Hyper-G Text Format (HTF)

Version 2.12

Written by: Frank Kappe (fkappe@iicm.tu-graz.ac.at)
July 17, 1995

Contents

Contents

1 Introduction

This specification describes HTF, the Hyper-G Text Format. It should be read by authors and other persons who want to manually create text documents that are to be inserted in Hyper-G, and developers of Hyper-G components such as scripts that generate HTF markup from other sources, clients, gateways and other software that has to understand HTF.

1.1 Status of this document

The document you are currently looking at has been given version number 2.12. An up-to-date version is available in hypertext form on the Hyper-G server of Graz University of Technology. If you prefer reading linear text on paper, a PostScript version is available by anonymous ftp from *iicm.tu-graz.ac.at* in file *pub/Hyper-G/doc/HTF.ps*.

1.2 Glossary

This section is to define the meaning of a number of words that are used throughout this documentation.

- **anchor:** The endpoint of a link (see link). Spans a part of a document or the whole document.
- **client:** The part of Hyper-G that displays documents and realizes the user interface.
- **collection:** A group of items, which may be either documents or collections. Can be compared to "directory" in a file system.
- **document:** The smallest granule of information that can be displayed. In the context of this specification, we deal with text documents only.
- **Hyper-G:** A distributed hypermedia system that uses HTF as one possible document format.
- **link:** The essence of hypertext. A user may click on one endpoint of a link (the "source anchor") and jump to the other endpoint (the "destination anchor").
- **rendering:** The form of presentation of information to the human reader, in the client.
- **server:** The part of Hyper-G that stores the documents, links, etc.
- **web:** A number of documents that are connected by links.

2 References

[BC93] Tim Berners-Lee and Dan Conolly. Hypertext markup language (HTML). June 1993. Available on the WWW at URL `http://info.cern.ch/hypertext/WWW/Markup/HTML.html`.

[BCLN94] Tim Berners-Lee, Robert Cailliau, Ari Luotonen, Henrik Frystyk Nielsen, and Arthur Secret. The World-Wide Web. *Communications of the ACM*, 37(8):76–82, August 1994.

[Cla] James Clark. SGMLS. An SGML parser by James Clark <`jjc@jclark.com`> derived from the ARCSGML parser materials which were written by Charles F. Goldfarb. The source is available by anonymous ftp from `ifi.uio.no` in directory `/pub/SGML/SGMLS`.

[Gol90] Charles F. Goldfarb. *The SGML Handbook*. Oxford University Press, 1990.

[ISO86] ISO. *Information Processing Systems – Text and Office Systems – Standard Generalized Markup Language (SGML), ISO IS 8879.* ISO, 1986.

[ISO87] ISO. *Information Processing – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No. 1, ISO IS 8859-1.* ISO, February 1987.

[Kap93] Frank Kappe. Hyper-G: a distributed hypermedia system. In Barry Leiner, editor, *Proc. INET '93, San Francisco, California*, pages DCC–1–DCC–9, Internet Society, August 1993.

[KM93] Frank Kappe and Hermann Maurer. Hyper-G: a large universal hypermedia system and some spin-offs. *ACM Computer Graphics, experimental special online issue*, May 1993. Available by anonymous ftp from `siggraph.org` in directory `publications/May_93_online/Kappe.Maurer`.

[KPS93] Frank Kappe, Gerald Pani, and Florian Schnabel. The architecture of a massively distributed hypermedia system. *Internet Research: Electronic Networking Applications and Policy*, 3(1):10–24, Spring 1993.

3 HTF and the rest of the world

3.1 HTF and Hyper-G

HTF is closely related to Hyper-G in the following sense: HTF is the preferred text format for Hyper-G. That means that all current and future Hyper-G clients will support this format, even if other text formats will become available in some future versions of Hyper-G.

Hyper-G clients employ a generic SGML parser [Cla] that makes it possible to display any kind of SGML markup, provided there is a DTD and a style sheet. There is a stand-alone incarnation of the parser called *hgparse* that is part of the Hyper-G server distribution that allows to use the parser to verify the correctness of the SGML markup, and to convert it to other text formats, given an appropriate style sheet.

3.2 HTF and SGML

HTF is defined in terms of the ISO Standard Generalized Markup Language (SGML) [ISO86]. SGML is a sort of meta-standard for defining structured document types and markup languages to represent instances of those document types.

Every SGML document has three parts:

- An SGML declaration, which binds SGML processing quantities and syntax token names to specific values.
- A prologue including one or more document type declarations (DTD), which are formal grammars that specify the element types, element relationships and attributes, and references that can be represented by markup.
- An instance, which contains the data and markup of the document.

When we speak about an HTF document, we mean only the instance part of the SGML document. Only this part gets stored in the Hyper-G server.

3.3 HTF and HTML

HTML [BC93] is the markup language used for the World-Wide-Web (WWW) [BCLN94]. It is rather similar to HTF, although there are some substantial differences. It is possible, however, to translate HTF to HTML with only minimal loss of information. Because Hyper-G (and hence HTF) is designed to be used together with WWW clients this is an essential feature. Therefore, it is documented in this specification how the individual tags are translated.

