# Folio Flat File Reference

## Overview of Folio Flat File

Click here for an overview of Folio Flat File

## Accessing the Reference

The Flat File Reference is divided into two sections: a flat file index and an alphabetical listing of all flat file codes. The index lists flat file codes by type and provides links into the reference. The reference lists all primary flat file codes alphabetically.

Click a section to go directly to it.

Flat File Code Index

Alphabetical Flat File Listing

## Using the Flat File Query Template

The Search Flat File Codes query template helps you find flat file codes quickly. You may search for the code by name, by code, or by description.

Search by the name of the code if you know the name of the code and need to see how to implement it (such as Border or Paragraph Style).

Search by code if you know the two-letter code and need to see how it is used or what it is used for (for example, you might be examining a flat file and need to see what the TA code is used for).

Search by description if you are not sure what code you are looking for, but know generally what you want to do (for example, you want to create a frame around a paragraph).

Overview of Folio Flat File

## Introduction

Folio flat file is a text-based markup language which represents all of the text and features of an infobase, including styles, levels, links, and objects (graphics).

Use flat file codes to mark up existing text files you wish to create an infobase from, to add features to an existing infobase that has been saved as a flat file (such as a word processor document you have imported and wish to enhance), or to update an infobase. Exporting an infobase to flat file is also the most reliable method to back up an infobase. (Save the infobase to flat file and archive it; if something happens to the infobase, you can recreate it from the flat file).

In this section, you should learn how to:

• Identify the different components of a flat file code

• Distinguish between document codes and definition codes

• Distinguish between open, paired, and three-state codes

• Identify the differences between the 3.x and 4.x flat file formats.

In addition, this section should help you answer the following questions:

• What are the general rules you must follow when implementing flat file codes?

• Since flat file codes use open angle brackets to identify the code, how can you include an open angle bracket in the finished infobase?

• What is the standard unit of measurement for flat file codes? What are the exceptions?

• What is the difference between the flat file (\*.FFF) and the definition file (\*.DEF)? How are they related?

• What are the new codes used in the Folio Views 4.x flat file format? What other codes have changed from the 3.x version of flat file?

This section also includes a sample flat file and definition file which contain many of the standard codes. These are provided to show you how the codes can work together to create different effects.

A good way to learn the flat file codes is to create a small infobase. Use the features you want to see the codes for, and then export the infobase to flat file. You can then open the flat file in a text editor and see the codes. To get a good feel for how the codes correspond to infobase features, you could change the flat file and reimport it to see the corresponding changes to the infobase. Refer to the Alphabetical Flat File Listing for specifics on each of the codes.

The Folio Flat File documentation is divided into three parts:

• Overview of Folio Flat File (this section) provides a broad overview of the format as well as a sample flat flat file.

• Flat File Code Index is an index of all the Folio flat file codes, listing the code IDs, the code names, and links to where the codes are discussed in greater detail.

• Alphabetical Flat File Listing is an alphabetized list of all the codes, complete with a short definition of each code, the parameters for the code, and examples showing correct usage of the code. Where possible, cross references to related codes are provided.

Note: This section is not designed to instruct you on the components of an infobase. You should already have an understanding of what an infobase is and what its various components are before building a flat file. If you do not understand the components of an infobase, please refer to the Folio Views Reference in this infobase for more details.

# Key Definitions

Below are definitions of some basic terms you should know in order to understand the information in this section.

Flat File

An text-based markup language which represents a Folio infobase (similar, in some ways, to SGML and other markup languages). The use of the term flat file may refer to an entire flat file (a combination of the definition file and document file) or just to the document file (which uses the .FFF extension).

Definition File

The portion of a flat file which stores all style definitions, level definitions, field definitions, object definitions, and other information which affects the entire flat file.

Definition Code

A flat file code which may be used in the definition file.

Document File

The portion of the flat file which stores the text of the infobase as well as formatting codes, level codes, links, object references, and other items. The document file is often referred to as the flat file since it uses the .FFF file extension.

Document Code

A flat file code which may be used in the document file.

Paired Code

One of the three types of document codes. A paired code has a begin code and an end code. An example of a paired code is the Character Style code. You must specify where the character style application begins and where it ends. Only the text between the begin and end codes is affected by the character style. For example:

<CS:Emphasis>Warning!</CS>

Note: Paired codes begin and end in a single record. If a record break <RD> is found before an end code, the formatting is ended. If you want paired code formatting to extend across record boundaries, you need to begin the formatting at the beginning of each record. For example:  
  
 <RD><CS:Important>Catch this!  
 <RD><CS:Important>And this!

Open Code

One of the three types of document codes. An open code has a beginning code and affects all text following the code until the end of the record or paragraph in which the code is placed. An example of an open code is a paragraph style. A paragraph style affects the entire paragraph it is applied to.

Three-State Code

One of the three types of document codes. A three-state code has three possible settings: ON, OFF, or DEFAULT. When a code is on, the attribute is applied. When the code is off, the attribute is cleared. When the code is default, the text returns to whatever state it was in before the attribute was turned on or off. An example of a three-state code is the bold code. It may be on <BD+> (text following the code is bold), it may be off <BD-> (text following the code is not bold), or it may be default <BD> (text following the code is whatever it was before bold was turned on or off). Generally, the three-state codes appear in pairs — a starting code to turn an attribute on or off, and an ending default code to restore the previous attribute.

