iNEWS NRCS

Engineering Specification News Story Markup Language 2.7

Version 2.7

27 August 2002

NSML – A Markup Language for News Stories Version 2.7

1. Introduction

The purpose of this document is to define the News Story Markup Language (NSML). NSML is an SGML based markup which is used to express all the content and information about a story. It also can describe a *form* that can be used to describe the presentation of fields when displayed.

NSML is used to express four separate aspects of a story. NSML expresses meta information about a story in the `<head>` section; it expresses the story content in the `<story>` section; it expresses the appearance of the story when displayed in the `<look>` section; and it expresses arbitrary attachments in the `<field-atts> and <aeset-atts>` sections.

2. NSML Grammar

The elements of a story are identified with NSML tags. These tags are organized in a hierarchy. At the top of the hierarchy is the `<nsml>` tag which delimits a complete NSML document and contains the story meta information, story content, story presentation, and attachment elements. The other elements of a story are nested inside these elements as described by the NSML grammar.

The story grammar defines the ordering and nesting of the tags and story elements. There are two sets of language elements: terminal – the actual words in the NSML language – and non-terminal – everything else. The terminals are represented in the `Courier` type face, non-terminals aren't. There is one predefined non-terminal, *plain_text*. ***Plain_text*** is defined to be regular characters, and character entities denoted by the ampersand character. A **CDATA section** can be used to encapsulate *plain_text* that contains many NSML markup characters that would otherwise be converted into character entities.

Elements inside square brackets are optional. Elements inside curly brackets are repeatable zero or more times.

```
nsml_tag    ::= <nsml>
                    [ head_tag ]
                    [ look_tag ]
                    [ story_tag ]
                    [ field_atts_tag ]
                    [ aeset_atts_tag ]
                </nsml>

head_tag    ::= <head>
                    [ <meta> ]
                    [ <storyid> plain_text </storyid> ]
                    [ <formname> plain_text </formname> ]
                    [ <rgroup> plain_text </rgroup> ]
                    [ <wgroup> plain_text </wgroup> ]
                    [ <source> plain_text </source> ]
                    [ <dist> plain_text </dist> ]
                    [ <wcode> plain_text </wcode> ]
                </head>

look_tag::= <look>
                    { form_tag }
            </look>
```

```
story_tag      ::= <story>
                      [ fields_tag ]
                      [ body_tag ]
                      [ aeset_tag ]
                   </story>

field_atts_tag     ::= <field-atts>
                      { attachment_tag }
                   </field-atts>

aeset_atts_tag     ::= <aeset-atts>
                      { attachment_tag }
                   </aeset-atts>

form_tag       ::= <form>
                      { row_tag }
                   </form>

row_tag        ::= <row>
                      { cell_tag }
                   </row>

cell_tag       ::= <cell>
                      [ plain_text ]
                   </cell>

fields_tag     ::= <fields>
                      { f_tag }
                   </fields>

f_tag          ::= <f>
                      plain_text
                   </f>

body_tag       ::= <body>
                      { p_tag }
                   </body>

p_tag          ::= <p>
                      { p_content }
                   </p>

p_content      ::= content_tag
               |   text

content_tag    ::= <pi> text </pi>
               |   <cc> text </cc>

text           ::= { text_content }

text_content   ::= <pb>
               |   <tab>
               |   <wp>
               |   <a>
```

```
                |     style_tag
                |     plain_text

style_tag       ::=   <b> text </b>
                |     <i> text </i>
                |     <u> text </u>

aeset_tag       ::=   <aeset>
                          { ae_tag }
                      </aeset>

ae_tag          ::=   <ae>
                          first_ap_tag
                          { ae_content }
                      </ae>

ae_content      ::=   <img>
                |     mc_tag
                |     ap_tag

mc_tag          ::=   <mc>
                          [ mc_content ]
                          { ap_tag }
                      </mc>

mc_content      ::=   ae_token
                      [ mc_hidden_text ]
                      [ plain_text ]

first_ap_tag    ::=   <ap>
                          ap_hidden_text
                      </ap>

ap_tag          ::=   <ap>
                          ap_text
                      </ap>

ap_hidden_text        ::=  start_hidden_delim
                           space
                           ae_token{ae_token}
                           end_hidden_delim

mc_hidden_text        ::=  start_hidden_delim
                           space
                           ae_token{ae_token}
                           end_hidden_delim
                           space

ap_text         ::= { ap_content }

ap_content      ::=   <tab>
                |     <wp>
                |     astyle_tag
                |     plain_text
```

```
astyle_tag    ::=  <b> ap_text </b>
                |  <i> ap_text </i>
                |  <u> ap_text </u>

ae_token      ::=  plain_text without any internal space charaters, but at least one trailing space

start_hidden_delim    ::=  ]]

end_hidden_delim      ::=  [[

space         ::=  a single space charater (hex20)

attachment_tag   ::=  <attachment>
                         plain_text
                      </attachment>
```

3. NSML Basics

NSML defines a set of tags that are embedded in the story document. NSML tags are delimited by a set of angle brackets (< and >) and contain a tag name and optional tag attributes. For example the tag `<f id=title>` has a tag name of `f` and one attribute `id=title`. Each tag has its own set of attributes that may be applied to it. In the previous example the attribute name `id` is set to the attribute value `title`. There are two forms of attributes, the *`name=value`* form and *`value`* form. An example of the *`value`* form is `<meta hold>`, this tag has a name of `meta` and an attribute value of `hold`. In general, attributes are optional and defaults are assumed if they are not specified. There are some exceptions to this. The description of each tag in this document points out all of the required attributes.