3.4 Character Set

The base character set (the SGML BASESET) for HTF is ISO Latin-1 [ISO87]. This is the set referred to by any numeric character references. When an element is encoded using 7-bit characters (ASCII), then the mechanisms of character references and entity references may be used to encode characters in the upper half of the ISO Latin-1 set. In addition to the numeric character references, mnemonic entity names are defined (see Appendix A and the DTD in Appendix B).

4 An Introduction to SGML

In SGML, text is not specified in terms of its visual appearance (e.g. by some special code that specifies "switch to 15pt. font"). Rather, its semantic structure is specified (e.g. this is a heading), and it is left to the rendering process to actually layout (or render) the text (e.g. by displaying headings with 15pt fonts).

SGML is sort of a meta-standard that gives you a large amount of freedom to actually specify what your markup should look like. Consequently, a number of different, complex, and incompatible SGML markups have been proposed to be used within certain knowledge domains. As HTF is designed to be independent of a certain discipline, this would mean that HTF would have to be a superset of them in order to be able to transport and store them all using Hyper-G.

However, in the design of HTF a different approach was taken. Rather then making it a superset of all semantic SGML markups, HTF is a markup language that is primarily targeted towards the renderer and *NOT* so much interested in the semantics of the document. For example, there are a few markup elements in HTF that allow to specify that a phrase should be emphasized. HTF is not interested in the semantic meaning of the phrase, i.e. in all the possibilities *why* this phrase should be emphasized. This makes HTF a very small markup language that can be authored and processed easily. A large SGML DTD can be translated to the small HTF DTD by applying a sort of "pre-renderer" that maps the semantic elements to simple HTF elements.

SGML purists will argue that this is against the SGML philosophy and makes authoring HTF difficult because it is not possible to reconstruct the semantics of the document. However, Hyper-G is not designed as a document preparation system, but as a document dissemination system, where it is more important to support a wide range of users and user interfaces, even starting with simple terminals.

4.1 Tags

Every HTF element is started by a tag, and every non-empty element ends with a tag. Start tags are delimited by `<` and `>` and end tags are delimited with `</` and `>`.

The element name immediately follows the tag open delimiter. Names consist of a letter followed by up to 33 letters, digits, periods, or hyphens. Names are not case sensitive.

In a start tag, attributes and whitespace are allowed between the element name and the closing delimiter. An attribute consists of a name, and optionally an equal sign and a value. Whitespace is allowed around the equal sign. The attribute value is specified in a string surrounded by single or double quotes.

4.2 Entity References

The HTF DTD allows to represent 8-bit characters in the ISO Latin 1 character set using only 7-bit characters, and to represent characters that would otherwise be recognized ar markup (the characters

< and &), using SGML entity references. The character & signals the beginning of an entity reference when it is followed by a letter or a digit. The delimiter is followed by the entity name and a semicolon. For example:

```
"&Ouml;sterreich" is the German word for "Austria".
```

The character & itself is represented by the entity reference `&`, while `<` denotes the < character. These should always be represented in this way when they appear in data.

4.3 Comments

SGML foresees the possibility to include comments in a document that will be ignored by the parser. A comment is started with `<!--`, ended with `-->`, and may appear anywhere in the text. Some comments are understood and elimitated by Hyper-G on document insertion, and allow to specify the position of the document (parent collection, sort key), language, expiration time, etc. For example:

```
<TITLE>Staff Meeting Nov. 23, 4pm
<!--TimeExpire=93/11/23 16:00:00-->
This document will expire at the specified time.
```

4.4 Line breaks, spaces, tabs

A line break character is considered markup (and ignored) if it is the first or last piece of content of an element. This allows you to write either

```
<XMP>some example text</XMP>
```

or

```
<XMP>
some example text
</XMP>
```

and they will be processed identically. Also, a line that's not empty but contains no content will be ignored altogether. For example, the element

```
<XMP>
first line
<!-- comment -->
second line

fourth line
</XMP>
```