# Understanding Folio Flat File Codes

Following are some key ideas that you need to be familiar with. The concepts discussed in this chapter include:

• Definition (.DEF) & Document (.FFF) Files

• General Format of Flat File Codes

• General Rules for Flat File Codes

• Using Angle Brackets < and > in Your Documents

# Definition (.DEF) & Document (.FFF) Files

A Folio flat file is divided into two parts: a Definition file and a Document file. The definition file contains the style, object, and field definitions for an infobase; the document file contains the text of the infobase, style applications, object references, and other infobase features. Both files are required to build an infobase.

The document file (also called the flat file because it has a .FFF extension) stores a reference to one or more definition files. This reference is required to ensure that the proper styles and objects are applied when the infobase is created.

The Definition Include <DI> code is used to store the reference to the definition file. This code must be the first code in the document file (.FFF file). To use two or more definition files for the same flat file, insert them one after another at the top of the document file. If the document file does not reference a definition file, both Folio Views and Create do not interpret the styles and objects you have placed in the flat file (unless you are merging the flat file with an infobase or another flat file which defines the styles).

Directory Structures

When infobases are exported to the flat file format, the .DEF file and .FFF file are exported to the specified directory. Object files and query templates are exported to a subdirectory of the specified directory. A separate sub-directory for the objects and query templates is created. The name of the directory is the name of the flat file with a .OB extension.

For example, if you export an infobase to a c:\flatfile directory and the name of the flat file is MYFILE, then the directory structure created is:

c:\flatfile  
c:\flatfile\myfile.ob

# General Format of Flat File Codes

All Folio flat file codes follow the same general format. The format is:

<ID[:option][,option][,...]>

The tab set code below is provided as an example:

<TS:1.5,CN,DS>

The components of this code are:

< Open angle bracket — indicates the start of a code.

TS Identification tag (ID) — identifies the code. In this case, the code is a Tab Set code.

: Colon — separates the ID from any parameters required by the code. If the code has no parameters, the colon is not used.

1.5 Parameter (option) — specifies that the tab should be set 1.5 inches from the left margin.

, Comma — separates multiple parameters from each other.

CN Parameter (option) — specifies that the tab set should be center justified. This parameter uses a two-character ID tag.

DS Parameter (option) — specifies that the tab set should have a dots-and-spaces dot leader ( . . . . . ). This parameter uses a two-character ID tag.

> Closed angle bracket — indicates the end of a code.

General Code Rules

• All flat file codes are delimited by angle brackets < and >.

• All flat file codes have a two-character identification tag (ID). The ID is generally mnemonic (where possible) for easy identification. In the preceding example, the ID is TS (which stands for Tab Set).

• If the code has additional parameters (or options), the options are separated from the code by a colon :.

• Many of the parameters for a code are represented by a two-character identification tag. In the preceding example, CN and DS are parameters for the tab set code which use a two-character ID.

• Multiple parameters for a code are separated by commas.

• Numeric measurements for most codes are given in inches (our apologies to the metric world) or twips. In the Tab Set code, above, the 1.5 specifies the location of the tab set in inches from the left margin.

• Groups of sub-codes within a flat file code may be delimited by semi-colons to improve readability. See the examples in the Paragraph Style Definition code for actual usage and Using Semi-colons & Commas for more information.

# General Rules for Flat File Codes

Style Names (and other Names)

Many codes require that a name be included as one of the parameters. Names may be from 1 to 127 characters long (including spaces). Names are defined in the Definition File and are used in codes in the Document File. Names in the Document File must match names in the Definition File for the code to function correctly.

If the name includes spaces, periods, commas, apostrophes, or other non-alphanumeric characters, the name MUST be enclosed in full quotes. For example:

<CS:"Style of the Banana Republic">

If you want to include part of the name in full quotes, you must use double quotes around it in the code. For example:

<CS:"Style of the ""Banana"" Republic">

Numbers

All numbers must be entered in decimal form (1/2 and 1/4 are invalid). All numbers representing inch or point size values may be specified out to the thousandths position (three decimal places). Twip values must be integers.

By default, numbers represent distances measured in inches. There are, however, three exceptions: font point size <PT>, which is measured in points; superscript/subscript, which are measured in points; and line spacing <LS>, which is a multiple of the line height.

Numbers may also represent distances measured in twips, centimeters, or points. You may set a flat file export option to export all values in one of these units; to specify a unit other than inches in a flat file that you are creating, append a t (twips), c (centimeters) or p (points) to the number. For example: 1440t or 2.54c or 72p.

Path names, File Names, & Command Lines

Path names, filenames, and command lines are used in various codes (such as object definitions and query link codes). These refer to the actual pathnames, filenames, or command lines used to access various files, infobases, and external applications.

If a pathname, filename, or command line contains spaces, colons, periods, commas, back-slashes, or other non-alphanumeric characters, it must be enclosed in full quotes.

Relative paths are also supported. Relative paths follow the MS DOS standard:

\directory\filename — go back to the root drive to find the directory

directory\filename — the directory is a sub-directory of the current directory.

..\directory\filename — the directory is one level up from the current directory.

Two sample paths are shown below:

"..\Builder\DEF\STYLES.DEF"

"C:\Builder\FFF\CORPORAT.FFF"

If quotation marks are part of the pathname, filename, or command line, then double quotes must be used. For example:

"Homer:Word:""JAMMIN"""

Order of Codes

Folio Flat File codes are extremely flexible. However, there are some constraints on where codes may be placed and the order in which codes are used. See the Sample Flat File for an example of proper use of many of the standard codes.