3.1 Start and End tags

Most tags define and affect a specific part of an NSML document. This part of the document begins where the tag with its attributes (known as the start tag) appears in the document and continues to the corresponding end tag. The end tag has the same name as its corresponding start tag but starts with a `</` sequence rather than just a `<` sequence. For example, `</f>` is the `<f>` tag's end tag. End tags have no attributes. The part of the document enclosed in a set of start and end tags is referred to as the tag's content.

The number of characters in a tag, including the tag name and all attribute names and values but not including the tag delimiters, cannot exceed 500. This excludes comments and CDATA sections (described below).

End tags may be omitted when they may be implied by the surrounding context. In addition an NSML parser may infer the presence of some start tags if it can do so unambiguously. For example if *plain_text* is encountered after a *`body_tag`* it is safe for a parser to assume the presence of a *`p_tag`* since (as the NSML grammar shows) all *plain_text* within the body of the story must be contained within paragraphs.

3.2 Character set

The set of characters allowed in tag names and attribute names is the ASCII characters a-z, A-Z, 0-9, dot and dash. The first character of a name is always in the set a-z or A-Z. A name may have a maximum of 12 characters. The tag and attribute names are case-insensitive, that is <nsml>, <NSML> and <Nsml> are all equivalent.

Attribute values can be divided into three categories, TEXT, TOKEN and NUMBER values. A NUMBER value is a string of decimal digits from the ASCII set 0-9. A TOKEN value is a string of ASCII characters which has the same lexical constraints as tag and attribute names (same character set , same restrictions, case-insensitive, etc.) A TEXT value is a string of bytes in the range 0x20 to 0xff excluding the double quote character. TEXT values are case-sensitive. If a TEXT value contains characters not allowed in a TOKEN then it must be enclosed in double quotes. Values of type TEXT are limited to 100 bytes.

The characters allowed in *plain_text* content are undefined. The *plain_text* content is treated as a sequence of bytes. There are only two exceptions to this. When the *plain_text* content is parsed byte values in the range 0x00 to 0x1f are stripped from the content and character entity conversion is done on the *plain_text* content. The characters in the range 0x00 to 0x1f may be included in the *plain_text* content if represented as character entities.

Plain_text can be encapsulated in a **CDATA** section. **CDATA** sections are used to escape blocks of text containing characters that would otherwise be recognized as NSML markup. A **CDATA** section starts with the nine character sequence (**<![CDATA[**) and ends with the three character sequence (**]]>**). Within a **CDATA** section, only the end sequence (**]]>**) is recognized as markup, so that left and right angle brackets and ampersands may occur in the literal form; they need not be escaped using character entities. **CDATA** sections cannot nest. Byte values in the range 0x00 to 0x1f, except 0x09, 0x0A, and 0x0D will be stripped from **CDATA** sections.

3.3 Character Entities

Anywhere in *plain_text* characters may be replaced with character entities. This is useful when the *plain_text* contains NSML markup characters that could confuse an NSML parser. NSML recognizes the full set of numeric character entities. Numeric character entities are 1 to 3 decimal digits preceded by &# and followed by a semicolon, ';'. Leading zeros are ignored so &#062; is the same as &#62;. The named character entities that may appear in plain_text are identified in **table 2.1**.

Table 2.1 – Character Entities

Named Entity	Numeric Entity	symbol	Description
&gt;	`&#062;`	>	Greater than
&lt;	`&#060;`	<	Less than
&amp;	`&#038;`	&	Ampersand

3.4 Comments

Comments may be included anywhere (outside of tags) within an NSML document. Comments can only be seen when viewing the raw form of the document. In almost all cases comments should be hidden from the user's view. It is allowable for applications that process a document to strip the comments from the processed document. A comment is any characters between the four character start , "`<!--`" and the four character end " `-->`" comment markers (i.e. `<!-- Comment Text Here  -->`). No translation of comment text is performed. This means that the characters '&', '<', and '>' will be stored in the NSML document unchanged. Comments cannot be nested - the first " -->" encountered terminates the comment.

For details on other text that is not intended for display, see sections 7.6.8 and 7.6.9

3.5 Base Units

Many of the attribute values are measurements expressed in *base units*. This measurement is equal to one-fourth of the average character width of the "system font". This unit is based on the Microsoft Windows definition of its *Dialog Base Unit*. This measurement is device independent, so that applications can use a single measurement to create a similar appearance on different types of display devices or different font sizes.

4. NSML top level elements

4.1 The <nsml> tag

<nsml>

Function:
Delimits a complete NSML document.
Attributes:
VERSION
End tag:
</nsml>; optional
Contains:
head_tag; optional
look_tag; optional
story_tag; optional
field_atts_tag; optional
aeset_atts_tag; optional
Used in:
Nothing

The `<nsml>` and `</nsml>` tags enclose the entire NSML document. This tag is not required since any document parser that knows NSML can infer from the enclosed content that it is NSML source.