contains only the strings:

```
first line
second line

fourth line
\end{verbatim

As a special case, however, HTF treats an empty line (two line breaks following
each other) identically to a paragraph break, except within an \tag{XMP} or
\tag{PLAIN} element.

Space characters are rendered as horizontal white space. The rendering of a
horizontal tab (HT) character is not defined, and HT should therefore not be
used, except within an \tag{XMP} or \tag{PLAIN} element.

Neither spaces nor tabs should be used to make SGML source layout more
attractive or easier to read!

\newpage
\section{HTF Tags}

\subsection{Overview of Tags}

This table shows which of the tags are supported by the current
version (1.63) of the Hyper-G Terminal Viewer (HGTV) and the Hyper-G
X11-Viewer (Harmony) configured with the default style sheet (Appendix
\ref{Stylehgtv} and Appendix \ref{Styleharmony}) and which are
translated to what HTML tags in the WWW gateway configured with the
default style sheet (Appendix \ref{htf2html}) .

\vspace{5mm}
\begin{center}
\begin{tabular}{|l|c|l|}
\hline
Element & HGTV/Harmony & HTML\\
\hline
\tag{AS}      & yes & \tag{A}\\
\tag{AE}      & yes & \tag{/A}\\
\tag{COMPACT} & yes & auto. applied\\
\tag{CSPC}    & yes & tag ignored\\
\tag{DL}      & yes & \tag{DL}\\
\tag{DT}      & yes & \tag{DT}\\
\tag{DD}      & yes & \tag{DD}\\
\tag{H0}      & yes & \tag{H1}\\
\tag{H1}      & yes & \tag{H1}\\
\tag{H2}      & yes & \tag{H2}\\
\tag{H3}      & yes & \tag{H3}\\
\tag{H4}      & yes & \tag{H4}\\
\tag{H5}      & yes & \tag{H5}\\
```

```
\tag{H6}      & yes & \tag{H6}\\
\tag{HP0}     & yes & nothing\\
\tag{HP1}     & yes & \tag{STRONG}\\
\tag{HP2}     & yes & \tag{STRONG}\\
\tag{HP3}     & yes & \tag{EM}\\
\tag{HR}      & yes & \tag{HR}\\
\tag{INLINE}  & no/yes  & \tag{IMG}\\
\tag{IT}      & yes & \tag{LI}\\
\tag{KEY}     & yes & element ignored\\
\tag{L LEFT}  & yes & \tag{BR}\\
\tag{L RIGHT} & yes  & \tag{BR}\\
\tag{L CENTER}& yes & \tag{CENTER}\\
\tag{L INDENT}& yes  & \tag{BR}\\
\tag{LI ALPHA}& yes  & \tag{OL}\\
\tag{LI ARABIC}& yes & \tag{OL}\\
\tag{LI DEFAULT}&yes& \tag{UL}\\
\tag{LI NONE} & yes  & \tag{UL}\\
\tag{LQ}      & yes/no & \tag{BLOCKQUOTE}\\
\tag{P}       & yes & \tag{P}\\
\tag{PLAIN}   & yes & \tag{PRE}\\
\tag{SUB}     & no & \tag{SUB}\\
\tag{SUP}     & no & \tag{SUP}\\
\tag{TITLE}   & yes & \tag{TITLE}\\
\tag{VERBATIM}& yes & \tag{XMP}\\
\tag{XMP}     & yes & \tag{PRE}\\
\hline
\end{tabular}
\end{center}

\subsection{The HTF Document}
\label{HTFoverview}

The tags of a valid HTF document (within this context 'HTF document'
shall mean the SGML instance of the document, see section
\ref{SGMLinstance}) may not appear in any order but are restricted to
the formal grammar specified in detail in the HTF DTD (Appendix
\ref{DTD}).  As the DTD is not easy to understand, we now give an
informal and simplified description of what is a valid HTF document.
For example, here are two typical HTF documents:

\begin{verbatim}
<TITLE>A structured HTF document
<H1>Heading Level One<P>
This is a typical paragraph.

And this is another one.
<LI>
<IT>first list item
```

```
<IT>second list item
</LI>

<title>A preformatted HTF document<plain>
This is a preformatted text that appears exactly as typed in:

         A | B | C
        ---+---+---
         1 | 5 | 7

Be careful to replace all &lt; and &amp; symbols !!!
```

The first tag of an HTF document is always the <TITLE> tag. The end tag may be omitted, as the title is terminated on one of the <H0>, <H1>, <COMPACT>, <P> or <PLAIN> tags.

From above examples you see that there are basically two kinds of HTF documents: Structured HTF and plain ASCII. If the title is followed by the <PLAIN> tag, the rest of the text is treated as plain ASCII text, i.e., it is rendered in a fixed font as it is typed in, with respect to line feeds. From the <PLAIN> tag on no other HTF tags may appear, with the exception of <AS>, <AE>, <KEY>, <INLINE>, <UCS> and <FN>.

Formatted HTF, on the other hand, usually starts with a <H1> tag and continues with a number of paragraphs, lists, other headings, etc. The advantage of formatted text over plain ASCII is that its layout is defined by the renderer that can use different (proportional-spaced) fonts and take window size and user preferences into account to make the document appear more pleasing.

There are six tags that may appear anywhere in an HTF document: <AS>, <AE>, <KEY>, <INLINE>, <UCS> and <FN>. In a formatted HTF document, the tags <HP0>, <HP1>, <HP2>, <HP3>, <SUB>, <CSPC>, <Q> and <SUP> (and their corresponding end tags) may appear anywhere after the title, including headings and example texts. The other tags may appear in paragraphs, unless explicitly noted. The exact rules are defined in the DTD (Appendix B).

4.5 <AS>, <AE>: Anchors

Anchors in Hyper-G (and thus HTF) may appear anywhere (including plain ASCII text except enclosed in <VERBATIM> tags), and they may be arbitrarily nested and overlapping (in contrast to HTML). To achieve this, there are actually two tags: <AS> (Anchor Start) and <AE> (Anchor End), of which only the start tags are used. The required `ID` attribute specifies what <AS> and <AE> tags belong together and form one anchor. This mechanism is used for both source and destination anchors. Source anchors have an additional `DHINT` (destination hint) attribute, and destination anchors an additional `KEY` attribute in the <AS> tag.