Definition File

Only definition codes and comment codes may be placed in the definition file. Generally, these codes may appear in any order in the definition file. However, the following constraints apply to certain codes:

Levels — If you use the level definition code <LN>, it must be listed before any level style codes <LE>. (If desired, you may ignore the <LN> code and simply list the <LE> codes in hierarchical order.)

For a complete list of all definition codes, see the Flat File Code Index.

Document File

The document file contains the text of the infobase and the majority of the flat file codes used. Any non-definition file codes and comment codes may be used in the document file.

While there is a great deal of flexibility in how codes may be ordered within a category (such as character-based codes), certain codes, if used, MUST precede other codes.

The general order of codes for the document file are:

Comment Code <CM> — Comments can be placed anywhere in a definition or document file using the <CM> code. When you export an infobase with the flat file filter, the different sections of the flat file codes are documented with the comment code.

Definition Include Code <DI> — If the <DI> code is included in a document file, it must be placed before header or footer codes and the first record. One or more definition files may be specified by specifying each with a separate <DI> code. This code specifies the name and location of a definition file to include with the infobase.

Header <HE> and Footer <FO> Codes — If these codes are used, they must immediately follow the <DI> codes and must come before any record codes. Only one header and one footer is allowed in the document file (all others are ignored).

Named Popup Definition Codes <DP> — All named popup definition codes must appear in the document file before any record codes.

Record Code <RD> — Indicates the start of a new record. A record at the Normal Level is begun with the <RD> code. A record at any other level (a heading record) is specified by including an optional parameter with the RD code (<RD:"Heading 1">).

Paragraph Code <HR> — Indicates the start of a new paragraph within a record. This code is needed only when you want two or more paragraphs in the same record (the <RD> starts a new paragraph when it starts a new record).

Paragraph Style Code <PS> — Applies a paragraph style to a paragraph. To apply a paragraph style it must be defined in the definition file.

Paragraph-Based Formatting codes — Any paragraph-based codes, in any order (includes codes such as indents, tabs, and line spacing).

Text and All Other Codes — All other codes (all codes other than those listed above) may be placed at any position following the above listed codes in or around text or objects. Comment codes may also be included.

Note: Paragraph style codes and paragraph formatting codes may be placed anywhere in the text of the paragraph to which they apply. This allows some conversion processes more flexibility when converting information to flat file. However, these codes should generally be placed at the beginning of the record or paragraph to ease readability.

Exceptions for some codes are included in the Alphabetical Flat File Listing.

Using Semi-Colons & Commas

You may use either semi-colons or commas to delimit sub-codes within a flat file code. For example, the Paragraph Style Definition code allows you to include any of the paragraph or character formatting options within the code. Since some of these formatting options have several parameters of their own, using semi-colons between these options may improve readability. Note that semi-colons can only be used between codes; they may not be used between standard options in a code (for example, the Foreground Color code requires that you specify the Red, Green, and Blue color values for the color; these parameters must be separated by commas).

Below is a complete Paragraph Style Definition, showing all locations where a semi-colon may be used:

<PA:"Sample Style"**;** IN:0.5,0,-0.5**;** JU:CN; BP:0.699; AP:0.2; BR:BT:0.029,0.029, FC:0,0,255; FC:121,121,121>

Semi-colons are used between sub-codes in the Paragraph Style Definition code. Required parameters for the sub-codes are separated by commas.

Also note that there are two FC (foreground color) codes in this example. The first specifies the foreground color for the border (there is a comma preceding it, so it is assumed to be part of the BR (border) sub-code). The second specifies the foreground color for the text in the paragraph (it is offset with a semi-colon). For clarity, you would probably not put these two codes next to each other. This example does so to make a point.

Note: Semi-colons are optional on import (to allow for backwards compatibility with the 3.x versions of flat file). They are used by default on export. If you are unsure of what to use, use a comma.

Using Whitespace

Flat file codes may contain whitespace to improve readability. Whitespace includes spaces, tabs, and carriage returns.

You may use whitespace between codes and between sub-codes within a code. You may not use whitespace between required parameters for a code. For example, you may use whitespace between an IN code and a BR code in a Paragraph Style Definition code, but you may not use whitespace between the parameters of the IN code.

**Valid Use of Whitespace:**

<PA:List, IN:0.5,0,-0.5; BR:AL:0.1,0.1,FC:0,0,255>

**Invalid Use of Whitespace:**

<IN: 0.5, 0, -0.5>

Null Characters

Some text files (such as database downloads) may contain null characters (ASCII 0). The filter assumes that files with null characters are either UNICODE files or a binary file and rejects the file. Make sure that your file is free of null characters before importing or creating into an infobase.

If you are processing your files on a DOS\Windows computer, you may use the following FSR argument to remove all null characters from your files:

"[^\x01-\xFF]" "" -c

FSR is distributed as shareware with Folio. For more information on FSR, see the Folio Search & Replace help file in the \ToolDemo directory on the Folio Builder CD-ROM.

Control Characters

Many ASCII files contain control codes such as hard returns, tabs, and form feeds. The flat file import filter does not translate these codes to anything; it is as if they were not there. If you wish to retain these control codes, you must replace them with flat file codes before you import the flat file. Replace hard returns with either record codes <RD>, paragraph breaks <HR>, or carriage returns <CR>. Replace tabs with the tab code <TB>. Replace form feeds with the page break code <PB>.