The *version* attribute is optional. Its value is TEXT that defines the NSML version used to compose a document. If included it must read:
version=”-//iNEWS//DTD NSML 2.7//EN”

If not specified an NSML parser is to assume the above version by default.

The version value must be quoted in this case because it contains slash and space characters that are not allowed in unquoted attribute values.

4.2 The <head> tag – define information about the story

<head>

Function:
Defines information about a story.
Attributes:
None
End tag:
</head>; optional
Contains:
meta_tag; optional
head_content; optional
Used in:
nsml_tag

The <head> and </head> tags delimit the story meta information. Meta information is information about a story or attached to a story that is not strictly content.

4.3 The <look> tag

<look>

Function:
Delimit a set of story forms.
Attributes:
None
End tag:
</look>; optional
Contains:
form_tag; optional
Used in:
nsml_tag

The <look> and </look> tags delimit a set of forms associated with the story. This tag is a container for information that can alter the appearance but not the meaning of the story content.

4.4 The <story> tag – Defining story content

<story>

Function:
Delimits the story content.
Attributes:
None
End tag:
</story>; optional
Contains:
fields_tag; optional
body_tag; optional
aeset_tag; optional
Used in:
nsml_tag

The `<story>` and `</story>` tags enclose a story content description.

4.5 The <field-atts> tag – Defining field attachments

<field-atts>

Function:
Delimits the field attachments.
Attributes:
None
End tag:
</field-atts>; optional
Contains:
attachment_tag; optional
Used in:
nsml_tag

The `<field-atts>` and `</field-atts>` tags enclose field attachments.

4.6 The <aeset-atts> tag – Defining anchored element set attachments

<aeset-atts>

Function:
Delimits the anchored element set attachments.
Attributes:
None
End tag:
</aeset-atts>; optional
Contains:
attachment_tag; optional
Used in:
nsml_tag

The `<aeset-atts>` and `</aeset-atts>` tags enclose anchored element attachments.

5. Story Header Definition

5.1 The <meta> tag – Define story meta information

<meta>

Function:
Defines meta information about a story.
Attributes:
WIRE
MAIL
LOCKED
WORDS
RATE
BREAK
MCSERROR
HOLD
FLOAT
DELETE
End tag:
None
Contains:
Nothing
Used in:
head_tag

The `<meta>` tag is used to express many miscellaneous attributes of a document. This tag has no content.

The *wire*= attribute is optional. If present it indicates that the document is an original, unmodified story received from a wire service provider. Its value is a TOKEN that indicates the priority of the wire and must be one of *f* (flash), *b* (bulletin), *u* (urgent), *r* (routine), or *o* (other). If not present, the document is not a wire story.

The *mail*= attribute is optional. If present it indicates that the story is an original, unmodified document received as a mail message. Its value is a TOKEN that indicates whether the receiver of the mail message has viewed it and must be one of *read* or *unread*. If not present, the document is not a mail message.

The *locked*= attribute is optional. If present it indicates that a user has locked the document. A user may restrict access to a document by locking it in one of two ways. Its value is a TOKEN and must be one of *pass* or *user.* If the value is *pass* then access is granted to users that can supply the correct password. If the value is *user* then access is granted only to the user who locked the document. If not present, the document is not locked.

The *words*= attribute is optional. Its value is a decimal NUMBER that, if present, specifies the number of countable words in the body of the document. Countable words are those words that contribute to the audio read time of a story. If not present, the number of words is assumed to be zero.

The *rate* attribute is optional. Its value is a decimal NUMBER that, if present, specifies the read rate to be used to calculate the audio read time. The read rate is in words per minute. If not present, the rate is assumed to be zero.

The *break* attribute is optional. It is a TOKEN which, if present, indicates that the document is a story in a rundown which marks a break or divider between segments of the rundown.

The *mcserror* attribute is optional. It is a TOKEN which, if present, indicates that the document is a story in a rundown which contains machine control instructions and at least one of the instructions cannot be understood by the machine control subsystem.

The *hold* attribute is optional. It is a TOKEN which, if present, indicates that the document is not eligible for automatic purge. A user will place a hold on a document that he wants to protect from the automatic data base story purge policy.

The *float* attribute is optional. It is a TOKEN which, if present, indicates that the document is a story in a rundown which has been "floated" by a user. A floating story deserves special treatment in the rundown because it is not yet included as a story to go on-air.

The *delete* attribute is optional. It is a TOKEN which, if present, indicates that the document has been deleted from the database.

5.2 Head content tags

<rgroup> <wgroup> <source> <dist> <wcode> <formname> <storyid>

Function:
Defines meta data attached to a story.
Attributes:

NUMBER	[RGROUP AND WGROUP ONLY]

End tag:
</rgroup> </wgroup> </source> </dist> </wcode> </formname> </storyid>; optional
Contains:
plain_text
Used in:
head_tag

The head content tags express miscellaneous information attached to or about a document.