Because Hyper-G separates links from documents and stores them in a link database, the <AS> and <AE> tags are handled specially by Hyper-G. When storing an HTF document, it is parsed and the anchors are stripped from the text and inserted into the link database. When retrieving the HTF text from the server, synthetic <AS> and <AE> tags are generated from the anchor information stored in the database. This allows to update (create, delete, modify) links without modification of the text and to use them independently of the text (e.g. for creating graphical overviews of the linking structure).

4.5.1 Definition of a source anchor

Although the Hyper-G clients allow interactive definition of links without having to worry about anchor tags, it is sometimes convenient (e.g. when inserting bulky data using some script language) to create links by directly specifying the anchor tags. A source anchor definition looks like this:

```
<AS ID="0x12345678" DHINT="collname lg:doctitle">anchor text<AE ID="0x12345678">
```

where *collname* is the name of the collection the destination document is located in, *lg* is the language it is written in (currently defined: 'en' (English), 'fr' (French), 'ge' (German), and 'it' (Italian)) and *doctitle* is its title. If the title within the destination collection is not unique, you may also specify a unique keyword that has been attached to the document (see the `<KEY>` tag). If the destination is not the whole document but a destination anchor, specify its `KEY` attribute (see below). Don't forget to surround the attributes by quotes. The values of the `ID` attributes are not relevant as long as they are unique within the document. When retrieving a text from the Hyper-G server, however, the 'real' anchor IDs within the Hyper-G database are inserted and should not be edited.

Should the target document not exist (yet), an 'open' link is inserted into the database. Open links are not shown when looking at the source document. Everytime a new document is inserted, it is matched against the open links for the corresponding collection. If a matching link is found, the link is closed (i.e. it is now visible and selectable from the source document). This feature allows easy inserting a number of documents with circular references in any order. Also, when a document is deleted the links pointing to it are marked as open, i.e. they cannot be selected any more from the source documents. However, when a document with the same name is inserted into the same collection, the links are restored again. This makes it easy to replace a document by a new version without destroying the links pointing to it.

4.5.2 Definition of a destination anchor

A destination anchor may specify a region or just a point within the text. The idea is that when jumping to a destination anchor the client will scroll to the start of the destination anchor, mark the region of the destination anchor and make sure that it is visible (if possible). The following is a valid definition of three nested, overlapping destination anchors:

```
<AS ID="0x00000001" KEY="en:TheWholeList">
<LI>
<IT>Item 1
<AS ID="0x00000002" KEY="en:Items2-4">
<IT>Item 2
<AS ID="0x00000003" KEY="en:Items3-5">
<IT>Item 3
<IT>Item 4
<AE ID="0x00000002">
<IT>Item 5
<AE ID="0x00000003">
</LI>
<AE ID="0x00000001">
```

The `KEY` values are exported as the titles of the destination anchors, so that you can refer to them, put them into collections or clusters and attach links to them. The values of the `ID` attributes are not relevant as long as they are unique within the document. When retrieving a text from the Hyper-G server, however, the 'real' anchor IDs within the Hyper-G database are inserted and should not be edited.

4.5.3 Rendering

In the Hyper-G Terminal Viewer configured with the default style sheet (Appendix C) source anchors are shown using inverse characters, with a small unique number attached in square brackets. Typing the number selects the corresponding link. Destination anchors are not shown. When jumping to a destination anchor, the text is scrolled so that the start point is visible.

The X11 client Harmony underlies the characters of the anchor with a special background color defined in the X11-Resources:

```
Harmony.Text.SGMLDocument.anchorColor:
Harmony.Text.SGMLDocument.additionalAnchorColors:
```

for Harmony.

In the WWW gateway source anchors are translated to HTML `<A>` tags, with the `HREF` attribute set to the global object ID of the destination. Typical WWW clients display this tag using a different color and allow the user to click on it using the mouse.

4.6 `<COMPACT>`: Whitespace compression

This tag is allowed only immediately after the `<TITLE>` tag and turns on the whitespace compression for the whole document. To compress whitespace multiple spaces are replaced by a single space, newline character are translated to spaces before the compression.

4.6.1 Example of use

```
<TITLE>This is the title
<COMPACT>
<H1>This is the heading
<P>
Some   text with whitespaces    compression.
```

4.6.2 Rendering

The renderer gathers multiple whitespaces and displays them as one single space only.

Both the Hyper-G Terminal Viewer and Harmony support this tag. The WWW gateway does no whitespace compresssion, because typically this is done by the WWW client automaticaly.

4.7 <CSPC>: Whitspace compression

The `<CSPC>` tag turns the whitspace compression on for the text between the start and the end tag.

4.7.1 Example of use

```
<TITLE>Title
<H1>Heading
<CSPC>
<P>
Some   text with whitespaces    compression.
</CSP>
Text without whitespace compression.
```

4.7.2 Rendering

The renderer gathers multiple whitespaces and displays them as one single space only.