If you are processing your files on a DOS\Windows computer, you may use the following FSR arguments to perform these transformations:

Hard Returns (Carriage Return Line Feed)

"\r\n" "<RD>" -c

Tabs

"\t" "<TB>" -c

Page Breaks

"\f" "<PB>" -c

# Using Angle Brackets in Your Documents

Since all flat file codes use angle brackets as delimiters, any text following an angle bracket in your flat file is treated as a flat file code. This can cause errors when you convert a flat file to infobase with the flat file import filter.

If you want an open angle bracket to appear in the text of your finished infobase, you must add an extra open angle bracket to the text. For example:

<<Sample Code

If you are working with a text file, add the extra open angle brackets before adding other flat file markup. (If you have exported a flat file from Folio Views, the export filter should have added the extra angle brackets for you.)

# Sample Flat File

Following is a sample flat file that uses many of the more common flat file codes. Both the definition file and the document file are included.

# Definition File

<CM> \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* Level Definitions \*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* </CM>

<LE:Book; PT:24; FC:0,0,255; JU:CN; BP:0.699306; AP:0.2; BR:BT:0.0291667,0.0291667,FC:0,0,255>

<LE:Section; BD+; PT:18; FC:0,0,160; BP:0.5; AP:0.25; BR:LF:0.05,0.0291667,FC:0,0,160,TP:0.1,0.05,FC:0,0,160>

<LE:Chapter; BD+; PT:14; FC:255,255,255; BP:0.25; IN:LF:0.25,FI:-0.25; TS:Right,RT,NO; BR:LF:0.0125,0,FC:0,0,128,TP:0.0125,0.00625,FC:0,0,128,RT:0.0125,0,FC:0,0,128,BT:0.0125,0.00625,FC:0,0,128; SD:0,0,128>

<LE:Heading; BD+; FT:Arial,SN; FC:0,0,160; BP:0.125>

<LE:"Sub-Heading 1"; BD+; UN+; FT:Arial,SN; PT:10; FC:0,0,160; BP:0.125; IN:LF:0.25>

<LE:"Sub-Heading 2"; FT:Arial,SN; PT:10; FC:0,0,160; BP:0.125; IN:LF:0.25; TS:0.25,NM,NO,0.5,NM,NO,0.75,NM,NO,1,NM,NO,1.25,NM,NO,1.5,NM,NO,1.75,NM,NO,2,NM,NO>

<LE:"Sub-Heading 3"; UN+; FT:Arial,SN; PT:10; FC:0,0,160; BP:0.125; IN:LF:0.5; TS:0.25,NM,NO,0.5,NM,NO,0.75,NM,NO,1,NM,NO,1.25,NM,NO,1.5,NM,NO,16.4181,NM,NO,16.4181,NM,NO>

<LE:"Sub-Heading 4"; FT:Arial,SN; PT:10; FC:0,0,160; BP:0.125; IN:LF:0.5; TS:0.25,NM,NO,0.5,NM,NO,0.75,NM,NO,1,NM,NO,1.25,NM,NO,1.5,NM,NO,16.4181,NM,NO,16.4181,NM,NO>

<LE:"Normal Level"; PT:10; FC:0,0,0; BP:0.1; IN:LF:0.5>

<CM> \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* Paragraph Style Definitions \*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* </CM>

<PA:"List 2"; PT:10; FC:0,0,0; BP:0.0701389; IN:LF:1,FI:-0.247917; TS:0.75,NM,NO,1,NM,NO,1.25,NM,NO,1.5,NM,NO,1.75,NM,NO,2,NM,NO>

<PA:Note; PT:10; FC:0,0,0; BP:0.1; IN:LF:0.5; BR:AL:0.0152778,0.0291667,FC:255,0,0>

<CM> \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* Character Style Definitions \*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* </CM>

<ST:Emphasis,CS; BD+; IT+>

<ST:"Example Text",CS; BD+; FT:Arial,SN; PT:9>

<ST:Keystrokes,CS; FT:Arial,SN; PT:8>

<ST:"Screen Items",CS; BD+>

<CM> \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* Link Style Definitions \*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* </CM>

<ST:Default,LK>

<ST:Jump,LK; UN+; FC:0,128,0>

<ST:Object,LK; UN+; FC:0,0,128>

<ST:Popup,LK; UN+; FC:128,128,0>

<ST:Program,LK; UN+; FC:128,0,0>

<ST:Query,LK; UN+; FC:128,0,128>

<CM> \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* Highlighter Definitions \*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* </CM>

<PD:"Fix This"; FC:255,0,0>

<PD:"Future Link"; BC:0,255,64>

<PD:Warning; FC:128,0,0>

<PD:"Cool Stuff", BC:0,255,255>

<CM> \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* Field Definitions \*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* </CM>

<FE:"3.1 - 4.0 Flat File",TX; BC:255,255,0>

<FE:Note,TX>

<FE:Popup,TX,IX:TF>

<FE:Publisher,TX>

<CM> \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* Query Templates \*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* </CM>

<QT:"&1 Search Folio Views Reference","whatsnew\FFF1.OB">

<QT:"&2 Search Flat File Codes","whatsnew\FFF2.OB">

<CM> \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* Folio Object Definitions \*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* </CM>

<OD:FO:"Folio Views 4.x Title Page",Bitmap,"whatsnew\FFF0.OB">

<OD:DL:"Samples","ExcelWorksheet",samples.xls">

<CM> Note that these objects are not included with this def file, so they will not appear when you import or create the file. </CM>

# Document File

<CM> \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* Definition File Include \*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* </CM>