The `<rgroup>` and `<wgroup>` tags are optional. They contain the read group and write group names, respectively, assigned to the story. These tags have a single optional attribute - *number*= - whose value is a decimal NUMBER which is the internal group number associated with the group name. It is valid to omit the group name and end tag and just specify the group number as in <head><wgroup number=20></head>. If not specified the document has no groups assigned to it.

The `<source>` tag is optional. If specified it is the name of the News DataBase Server that generated the document.

The `<dist>` tag is optional. If specified it is a distribution code that a user has attached to a document.

The `<wcode>` tag is optional. If specified the document originated as a wire story from a wire service provider and `<wcode>` is the wire distribution codes assigned to the wire when it was originally received. Not all wire story documents will have a `<wcode>` tag.

The `<storyid>` is an identification string used by the News DataBase Server to locate a story. The actual content of the `<storyid>` is not specified in NSML.

The `<formname>` content is an identifier associated with the form that was used to create the first instance of the story.

6. Story Presentation

6.1 The <form> tag – define the appearance of the story fields

<form>

Function:
Defines a presentation/appearance of the fields.
Attributes:
VT | GI
STYLE
ALIGN
RO
LLEFT | LRIGHT | LTOP | LBOTTOM
LSTYLE
LALIGN
End tag:
</form>; optional
Contains:
row_tag; optional
Used in:
look_tag

The `<form>` and `</form>` tags enclose a presentation description for the fields of a story. It does not contain any story content, all story content is defined within the `<story>` tag. A *form* is a definition of the layout of the story fields on a "page" when viewed. An NSML document need not contain a form definition. The presentation of the story is the responsibility of the application displaying the document that must acquire a form.

The *vt* and *gi* attributes are optional. These are TOKENS that identify the *form* as having been designed for use on a graphical interface or a video terminal. If not present *gi* is assumed. These are mutually exclusive attributes.

All other attributes are optional and are a subset of the attributes of the *row_tag*. The *form_tag* contains a group of *row_tags* that inherit the values of these attributes. See the description of these attributes under the *row_tag* description.

6.2 The <row> tag

<row>

Function:
Defines a presentation/appearance of a row of fields.
Attributes:
STYLE
ALIGN
RO
LLEFT | LRIGHT | LTOP | LBOTTOM
LSTYLE
LALIGN
End tag:
</row>; optional
Contains:
cell_tag; optional
Used in:
form_tag

The `<row>` and `</row>` tags group a set of *cell_tags* which are to appear in order on the same row when displayed.

All the attributes are optional and are a subset of the attributes of the *cell_tag*. The *row_tag* contains a group of *cell_tags* that inherit the values of these attributes. See the description of these attributes under the *cell_tag* description.

6.3 The <cell> tag

<cell>

Function:
Defines a presentation/appearance of a row of fields.
Attributes:
IDREF
LENGTH
STYLE
ALIGN
RO
LLENGTH
LSTYLE
LALIGN
LLEFT | LRIGHT | LTOP | LBOTTOM
AREADY
End tag:
</cell>; optional
Contains:
plain_text; optional
Used in:
row_tag

The `<cell>` tag is used to define a presentation area for a single field of a document.

The *idref*= attribute is optional. Its value is a TOKEN that identifies the specific field content from the `<fields>` section which is to be displayed in this cell. If not specified the cell is assumed to be blank and read-only.

A cell that references a field can inherit some behavior that is associated with the field's id. For example, a cell that references the TOTAL-TIME field will be read-only and must contain the sum of the *TAPE-TIME* field and the story's AUDIO-TIME field. This behavior is an implied attribute of the field with *id=TOTAL-TIME*.

The *ro* attribute is optional. A cell is either read-write or read-only depending on the behavior inherited from the referenced field. A cell than includes this token overrides the read-write access of the referenced field. A cell presenting a field that is read-only can not force the cell to read-write.

The *length*= attribute is optional. Its value is a decimal NUMBER that, if present, defines the length, in base units, of the cell. If not specified the length of the cell is assumed to be of zero.

The *style*= attribute is optional. Its value is a TOKEN that, if specified, defines the physical style of the field content displayed in the cell. If not specified the cell inherits a style from the enclosing `<row>` or `<form>` tags. If no style is specified in either the `<cell>`, `<from>` or `<row>` then the style defaults to normal. The style value may be one of b (bold), i (italic), or u (underline). It is possible to combine styles by specifying a quoted, space separated list as the value. For example *style="b u"* specifies a style of bold-underline. To specify a normal style, *style=""* may be used.

The *align*= attribute is optional. Its value is a TOKEN that, if specified, defines the alignment of the field content displayed in the cell. The value may be one of *left*, *right*, or *center*. If not specified, *left* alignment is assumed.

The *aready* attribute is optional. It is a TOKEN which, if specified, indicates that the field referenced by the cell via the *idref*= attribute is to acquire the *aready* attribute when a new story is created.

The content of a *cell_tag* is the text of the cell label. Each cell may have one label, the label appears in its own presentation area either to the left, right, top or bottom of the cell. The *cell_tag* has some attributes that affect the appearance of the label.