Both the Hyper-G Terminal Viewer and Harmony support this tag. The WWW gateway does no whitespace compresssion, because typically this is done by the WWW client automaticaly.

4.8 <DL>, <DT>, <DD>: Glossary

A glossary (or definition list) is a list of paragraphs each of which has a short title alongside it. Apart from glossaries, this element is useful for presenting a set of named elements to the reader. The glossary is started by the `<DL>` tag (and ended by `</DL>`). In between (and only there) the following tags are allowed:

`<DT>`: The *term*, typically a single word.

`<DD>`: The *definition* of the term, which may wrap onto many lines.

These elements must appear in pairs, `<DT>` first. End tags can be omitted. It should be noted that glossaries and lists can be arbitrarily nested.

4.8.1 Example of use

```
<DL>
<DT>DT<DD>The <HP2>term</HP2>, typically a single word.
<DT>DD<DD>The <HP2>definition</HP2> of the term, which may
wrap onto many lines.
</DL>
```

4.8.2 Rendering

In the Hyper-G Terminal Viewer configured with the default style sheet (Appendix C) the *term* is rendered in boldface on a new line, with a trailing colon and a tab after the colon. The *definition* starts at a new line and is indented by 8 columns.

The Hyper-G X11 Textviewer renders the *term* with medium emphasis using the font and foreground color defined in the X11-Resources:

```
Harmony.Text.SGMLDocument.medium.foreground:
Harmony.Text.SGMLDocument.medium.font:
```

The *definition* is rendered in the default font and color.

In the WWW gateway configured with the default style sheet (Appendix F) the `<DL>`, `<DT>`, and `<DT>` tags are passed through unchanged, as they are defined similarly in HTML. However, the original HTML specification [BC93] does not foresee nesting of glossaries and lists, although better clients will be able to do so.

4.9 `<H0>`..`<H6>`: Headings

Seven levels of headings are supported. However, using all of them is discouraged, since a document that needs all these levels is probably too large and ugly to function well as hypertext. Rather, the document should be split into smaller pieces and the high-level structure be reflected in the collection hierarchy.

A heading element implies all the font changes, paragraph breaks before and after, and white space (for example) necessary to render the heading. Further emphasis (e.g., using `<HP0>`..`<HP3>` tags) is not necessary (although possible).

The highest level of heading is `<H0>` or `<H1>`. It is recommended to use `<H1>` as the first level (usually immediately after the title; although it is legal HTF to define a few paragraphs between the title and `<H1>`). If one later wants to combine a number of documents into a larger one (e.g., for printing), one can use the `<H0>` tag to define one additional level on top (corresponds to the `\chapter` directive of LaTeX).

The end tag is not used. The title itself is ended on a paragraph break, while the element continues until the next heading. It is illegal to jump from one heading to a level more than one below, e.g., `<H3>` after `<H1>`.

4.9.1 Example of use

```
<TITLE>Example Document
<H1>Section 1: How to do it<P>
This is some text.
<H2>Subsection 1.1: I dont know.<P>
More text.
<H2>Subsection 1.2: No idea.<P>
Some text.
<H1>Section 2: Now I know.<P>
Last text.
```

4.9.2 Rendering

The Hyper-G Terminal Viewer configured with the default style sheet (Appendix C) increments the paragraph indent by 2 columns on every heading. `<H0>` headings are displayed centered, bold face and underlined, `<H1>`, `<H3>`, `<H5>` left, boldface and underlined, `<H2>`, `<H4>`, `<H6>` left, boldface.

Harmony uses the fonts and foreground colors configured with the X11-Resources for the headings and increments the indentation level by 2 units on every heading. `<H0>` headings are displayed centered and heavyly emphasized, `<H1>`, `<H2>`, `<H3>` use heavy, medium and light emphasis and are indented 2 units less than the following text. `<H4>`, `<H5>` and `<H6>` text is rendered in the default text style.

The X11-Resorces used are:

```
Harmony.Text.SGMLDocument.Headings.H4.foreground:
Harmony.Text.SGMLDocument.Headings.H4.font:

Harmony.Text.SGMLDocument.Headings.H3.foreground:
Harmony.Text.SGMLDocument.Headings.H3.font:

Harmony.Text.SGMLDocument.Headings.H2.foreground:
Harmony.Text.SGMLDocument.Headings.H2.font:

Harmony.Text.SGMLDocument.Headings.H1.foreground:
Harmony.Text.SGMLDocument.Headings.H1.font:
```

In the WWW gateway configured with the default style sheet (Appendix F) the `<H0>` tag is translated to `<H1>`, all others pass unchanged. X Mosaic uses bold fonts with decreasing font size for the headings.

4.10 `<HP0>`..`<HP3>`: Highlighted Phrases

The tags `<HP0>`, `<HP1>`, `<HP2>`, and `<HP3>` (and their corresponding end tags) may appear anywhere in the text, and are used to provide emphasis or highlights on individual words or phrases. They do not cause a paragraph break.