<DI:"whatsnew.DEF">

<CM> \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* Infobase Information \*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* </CM>

<TT:"What's New to Folio 4 (Partial List)">

<RE:"9/16/96 4:20:40 PM">

<TP:"Folio Views 4.x Title Page">

<AU>"NextPage, Inc.</AU>

<RM>Copyright 2003, NextPage, Inc.. All rights reserved.</RM>

<HE><TS:Right,RT,NO><BR:LF:0,0,TP:0,0,RT:0,0,BT:0.0152778,0.0291667><SD:NO><IT+><FT:Arial,SN><PT:8>Folio Views Help<TB><GP><IT><FT><PT></HE>

<FO><LW:10><BP:0.1><AP:0.125><IN:LF:0.75,RT:0,FI:0><TS:Right,RT,NO><BR:LF:0,0,FC:128,128,128,TP:0.0152778,0.0291667,FC:128,128,128,RT:0,0,FC:128,128,128,BT:0,0,FC:128,128,128><SD:NO><IT+><FT:Arial,SN><PT:8>© Copyright 2002, NextPage, Inc..<TB><GP><IT><FT><PT></FO>

<CM> \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* Named Popup Definitions \*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* </CM>

<DP:"Hamlet">1. A small village. 2. A play by Shakespeare. 3. Contraction for 'ham omelet.'</DP>

<CM> \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* Record Text \*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* </CM>

<RD:Heading,CH><BH><JD:"Intro-What's New with Folio 4">What's New with Folio 4<EH>

<RD>There are several new and enhanced features in the Folio 4 product line. Some of the key features to note are listed below, divided by product.<HR><PS:"List 2">ï<TB><JL:Jump,"Intro-New to Folio Views">New to Folio Views<EL><HR><PS:"List 2">ï<TB><JL:Jump,"Intro-New to Folio Builder">New to Folio Builder<EL> (formerly the Infobase Production Kit)<HR><PS:"List 2">ï<TB><JL:Jump,"Intro-New to Folio Publisher">New to Folio Publisher<EL> (formerly the Professional Infobase Developer's Kit)

<RD>Refer to <QL:Query,"[Contents 'Folio 4.x Help Infobase','Welcome to Folio 4','The Benefits of Folio 4']",RH>The Benefits of Folio 4<EL> for a brief overview of each product.

<RD:"Sub-Heading 1"><JD:"Intro-New to Folio Views">New to Folio Views

<RD:"Sub-Heading 2">The Interface

<RD>The interface for Folio Views has changed considerably from previous versions. Rather than using multiple windows for displaying information in a single infobase, Folio Views now uses a single window composed of one or more panes arranged in a set of predefined views.

<RD>A view may contain a Document pane, Contents pane, Hit List pane, Reference pane, Object pane, or a combination of these panes. Changing the view of an infobase is as easy as clicking on a tab at the bottom of Folio Views. Within each view, you may size each pane to meet your needs.

<RD>This integration of information simplifies working with several infobases and their tables of contents at the same time by reducing the number of windows you have open.

<RD>See <QL:Query,"[Contents folio views reference,view,'views & panes']",RH>Views><EL> for more information on each of the default views and what each is designed for.

<RD:"Sub-Heading 2">Searching Enhancements

<RD>Finding information fast is what Folio Views does best, and this version does it better than ever. In addition to significantly speeding up phrase and proximity searches, Folio Views also supports true relevancy ranking in the Hit List and natural language queries.

<RD>Simple ranked queries allow you to use a natural language to search infobases (rather than the query syntax) and rank the hits in the Hit List pane.

<RD>A new Query toolbar allows you to quickly enter a simple ranked search at any time.

<RD>Of course, Boolean searches, wildcards, phrase, proximity, word stem, and synonym searches are still supported, as are searches in infobase structures (levels, fields, notes, popups, highlighters, and groups). A new Contents operator replaces the older level heading search. Phrase and proximity searches are enhanced not only to be faster, but also to support wildcards within phrase and proximity searches (<CS:"Example Text">"work\* not play$"</CS> is now a valid search).

<RD>And for those people who create long and complex queries and who never thought the old Query dialog was big enough, you can now make the Advanced Query dialog bigger: just drag a corner of the dialog to make it as large as you need (limited only by the size of your screen).

<RD>Another enhancement is the Query Selection tool. Select text in the infobase that contains the keywords or ideas that you want to search for, choose the Query Selection tool, and the selected text is used to perform a ranked query on the infobase.

<RD>See <JL:Jump,"FV-Advanced Query">Advanced Query<EL> and <JL:Jump,"FV-Query">Query<EL> for more information.

<RD:"Sub-Heading 2">Navigation Enhancements

<RD>Getting around in an infobase keeps getting easier. Single-click links move you from point to point faster than ever. Backtrack is replaced with Go Back and Go Forward ó after you backtrack from a link or a query, you can go forward again. Next Partition Hit and Previous Partition Hit moves you back and forth between partitions that have hits. This usually moves you from record to record, but can also be set up to move you from chapter to chapter or case to case (depending on the default partition that you set for the infobase or specify when performing the search).

<RD:"Sub-Heading 2">Personalization Enhancements

<RD><PN:"Cool Stuff">Folio Views provides tools to help you personalize and annotate your infobases.</PN> These tools not only help you mark information important to you, but also help reinforce the learning of that information (when you annotate something, you are more likely to remember it later). All of the standard personalization features available in earlier versions of Folio Views are still available in this version, and most of them have been enhanced. In addition, a few new ones have been added.