The *ltop, lright, lleft, and lbottom* attributes are optional. These are TOKENS that, if specified, determines the placement of the label relative to the cell. Only one of these values may appear in the *cell_tag*. If not specified *ltop* is assumed.

The *llength*= attribute is optional. Its value is a decimal NUMBER that, if present, defines the length, in base units, of the cell label. If not specified the length of the cell label is assumed to be of zero.

The *lstyle*= attribute is optional. Its value is a TOKEN that, if specified, defines the physical style of the cell label. If not specified the cell label inherits a style from the enclosing `<row>` or `<form>` tags. The *lstyle* attribute value has the same form as the *style* attribute value.

The *lalign*= attribute is optional. Its value is a TOKEN that, if specified, defines the alignment of the cell label. The value may be one of *left*, *right*, or *center*. If not specified, *left* alignment is assumed.

7. Story Content Definition

7.1 The <fields> tag

<fields>

Function:
Delimits the story fields content.
Attributes:
None
End tag:
</fields>; optional
Contains:
f_tag; optional.
Used in:
story_tag

The `<fields>` and `</fields>` tags enclose all the story fields.

7.2 The <body> tag

<body>

Function:
Defines the story text content.
Attributes:
TABS
SCRIPT
WIDTH
PINDENT
RINDENT
FINDENT
End tag:
</body>; optional
Contains:
p_tag; optional.
Used in:
story_tag

The `<body>` and `</body>` tags enclose the text of the story. All text must be within paragraphs in the body element.

The *tabs*= attribute is optional. Its value is a list of decimal NUMBERS that, if specified, defines the tab stop positions for all `<tabs>` within the text. Each number in the list is the distance from the last stop. The last value in the list may by a dash, this indicates that the last number in the list is to repeat indefinitely. The numbers are in base units. For example, *tabs=”40* `-`“ may be specified to place stops at every 40 base units. If not specified *tabs=24* is assumed. If more than one number is specified in the list, the numbers are separated by spaces and the list must be quoted (i.e. tabs=”72 144”).

The *script*= and *width*= attributes are optional. They are used to preserve the margin settings used for word wrapping by the last application that modified the story. The *width* value is a decimal NUMBER that specifies the width, in base units, used to word wrap the text contained in the `<body>` tag. The *script* value is a decimal NUMBER that specifies the width, in base units, used to word wrap the text in the anchored elements. If the *script* attribute is not present this indicates the story is not scripted. If the *width* attribute is not present the application displaying the body text must choose an appropriate width. See description of related `<wp>` tag.

The *pindent*= attribute is optional. Its value is a decimal NUMBER that defines the left paragraph indent for all paragraphs in the body of the story. The indent value is expressed in base units from the left. If not set it is assumed to be zero.

The r*indent*= attribute is optional. Its value is a decimal NUMBER that defines the right paragraph indent for all paragraphs in the body of the story. The indent value is expressed in base units from the right. If not set it is assumed to be zero.

The *findent*= attribute is optional. Its value is a decimal NUMBER that defines the indent for the first line of all paragraphs in the body of the story. The indent value is expressed in base units from the left. If not set it is assumed to be the same as *pindent*.

7.3 The <aeset> tag

<aeset>

Function:
Defines the set of anchored elements within the story.
Attributes:
None
End tag:
</aeset>; optional
Contains:
ae_tag; optional.
Used in:
story_tag

The `<aeset>` and `</aeset>` tags enclose the set of anchored elements within a story.

7.4 Story Fields

7.4.1 The <f> tag – defining a field

<f>

Function:
Defines a field content.
Attributes:

ID	[REQUIRED]
URGENCY	
AREADY	
UEC	

End tag:
</f>; optional
Contains:
plain_text; optional.
Used in:
fields_tag

The `<f>` and `</f>` tags enclose *plain_text* which is uniquely associated with an identifier.

The *id*= attribute is required. Its value is a TOKEN that uniquely identifies the field in the story so that it may be referenced and used. The *id* is referenced by the *idref*= attribute of the *cell_tag* within the `<form>` element of the NSML document. Each story will have only one field with a specific *id*. There are some reserved *ids* that identify fields with specific meaning and in some cases contain system-supplied content.