As the availability of highlighting features varies from client to client (e.g., bold, italic, slanted, underline, color, font size), the HTF highlighted phrases are logical styles, rather than physical. The idea is that `<HP0>` denotes the default text style, `<HP1>` is a light emphasis, `<HP2>` a strong emphasis, and `<HP3>` a very strong emphasis. The author should not make assumptions on how the highlight will be realized by the browser (e.g., do not use "the italic parts are entered by the user"). A client may even ignore these tags (or some of them) altogether.

4.10.1 Example of use

These tags may be arbitrarily nested. For example, the `<HP0>` tag may be used to temporarily switch back to normal in the middle of an emphasized section:

```
The <HP2>'create document', 'delete document', <HP0>and</HP0>
'edit document'</HP2> commands may be used to manipulate documents.
```

End tags may be omitted if the situation is unambiguous:

```
standard<hp1>emphasis<hp2>strong emphasis<HP3>strongest emphasis</HP1>standard
```

4.10.2 Rendering

In the Hyper-G Terminal Viewer configured with the default style sheet (Appendix C) the highlight tags are interpreted like this:

`<HP0>`: standard

`<HP1>`: bold

`<HP2>`: underlined

`<HP3>`: bold and underlined

The strongest emphasis available, inverse video, is reserved to highlight anchors. Bold has been chosen as lightest emphasis because it is not available on all terminals. However, some terminals that do not support bold render bold as inverse video (like "cmdtool" on SUNs), or using a different color (shell terminal on SGIs). In such cases it may be necessary to modify the style sheet to achieve the desired effect.

The Hyper-G X11 client Harmony uses the fonts and foreground colors specified in the none, light, medium and heavy X11-Resource for default text:

```
Harmony.Text.SGMLDocument.none.foreground:
Harmony.Text.SGMLDocument.none.font:

Harmony.Text.SGMLDocument.light.foreground:
Harmony.Text.SGMLDocument.light.font:

Harmony.Text.SGMLDocument.medium.foreground:
Harmony.Text.SGMLDocument.medium.font:

Harmony.Text.SGMLDocument.heavy.foreground:
Harmony.Text.SGMLDocument.heavy.font:
```

In the WWW gateway configured with the default style sheet (Appendix F), the HTF highlight tags are translated to the following HTML tags:

`<HP0>`: nothing

`<HP1>`: `<STRONG>`

`<HP2>`: `<STRONG>`

`<HP3>`: `<EM>`

4.11 `<HR>`: Horizontal Rule

The `<HR>` tag has a similar function to the paragraph tag (see section 4.17) but also draws a horizontal line across the page. No end tag is necessary.

4.11.1 Example of use

```
This is a paragraph<HR>
A second paragraph

This is another paragraph
and this is not.
```

4.11.2 Rendering

The Hyper-G Terminal Viewer does not support this feature.

The Hyper-G X11 Text Viewer configured with the default style sheet (Appendix D) draws a shaded horizontal line across the page.

In the WWW gateway configured with the default style sheet (Appendix F) the `<HR>` tag is passed unchanged as it has the same effect in HTML.

4.12 <INLINE>: Inline images

The `<INLINE>` tag is used to define an inline image in the document. The image is inserted at the tag position and aligned with the base line of the current text line. The following graphic formats are supported:

- GIF with 1 to 24 bits depth
- TIFF with 1 to 24 bits depth
- JPEG with 24 bits depth
- XBM with 1 bit depth

4.12.1 Example of use

To display an image inside a text document, it has to reside somewhere in the Hyper-G database as a image document. The mechanism to reference the image is the same as used for anchor tags. The `<INLINE>` tag has two attributes, the ID and the DHINT attribute, which have the same meaning as in the anchor start tag `<AS>`. `<INLINE>` tags are allowed everywhere in the text; there are no restrictions.

```
This is a picture of the grand canyon:
<INLINE ID="0x0004711" DHINT="pictures en:grndcan">
```

4.12.2 Rendering

The Hyper-G Terminal Viewer does not support inline images, so they are not rendered at all.

Harmony displays the referenced image at the position of the `<INLINE>` tag.

In the WWW gateway configured with the default style sheet (Appendix F) the `<INLINE>` tag is translated to the `<IMG>` tag.

4.13 <KEY>: Keywords

The `<KEY>` tag allows to specify invisible keywords that may be used to search for the document. When inserting the HTF text into Hyper-G, the text is scanned, and the words contained within `<KEY>` and `</KEY>` are considered additional keywords that get stored with the Hyper-G object that describes the document, and are indexed so that they can be searched for. `<KEY>` may be specified anywhere in the text (even in plain ASCII text), any number of times.

You do not have to specify words that are contained in the title as keywords, as they are automatically extracted and indexed. Do not confuse this tag with the `KEY` attribute of the `<AS>` tag.