<RD>Some of the enhancements include: removing the limit on the amount of data stored in notes and popups, allowing you to change the caption for notes and popups, adding an option to move a bookmark, and providing a simpler method for adding and removing groups from records.

<RD>Also added is a simplified method for applying highlighters to an infobase. A new tool on the toolbar allows you to select a highlighter and then drag a highlighter pen across the text.

<RD>Three new link types are also included. <DL:Data,Samples>Data links<EL> allow you to specify a document to link to (such as a word processor document or a spreadsheet file) without specifying an application to open the document. The data link opens the appropriate application based on the file associations stored in the operating system's registry. <PX:"Named Popup",Hamlet>Named popup links<EL> allow you to store the contents of the popup as an object resource in the infobase: you can name the contents of the popup and use it multiple times without retyping or pasting any text. <EN:"End Note","link or highlighter",2.25,1.5,"Custom End Note Caption">End note links<EL> allow you to query the infobase and display the results of the query (all records with hits) in a popup window.

<RD:"Sub-Heading 2">Customizable Interface

<RD>The Folio Views toolbars, menus, and quick keys may be changed at any time. Through an intuitive dialog, you may rename, reorganize, add, or remove menu items. If you prefer to call the File menu the Access menu, you may do so. If you would rather have the Options menu item listed under Edit rather than Tools, you may move it. You may also change quick keys for common actions, or you may assign new quick keys for other actions that you perform regularly.

<RD>See <QL:Query,"[Contents 'Folio Views Reference,Tools,Customize ]",RH>Customize<EL> for more information.

<RD:"Sub-Heading 2">Tables

<RD>Tables are now fully supported in a Folio infobase. You can create your own tables directly in an infobase, or you may import tables from other formats. Standard table functionality is provided, including defining header columns and header rows, joining cells, alignment within cells, shading within cells, and border options for each cell.

<RD>

<TA:3,1,1,1; HG:0.138194; IN:LF:0.25; BR:AL:0.0208333,0,FC:0,0,128,HZ:0.0104167,0,FC:0,0,128,VT:0.0104167,0,FC:0,0,128; SD:192,192,192>

<RO,HE><CE: BR:AL:0.0416667,0.0298611,FC:0,0,128><JU:RT><BP:0><IN:LF:0><BD+><FT:"Arial Narrow",SR><PT:11><BC:DC>Element</CE><CE: BR:AL:0.0416667,0.0298611,FC:0,0,128><JU:CN><BP:0><IN:LF:0>Symbol</CE><CE: BR:AL:0.0416667,0.0298611,FC:0,0,128><BP:0><IN:LF:0>Number<BD><FT><PT><BC></CE>

<RO><CE><JU:RT><BP:0><IN:LF:0>Boron</CE><CE: BR:AL:0.0104167,0,FC:128,0,128><JU:CN><BP:0><IN:LF:0>B</CE><CE><BP:0><IN:LF:0>5</CE>

<RO><CE><JU:RT><BP:0><IN:LF:0>Lead</CE><CE: BR:AL:0.0104167,0,FC:128,0,128><JU:CN><BP:0><IN:LF:0>Pd</CE><CE><BP:0><IN:LF:0>46</CE>

<RO><CE><JU:RT><BP:0><IN:LF:0>Gold</CE><CE: BR:AL:0.0104167,0,FC:128,0,128><JU:CN><BP:0><IN:LF:0>Au</CE><CE><BP:0><IN:LF:0>79</TA>

See <JL:Jump,Table>Table<EL> for more information.

Flat File Code Index

The Flat File Index lists the 2-letter Folio flat file codes and their descriptive names. Each code is linked to the section in the reference containing more information about the code.

The codes are listed in two categories: Definition File codes and Document File codes.

If you are unsure where the code is listed, use the Search Flat File query template to look for the code based on the 2-letter code or descriptive name for the code.

## Definition File Codes

AP Append Merge

AS Abstract

AU Author

CL Class Object

CM Comment

CS Character Style

DF Default Font

DL Data Link Object

DQ Default Query

DT Date (field)

DT Date (Field indexing option)

FE Field Definition

FO Folio Object

FP Fast Phrase (Field indexing option)

FP Floating Point (field)

HL Hit List Properties

IR Integer (field)

IX Index Options (field)

LE Level Style Definition

LI Link to File option

LK Link Style

LN Level Definition

NO Not Indexed (Field option)

OD Object Definition

PA Paragraph Style Definition

PD Highlighter Pen Definition

PF Phrase Field (Field indexing option)

PR Default Partition

QT Query Template

RC Reconcile Shadow File

RE Revision Date

RM Remark

RP Replace Definition

ST Text Style Definition

SU Subject

SW Stop Words (Field indexing option)

TE Terms Enclosing (Field indexing option)

TF Terms Field (Field indexing option)

TM Time (field)

TP Title Page

TT Title

TX Text (field)

VI Version Information

## Document File Codes

AL All (Border option)

AP After Paragraph Spacing

BC Background Color

BD Bold

BH Begin Heading

BK Bookmark

BO Bottom (Table cell alignment option)

BP Before Paragraph Spacing

BR Border

BT Bottom (Border option)

CA Character Aligned Tab Set (Decimal Tab)