Reserved field identifiers

AIR-DATE	decimal digits	Seconds since Jan 1, 1970 00:00:00 GMT
AUDIO-TIME	decimal digits	Audio read time of story in seconds. Normally based on read-rate and word-count but can be user entered.
BACK-TIME	special•	Hard in-time of the story in seconds
CA-CAPTURED	character string	Connect session information – number of characters captured.
CA-DIRECTION	character string	Connect session information – direction of connection (in or out)
CA-ELAPSED	character string	Connect session information – duration as HH:MM:SS
CA-IDENT	character string	Connect session information – session identifier
CA-ORIGIN	character string	Connect session information – originating computer name
CA-RECEIVED	character string	Connect session information – number of characters received
CA-REMOTE	character string	Connect session information – remote computer name
CA-SENT	character string	Connect session information – number of characters sent
CG-ADDR	character string	MCS / BCS – CG device address
CG-TEMPLATE	character string	MCS / BCS – CG template address
CG-TEXT	character string	MCS / BCS – CG text
CHANNEL	character string	MCS / BCS – generic event channel
CREATE-BY	character string	User name of creator of first version of the story
CREATE-DATE	decimal digits	Seconds since Jan 1, 1970 00:00:00 GMT
CUME-TIME	special•	Hard out-time of the story in seconds
DEVICE-MGR	character string	MCS / BCS – device manager name
DURATION	character string	MCS / BCS – generic event duration
EFFECT	character string	MCS / BCS – event effect
ENDORSE-BY	character string	Name of user who endorsed the story
EVENT-STATUS	character string	MCS / BCS – (video) event status
LINE-COUNT	decimal digits	Number of lines
MAIL-CC	character string	Mail information – cc addressee list
MAIL-TO	character string	Mail information – to addressee list
MODIFY-BY	character string	User name of the last modifier of the story
MODIFY-DATE	decimal digits	Seconds since Jan 1, 1970 00:00:00 GMT
MODIFY-DEV	character string	Device name on which story was last modified
MOS-ACTIVE	character string	MOS – event identifier
MOS-SUBEVENT	character string	MOS – event details
MOS-TITLE	character string	MOS -- event description
PAGE-NUMBER	character string	User entered story identifier
PRESENTER	character string	The name of the person who will read the story on-air
READY	character string	State of the story. Either READY or ?.
RESULT-INDEX	character string	Search result information – story indentifier
RESULT-LOC	character string	Search result information – story location (queue name)
SEARCH-ID	character string	Search result information – search request identifier
STATUS	character string	Status of some element of the story.
STILL-ID	character string	MCS / BCS – still store event identifier
STILL-PRESET	character string	MCS / BCS – still store event preset
STYLE	character string	MCS / BCS – event style
TAPE-TIME	decimal digits	The run time in seconds of a tape to be played with the story
TITLE	character string	User entered story title
TOTAL-TIME	decimal digits	Total story time in seconds, sum of audio-time and tape-time
VIDEO-ID	character string	MCS / BCS – video (tape / clip) identifier

WRITER	character string	User name of the writer taking credit for the story.

•The back-time and cume-time field content has a special encoding. The hard in/out times are expressed in seconds as either a relative time or an absolute time (a.k.a. time of day.) The time is assumed to be relative unless the first character of the field is a @ character. For example a back-time content of 600 specifies a hard in-time of 10 minutes relative to the start of the show. A back-time content of @600 specifies a hard in-time of 12:10:00 am (10 minutes passed midnight.)

The *uec* attribute is optional. It is a TOKEN which, if present, indicates that a user has entered content to override the system supplied content normally provided in the field.

The *urgency*= attribute is optional. Its value may be a NUMBER from the set 1, 2, or 3 (if not specified 1 is assumed.) It is intended as a clue to the applications that the data in this field has some exceptional meaning and may need to be brought to the users attention. The specific method for presenting this information to the user is not defined by NSML. It could be ignored. The specific meaning of the *urgency* is dependent on the specific *id*.

The urgency attribute can be applied to the STATUS field to allow for functionality available in NetStation V14. The STATUS field contains strings from the MCS device drivers to indicate status of tape events (like PLAYING, OFFLINE, ERROR). If the tape is OFFLINE or an error has NetStation can highlight the field in some way to draw the user's attention. In NSML, MCS will put the string into the STATUS field and set the urgency flag as appropriate.

An application presenting a STATUS field with an urgency of 2 or 3 should draw the user's attention to this status because this indicates that a video clip used by the story is not ready for air. No specific meanings have been assigned to urgencies on other fields at this time.

The *aready* flag is optional. It is a TOKEN which, if specified, indicates that the content of the field affects the content of the READY field. Specifically, if any field with the *aready* attribute has a ? (question mark) as the first character of its content or it has no content, the READY field will have a ? as its content.

7.5 Story Body

7.5.1 The <p> tag – body paragraph blocks

The `<p>` and `</p>` tags enclose the text of a paragraph. All text is inside paragraphs. The first `<p>` following the `<body>` tag is optional. The first appearance of *plain_text* in the body will infer its presence.

<p>

Function:
Defines a paragraph of text.
Attributes:
None
End tag:
</p>; optional
Contains:
content_tag; optional
text; optional.
Used in:
body_tag

7.5.2 Content-based Tags

Content-based tags attach a specific meaning, context or usage to the enclosed text. Applications parsing the NSML document can use these tags to do content based processing on the text.

Content-based tags may not be nested. If a new content-based tag is encountered before the end tag of the current tag, an end tag is assumed.

The physical style tags may be nested in the content-based tags to create highlighting effects. Applications are free to render nested styles in content-based text in any way that matches their capabilities. If an editing application is limited in its ability to present the styles to the user in a meaningful way it is free to strip the physical styles it cannot handle.

<pi> <cc>

Function:
  Define the contained *text* as having a specific meaning.
Attributes:
  None
End tag:
  </pi>, </cc>; optional
Contains:
  text
Used in:
  p_tag

The `<pi>` and `</pi>` tags enclose text which are instructions to the presenter reading the story on-air. This text is not included in the timing of the story. It is not intended for display on the closed caption device and is intended for display on the prompter.