4.13.1 Example of use

There are two main uses of the <KEY> tag. Sometimes the title does not contain all the words you may find the document under. For example, this lets us find the description of the "sprintf(3)" subroutine in the manual page with title "printf(3)":

```
<TITLE>printf(3)
<KEY>fprintf,sprintf</KEY>
This manual page describes the subroutines <HP2>printf, fprintf, sprintf</HP2>.
```

The invisible keywords are also taken into account for the automatic link closing operation described in section 4.5.1. Sometimes when importing bulky data from some other database, it is convenient to attach a unique record identifier (or filename) with a document and to direct links to that identifier. In this case the additional keyword is not intended to be entered by humans, but rather to be used by the system to uniquely identify the document when the title is not sufficient or inconvenient:

```
<TITLE>Orbit 123, January 4th, 1992<PLAIN>
<KEY>ID00123DATE920104</KEY>
The big table starts here.

Somewhere else I might make a <AS ID="0x00000000"
DHINT="orbits en:ID00123DATE920104">link<AE ID="0x00000000"> to this document
```

4.13.2 Rendering

The text contained within <KEY> and </KEY> is not rendered at all. It is also not translated to HTML.

4.14 <L>: Line

The <L> tag allows to influence the rendering of the text. It forces a single line break, without the vertical white space that is usually generated by the <P> tag. In addition, rendering of the following line(s) can be influenced with a position attribute:

LEFT	left justified
RIGHT	right justified
CENTER	centered
INDENT	left justified with additional indentation according to the current level of heading (this is the default value for the POSITION attribute)

This setting is valid for the following paragraph only (more precisely, until the next tag is received, with the expection of <HP0>..<HP3>, <SUB>, <CSPC> and <SUB>. Remember that an empty line is equivalent to a <P> tag). If you want to continue in the same style for another paragraph, use the <L> tag without an attribute.

4.14.1 Example of use

A common use of the `<L LEFT>` tag is to force a line break without generating vertical space, when you don't want to use the `<XMP>` tag (e.g., because you don't want to switch to a fixed font), like in this poem:

```
<L LEFT>Dearest creature in creation
<L>Studying English pronunciation,
<L>I will teach you in my verse
<L>Sounds like corpse, corps, horse and worse<P>
```

4.14.2 Rendering

The text between the start tag and the terminating tag is rendered according to the POSITION attribute in the same way for both the Hyper-G Terminal Viewer and the X11 Viewer Harmony. For backward compatibility to older documents the default style sheet (Appendix C) defines the `<L LEFT>` tag in the same way as the `<L INDENT>` tag.

An equivalent tag does not exist in HTML. Therefore, in the WWW gateway configured with the default style sheet (Appendix F) all `<L>` tags are translated to `<P>`, except `<L CENTER>`, which generates a `<CENTER>` tag. This will start a new paragraph.

4.15 `<LI>`,`<IT>`: Lists

A list is a sequence of items (paragraphs, other lists, etc.), each of which may be preceded by a special mark or sequence number. An optional attribute specifies how they should be marked:

ALPHA:	a, b, c ...
ARABIC:	1, 2, 3 ...
DEFAULT:	Typically bullets, etc.
NONE:	nothing

If no attribute is specified, DEFAULT is used. It should be noted that lists may be arbitrarily nested.

4.15.1 Example of use

```
<LI arabic>
<IT>(Item 1)
<LI alpha>
<IT>(Item 1a)
<IT>(Item 1b)
</LI>
<IT>(Item 2)
</LI>
```

4.15.2 Rendering

In the Hyper-G Terminal Viewer configured with the default style sheet (Appendix C) a "o" is used in front of each item for the list with the DEFAULT attribute. The ALPHA and ARABIC list items are preceeded by a number or letter depending on the sequence number of the item in the list followed by a closing brace or a dot. The following text is indented by the width of the string used for numbering the item.

The X11 Client Harmony uses a similar rendering as the Terminal Viewer configured with the default style sheet (Appendix D).

In the WWW gateway configured with the default style sheet (Appendix F) ALPHA and ARABIC lists are translated to `<OL>` (ordered list), other `<LI>` tags are translated to the HTML tag `<UL>` (unordered list). The HTF tag `<IT>` is translated to the HTML tag `<LI>` (list item). Note that the original HTML specification [BC93] does not foresee nesting of glossaries and lists, although better clients will be able to do so.

4.16 `<LQ>`: Long Quote

The `<LQ>` tag allows text quoted from elsewhere to be rendered specifically. Typically, this means a light extra left and right indent, and/or italic font (if available). White space should be rendered before and after it.

4.16.1 Example of use

```
In his 1945 article ''As we may think'' Vannevar Bush writes:
<LQ>
Mendel's concept of the laws of genetics was lost to the world
for a generation because his publication did not reach the few who were
capable of grasping and extending it; and this sort of catastrophe is
undoubtedly being repeated all about us, as truly significant attainments
become lost in the mass of the inconsequential...
</LQ>
```

4.16.2 Rendering

In the Hyper-G Terminal Viewer (Appendix C) and the X11 Client (Appendix D) configured with the default style sheets, `<LQ>` means an additional left indent of 2 columns; in addition Harmony displays the text with light emphasis.

In the WWW gateway configured with the default style sheet (Appendix F), the HTF `<LQ>` tag is translated to the HTML `<BLOCKQUOTE>` tag which has the same meaning.

