:	World Wide Web Consortium	
URL:		
Relevant Position:	1.3 Terminology	29

External FILES

Designation:	SDATA Entities	27
URL:	msrsdata.ent	
Format:	SGML	
Tools:	-	
Version:	-	

Index

Technical Terms

OTHER

Symbols

Products

Symbols

SGML Attributes

Symbols

SGML Elements

Symbols