The `<cc>` and `</cc>` tags enclose text which is not read by the presenter on-air but is usually part of a package voice-over which is to be closed captioned. This text is not included in the timing of the story. It is not intended for display on the prompter.

7.5.3 Physical Style Text Tags

The physical style tags enclose text that is to be enhanced when presented to the user. Physical style tags may be nested within other physical style tags to combine highlighting effects. Applications are free to render styles in any way that matches their capabilities but the standard or preferred renderings are <b> to bold, <i> to italic and <u> to underline. If an editing application is limited in its ability to present the styles to the user in a meaningful way it is free to strip the combinations of styles it cannot handle.

<b> <i> <u>

Function:
  Specify a physical style (appearance) for the contained *text.*
Attributes:
  None
End tag:
  </b>, </i>, </u>; optional
Contains:
  text
Used in:
  text

These tags are unlike the content-based tags in that they do not identify the enclosed content, they are just available to the user to highlight words to improve readability.

7.5.4 Spacing and Layout Text Tags

<pb> <tab> <wp>

Function:
Indicates spacing and layout for text.
Attributes:
None
End tag:
None
Contains:
Nothing.
Used in:
text

The `<pb>` tag indicates that when printing a story a page break is to occur at that position in the text.

The `<tab>` tag indicates that the next character is to start at the next tab stop position.

The `<wp>` tags will appear at the positions in the text where the word wrapping was performed by the last application to modify the text. These wrap points may be used by an application if it wishes to present the text of the story with the same appearance as it last appeared.

7.5.5 The <a> tag – anchor to anchored elements

<a>

Function:
Inserts a reference to an anchored element.
Attributes:

IDREF	[REQUIRED]

End tag:
</a>; optional
Contains:
Nothing.
Used in:
p_tag

The `<a>` tag marks the spot within a paragraph of the document body at which an anchored element is referenced. The order and position of these *anchors* are intended to determine the order and position of the anchored elements when displayed by an application.

The *idref*= attribute is required. Its value is a TOKEN that identifies a specific anchored element in the `<aeset>` that is referenced by the *a_tag*.

7.6 Story Anchored Element Set

7.6.1 The <ae> tag – anchored element

<ae>

Function:

Defines an anchored element.

Attributes:

ID [REQUIRED]

End tag:

</ae>; optional

Contains:

first_ap_tag

ae_content; optional

Used in:

body_tag

The `<ae>` and `</ae>` tags enclose an anchored element. An anchored element is an object that is anchored at a specific position in the body of the document. How these objects are displayed is dependent on the object content and the application displaying the object. Anchored elements are referenced by anchors (*a_tags*) within body paragraphs.

The *id*= attribute is required. Its value is a TOKEN that uniquely identifies the anchored element in the story so that it may be referenced and used. The *id* is referenced by the *idref*= attribute of the *a_tag* within the `<body>` element of the NSML document. An *id* will appear only once within an anchored element set (`<aeset>`).

Must start with a *first_ap_tag* followed by zero or more repetitions of *ae_content.*

7.6.2 Anchored element content – *ae_content*

ae_content

Function:

Defines the optional content elements in an *ae_tag*.

Start tag:

None

Attributes:

None

End tag:

None

Contains:

mc_tag; optional

ap_tag; optional.

img_tag; optional

Used in:

ae_tag

May contain *mc_tag*, *ap_tag*, or *img_tag*.

7.6.3 The <img> tag – images

<img>

Function:
Display an image.
Attributes:

HREF	[REQUIRED]
ALT	
TITLE	

End tag:
None
Contains:
Nothing.
Used in:
ae_tag

The <img> tag is used to insert an image into an anchored element. This tag references the image via its *href* attribute.

The *href*= attribute is required. Its value is TEXT that specifies a Uniform Resource Locator (URL) that references the image to be displayed.

The *alt*= attribute is optional. Its value is TEXT that is to be displayed in the event that the referenced image can not be located.

The *title*= attribute is optional. Its value is TEXT that is to be displayed near the image. It is a title or some other descriptive text associated with the image.

7.6.4 The <mc> tag – machine control tag

<mc>

Function:
Defines a set of instructions used for machine control.
Attributes:
ERROR
End tag:
</mc>; optional
Contains:
mc_content; optional.
ap_tag; optional.
Used in:
ae_tag

The `<mc>` and `</mc>` tags enclose machine control instructions which are understood by Machine Control Subsystem of the News System.

The *error* attribute is optional. It is a TOKEN that, if specified, indicates that the News System's Machine Control Subsystem was not able to understand or carry out the instructions contained within the tag.

Starts with zero or one *mc_content* followed by zero or more *ap_tag*.

7.6.5 Machine control content - *mc_content*

mc_content

Function:
Defines a machine control instruction, that can contain *mc_hidden_text*.
Start tag:
None
Attributes:
None
End tag:
None
Contains:
ae_token
mc_hidden_text; optional.
plain_text; optional.
Used in:
mc_tag

Starts with an *ae_token* followed by zero or one *mc_hidden_text* and zero or one *plain_text*.