4.17 `<P>`: Paragraph

The `<P>` tag indicates a paragraph break. There is no end tag necessary. To ease importing of existing material, the LaTeXconvention that an empty line should also imply a paragraph break was adopted in HTF (except when in `<XMP>` context, see section 4.22).

You should not use `<P>` to generate extra white space around headings, lists, etc. It is the responsibility of the rendering software to generate that white space. If you want to specify a line break with no white space, you may use the `<L>` tag.

4.17.1 Example of use

```
This is a paragraph<P>
A second paragraph

This is another paragraph
and this is not.
```

4.17.2 Rendering

In the Hyper-G Terminal Viewer configured with the default style sheet (Appendix C) the `<P>` tag generates one empty line. The style sheet foresees the possibility to generate a small extra left indent on the first line of the paragraph.

The Hyper-G X11 Text Viewer configured with the default style sheet (Appendix D) generates some vertical space after the contents of the paragraph.

In the WWW gateway configured with the default style sheet (Appendix F) the `<P>` tag is passed unchanged as it has the same effect in HTML.

4.18 `<PLAIN>`: Plain Text

The `<PLAIN>` tag may appear **ONLY** immediately after the title (see section **??**) to indicate that the remaining text is to be treated as plain ASCII, i.e., it is rendered in a fixed font as it is typed in, with respect to line feeds. From the `<PLAIN>` tag on no other HTF tags may appear, with the exception of `<AS>`, `<AE>`, and `<KEY>`, i.e., there may be anchors and invisible keywords. It is important that all $<$ and & characters are escaped (i.e., by replacing $<$ by `<` and & by `&`).

4.18.1 Example of use

```
<TITLE>table 1<PLAIN>

+---+---+
! 1 ! 2 !
+---+---+
! 3 ! 4 !
+---+---+

This has been done using the &lt;PLAIN> tag.
```

4.18.2 Rendering

In the Hyper-G Terminal Viewer the text is laid out exactly as specified with respect to line breaks (the Terminal Viewer always uses a fixed font). If a line is too long, it is not truncated, but wraps around onto the next line.

The X11 Client Harmony uses in addition to the rendering mentioned above a fixed font specified in the X11-Resources:

```
Harmony.Text.SGMLDocument.Fixed.none.foreground:
Harmony.Text.SGMLDocument.Fixed.none.font:

Harmony.Text.SGMLDocument.Fixed.light.foreground:
Harmony.Text.SGMLDocument.Fixed.light.font:

Harmony.Text.SGMLDocument.Fixed.medium.foreground:
Harmony.Text.SGMLDocument.Fixed.medium.font:

Marmony.Text.SGMLDocument.Fixed.heavy.foreground:
Harmony.Text.SGMLDocument.Fixed.heavy.font:
```

In the WWW gateway configured with the default style sheet (Appendix F) the `<PLAIN>` tag is translated to the HTML `<PRE>` tag.

4.19 `<SUB>`,`<SUP>`: Subscript/Superscript

The `<SUB>` and `<SUP>` tags allow subscript and superscript, respectively (if possible). This may be useful for, e.g., chemical or mathematical formulas. However, the tag is not supported by either the Hyper-G Terminal Viewer or Harmony. Some WWW clients do the correct rendering, so the tags are sent to them unchanged.

4.19.1 Example of use

```
H<sub>2</sub>O
E = mc<sup>2</sup>
```

4.19.2 Rendering

In the Hyper-G Terminal Viewer configured with the default style sheet (Appendix C) the contents of a `<SUP>` block are placed between parentheses (for certain religious reasons), while the `<SUB>` tag is ignored altogether, i.e., its contents are rendered as usual.

Harmony, the X11 client for Hyper-G, does not support these tags yet.

In the WWW gateway configured with the default style sheet (Appendix F), the tags are leaved unchanged, i.e., the tags and their contents are passed to the WWW client as they are (HTML also has a `<SUB>` and `<SUP>` tags).

4.20 `<TITLE>`: Document Title

The title of a document is specified by the `<TITLE>` tag. It has to appear first in the document, and there may be only one `<TITLE>` tag. The end tag may be omitted, as the title is terminated on one of the `<H0>`, `<H1>`, or `<PLAIN>` tags. The title is not part of the text of the document (i.e., it is not rendered), but a property of the whole document.

When inserting an HTF text into Hyper-G, the document is scanned and the title stored in the Hyper-G database and indexed so that it can be searched for. You may later on modify the document title in the database as often as you wish without really editing the text. The `<TITLE>` tag is in fact only relevant when the document is inserted for the first time.

4.20.1 Example of use

Choose titles such that they are meaningful not only within context, e.g., choose:

```
<TITLE>Introduction to Hyper-G Text Format (HTF)
<H1>Introduction<P>
```

instead of

```
<TITLE>Introduction
<H1>Introduction<P>
```

In both cases, only "Introduction" will be rendered as heading, but in the first case the document will be found when titles are searched for "HTF", and an informative title can be displayed.