CD Condensed

CE Cell

CH Character

CH Custom Heading

CM Comment

CN Center Justified Tab Set

CN Center Justification

CR Carriage Return

CS Character Style

DA Dashes & Spaces Leader

DC Default Color Option

DP Named Popup Link Definition

DI Definition Include

DL Data Link

DO Dot Leader

DS Dots & Spaces Leader

DV Decorative

EH End Heading

EL End Link

EN End Note Link

FC Foreground Color

FD Field Application

FI Flat File Include

FL Full Justification

FO Footer

FT Font

FX Fixed Pitch Font

GA Generate Infobase Author

GD Generate Current Date

GF Generate Infobase File Name

GI Generate Infobase Title

GM Generate Last Modification Date

GP Generate Page Number

GQ Generate Current Query

GR Group

GT Generate Current Time

HC Header Cell (Table option)

HD Hidden

HE Header

HG Horizontal Gap (Table option)

HI Height (Table option)

HR Hard Return

HS Hard Space

HZ Horizontal Border (Table code)

ID Record ID

IN Indent

IT Italic

JD Jump Destination

JL Jump Link

JU Justification

JU Justification (Table option)

KN Keep Next (Paragraph formatting option)

KN Keep Next (Table option)

KT Keep Together (Paragraph formatting option)

KT Keep Together (Table option)

LF Left Justification (Paragraph formatting option)

LF Left Justification (Table option)

LH Line Height

LS Line Spacing

LT Popup Link Text

LV Level Application

LW Line Width

MD Merge Cells Down

ML Command (Menu) Link

MR Merge Cells Right

NM Normal Tab Set

NT Note

NO No Dot Leader

OB Object Application

OL Object Link

OU Outline

PB Page Break

PC Use the IBM PC character set (Font option)

PL Program Link

PN Highlighter Pen Application

PP Paragraph Proximity

PS Paragraph Style Application

PT Point Size

PW Popup Link

PX Named Popup Link

QL Query Link

RD Record

RH Records with Hits (Query link option)

RO Row (Table option)

RT Right Justification (Paragraph formatting option)

RT Right Justification (Table option)

RT Right Justified Tab Set

SB Subscript

SC Script

SD Shade

SE Sentence Proximity

SH Shadow

SN Sans Serif Font

SO Strikeout

SP Superscript

SR Serif Font

SS End superscript or end subscript

SY Use the Symbol character set (Font option)

TA Table Codes

TB Tab

TO Top cell alignment (Table option)

TS Tab Set

UL User Link

UN Underline Dot Leader

UN Underline

UX User Extension

VA Vertical Alignment (Table option)

VG Vertical Gap (Table option)

VI Version Information

VT Vertical Border (Table option

WP Wallpaper

WW Web Link

ZM Zoom

Alphabetical Flat File Listing

This section lists all of the Folio flat file codes in alphabetical order. The generic format for the codes follows:

# Code Name

Lists the name of the code as referred to from other places in this document.

# Syntax

<ID:Parameters>

Shows the general format for the code, including the 2-letter code ID and any parameters. Parameters may refer to other codes listed in this document. If the code is a paired code, the end code is displayed.

An icon appears next to the syntax for the code, indicating where the code may be used (either in the .FFF or in the .DEF file).

# Parameters

Lists all of the parameters for the code, including explanations.

# Description

Contains information on what the code is used for, as well as any special instructions for the code.

# Examples

At least two examples of each code are provided. The code being explained appears in Bold. The code is contained in the field Flat File Example Code.

Abstract

# Syntax

 <AS>Abstract Text</AS>

# Parameters

Abstract text may be any length. The text may contain carriage return codes <CR>, but no others.

# Description

Abstract is an Infobase Information Field designed to hold a short summary of the infobase. Please refer to Infobase Information for details.

Abstract is a paired code. All text to be included in the abstract must be contained between the begin code <AS> and the end code </AS>.

# Examples

<AS>The ancient Mayans developed an incredibly advanced society, rivaling modern day cultures for high honors. Included in their society was a highly accurate calendar and advanced writing and communication skills.</AS>

<AS>As we get closer to the future, life as we know it is changing less and less.</AS>

After Paragraph

# Syntax

 <AP:Value>

# Parameters

Value is a Decimal Number in inches (XX.xxx)

# Description

After Paragraph is a paragraph-based formatting code which allows you to insert additional spacing (Value) at the bottom of the paragraph. This code also reduces total storage space for the infobase by reducing the number of hard return codes or empty records used for spacing purposes in the infobase.

After Paragraph is additive with Before Paragraph (the two codes may be combined to increase the space between paragraphs).

# Examples

<AP:0.25>This paragraph would add ¼ inch of space between the bottom of this paragraph and the paragraph following.

<AP:0.5>This paragraph would insert an additional ½ inch of space at the bottom of it.

Attributes

# Description

Attributes are character-based formatting codes which are used to add visual emphasis to areas of text within the infobase. Character-based codes may be a single code or a pair of codes. A single code either turns the attribute ON or OFF. A pair of codes turns the attribute ON or OFF and then returns the attribute to whatever it was before it was turned on (see the following examples).

Attributes may be nested or overlapped, if necessary, to achieve any number of desired effects.

Five Attribute codes are available for display:

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | ON | OFF | DEFAULT |
| Bold | <BD+> | <BD-> | <BD> |
| Italic | <IT+> | <IT-> | <IT> |
| Hidden | <HD+> | <HD-> | <HD> |
| Strikeout | <SO+> | <SO-> | <SO> |
| Underline | <UN+> | <UN-> | <UN> |
| Double Underline | <UN:2+> | <UN-> | <UN> |

Hidden text is not normally visible in the infobase but is indexed and is searchable. (Hidden text may be viewed in the infobase if the appropriate options are set).