7.6.6 First <ap> tag – *first_ap_tag* – The first anchored paragraph block

<ap>

Function:
Defines a first paragraph in anchored element. This is a special case of *ap_tag* (see below).
Attributes:
None
End tag:
</ap>; optional
Contains:
ap_hidden_text
Used in:
ae_tag
Also see:
ap_tag (below)

The first paragraph in an anchored element must contain only a single instance of *ap_hidden_text*.

7.6.7 Non-first <ap> tag – *ap_tag* – anchored paragraphs block

<ap>

Function:
Defines a paragraph (after the first paragraph) in anchored element.
Attributes:
None
End tag:
</ap>; optional
Contains:
ap_text; optional.
Used in:
ae_tag
Also see:
first_ap_tag (above)

The `<ap>` and `</ap>` tags enclose the text of a paragraph in an anchored element. *Ap_text* is differentiated from *text* because *ap_text* cannot contain `<pb>` or `<a>` tags.

7.6.8 First anchored paragraph hidden test - *ap_hidden_text*

ap_hidden_text

Function:
Defines hidden text that appears in the first paragraph of an anchored element.
Start tag:
None
Attributes:
None
End tag:
None
Contains:
start_hidden_delim
space
ae_token{*ae_token*}
end_hidden_delim
Used in:
first_ap_tag

Starts with start*_hidden_delim*, followed by a *space*, followed by one or more *ae_token*, and closes with *end_hidden_delim*. The only syntactical difference between *mc_hidden_text* and *ap_hidden_text* is that *mc_hidden_text* has a *space* on the end and *ap_hidden_text* does not.

7.6.9 Machine control hidden test - *mc_hidden_text*

mc_hidden_text

Function:
Defines hidden text that can be included in a machine control instruction.
Start tag:
None
Attributes:
None
End tag:
None
Contains:
start_hidden_delim
space
ae_token{*ae_token*}
end_hidden_delim
space
Used in:
mc_content

Starts with *start_hidden_delim*, followed by a *space*, followed by one or more *ae_token*, and closes with an *end_hidden_delim* and a final *space*. The only syntactical difference between *mc_hidden_text* and *ap_hidden_text* is that *mc_hidden_text* has a *space* on the end and *ap_hidden_text* does not.

8. Field-atts and AEset-atts Content Definition

8.1 The <attachment> tag

<attachment>

Function:
Defines a sequence of text.
Attributes:
ID [REQUIRED]
End tag:
</attachment>; optional
Contains:
Plain_text; optional
Used in:
attachments_tag

The `<attachment>` and `</attachment>` tags enclose a sequence of text.

The *id*= attribute is required. Its value is a TOKEN that uniquely identifies the attachment in the story so that it may be referenced and used. It is beyond the scope of the NSML specification to define where references to the attachment reside or how they are used.

9. Example

<nsml>
<look><form gi>

```
<row lstyle="b i" ltop>
<cell idref=PAGE-NUMBER length=18 llength=18>PAG
<cell idref=PRESENTER length=18 llength=18 style=i>TAL
<cell idref=VAR-1 length=18 llength=18>CAM
<cell idref=VAR-2 length=30 llength=30>EFFX
<cell idref=TITLE length=102 llength=102>STORY
<cell idref=VAR-3 length=18 llength=18>WRT
<cell idref=VAR-4 length=48 llength=486>SOURCE
<cell idref=APP1-1 length=30 llength=30>TAPE #
<cell idref=AFF-READY-1 length=48 llength=48 aready>STATUS
<cell idref=AUDIO-TIME length=30 llength=30>TIME
<cell idref=BACK-TIME length=36 llength=36>BKTIME
<cell idref=READY length=6 llength=6 ro>?
<cell idref=VAR-5 length=6 llength=6 style=i>P
<row lleft lstyle=i style=u>
<cell idref=MODIFY-DATE length=126 llength=60 ro>Modified:
<cell idref=MODIFY-BY length=48 llength=24 ro>By:
<cell idref=VAR-6 length=144 llength=60>Comments:
</form></look>
<story><fields>
<f id=PAGE-NUMBER>75
<f id=PRESENTER>SB*
<f id=VAR-1>C 2
<f id=VAR-2>2 SHT
<f id=TITLE>TEASE/BREAK
<f id=VAR-3>KG*
<f id=VAR-4>********
<f id=AFF-READY-1 aready>
<f id=AUDIO-TIME uec>130
<f id=READY>R
<f id=VAR-5>*
<f id=VAR-6>**
</fields>
<body script=186 width=234>
<p><a idref=1><a idref=2>
<p>         (BELL)
<p>
<p>     STILL AHEAD ON EYEWITNESS NEWS...
<p>
    THE MAXWELL CLUB HONORS SOME OF <wp>
OUR AREA'S BEST HIGH SCHOOL FOOTBALL <wp>
PLAYERS.  YOU'LL MEET THEM... COMING <wp>
UP.</p>
</body>
<aeset>
<ae id=1><ap>]] S1.5 G 0 [[</ap><ap><i>VO ENG</ap>
<ae id=2><ap>]] S1.5 G 0 [[</ap><mc><ap>cg2 Coming Up Next…</ae>
</aeset>
</story></nsml>
```