Three other codes are supported in the infobase and in flat file. These codes are provided to maintain compatibility with various word processing systems. They do not affect the display of the information in the infobase, but may be imported to or exported from the infobase. These codes are:

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | ON | OFF | DEFAULT |
| Condensed | <CD+> | <CD-> | <CD> |
| Outline | <OU+> | <OU-> | <OU> |
| Shadow | <SH+> | <SH-> | <SH> |

Note: When defining these attributes in a Style, you must indicate whether you want each available attribute to be ON or OFF. Those which are not specified are assumed to be DEFAULT. (Those that are not supported under a particular application need not be specified.) See any of the Style Definitions for details and examples.

# Examples

Note: These examples visually portray how these codes function. The attributes displayed in the infobase will not display the same way in your text editor (that is, text following a <BD+> code will not appear in bold in the text editor).

<BD+>**Single codes affect all text from the code to the end of the paragraph (so all of this should appear in Bold).**

<IT+>*You can turn a code on (all of this section is italicized)<*IT-> and then turn it off (all of this piece is now normal text). <IT>Then, whenever you want (within the same paragraph), you can go back to the default state for the paragraph style (usually, just plain text).

<UN->Character attribute codes<UN+> may also be <BD+>**nested**<BD> and overlapped to achieve special effects.<UN> How you use them is up to you.

Author

# Syntax

 <AU>Author Text</AU>

# Parameters

Author text may be any length. The text may contain carriage return codes <CR>, but no others.

# Description

Author is an Infobase Information Field designed to hold the name of the author of the infobase. The author may be a person, a department, an organization, or a company. Please refer to Infobase Information for details.

Author is a paired code. All text to be included in the abstract must be contained between the begin code <AU> and the end code </AU>.

# Examples

<AU>James A. Mosher, Esq.</AU>

<AU>Folio Writing Department, NextPage, Inc.</AU>

Background Color

# Syntax

 <BC:Red,Green,Blue,DC>

# Parameters

Red is an Integer between 0 and 255

Green is an Integer between 0 and 255

Blue is an Integer between 0 and 255

DC is an OPTIONAL parameter. If specified, the system default color is used.

# Description

Background Color is a character-based formatting code which changes the color of the screen behind text.

Background Color codes affect all text following the code up to the next Background Color code or the end of the paragraph, whichever comes first.

Background colors are defined using combinations of Red, Green, and Blue. Values for each color component may be from 0 to 255. 0 indicates an absence of color and 255 indicates maximum color. If all three values are set to 0, the color Black is created. If all three values are set to 255, the color White is created. For additional color combinations, see the RGB color chart.

DC (Default Color Option) is an optional parameter which sets the background color to the current system default background color (as specified through the Windows' Control Panel or the DOS Color option in the Preferences dialog). The two letter code for this option is DC. If this parameter is not included, the RGB color is used.

<BC> used without any RGB parameters is a Default Code. This code ends the current background color and reverts back to the previous background color. This option is provided to save space in the flat file and to permit styles to function correctly.

See the appendix RGB Color Combinations for a list of the RGB values for the standard colors available through the Folio Views dialogs.

Note: Since MS Windows does not allow dithered colors to be used for text foreground or background colors (it forces a pure color), you should not specify dithered colors for use in the infobase if any of your users may look at the infobase on the Windows platform. Pure colors use only 255, 128, and 0 values in any of the RGB positions. You may use a dithered color only for the border line colors.

# Examples

<BC:0,0,0>The background color for this text is black.

<BC:255,0,0>This text has a red background. <BC:100,0,100> But this text has a purple background. <BC>And this text should revert to the Default background (usually white).

<BC:0,0,0,DC>This text uses the system default background color for text (whatever that color happens to be set to on the user's system). This allows you to set your background to match the background color of any user's screen (not all are the same).

Before Paragraph

# Syntax

 <BP:Value>

# Parameters

Value is a Decimal Number in inches (XX.xxx).

# Description

Before Paragraph is a paragraph-based formatting code which allows you to insert additional spacing (Value) at the top of the paragraph. This code also reduces total storage space for the infobase by reducing the number of hard return codes or empty records used for spacing purposes in the infobase.

Before Paragraph is additive with After Paragraph (the two codes may be combined to increase the space between paragraphs).

# Examples

<BP:0.25>This paragraph would have ¼ inch of space between the bottom of the preceding paragraph and itself.

<BP:0.5>This paragraph would insert an additional ½ inch of space at the top of it.

Bold

 See Attributes.

Bookmark

# Syntax

 <BK:"Bookmark name">

# Parameters

Bookmark name may be up to 127 characters long (including spaces).

# Description

Bookmarks are used to hold your place or mark areas of interest in an infobase. Each bookmark has its own name for easy reference. In the finished infobase, bookmarks function similar to jump links from a dialog box. When a bookmark is selected, your view of the infobase is immediately shifted to the marked location.

Bookmark codes should be placed at the point you wish to go to when the Bookmark is selected.

# Examples

<BK:"Chapter 4">

<BK:"April 15, 1992 research notes">

Border

# Syntax

 <BR:Side:Width,Inside space,FC:Line Color>

The parameters for the Border code may be listed two ways. Each side may be listed individually with separate border codes, or one border code may list all of the sides to be bordered.

If the sides are listed within the same code, then the parameters for each side must be separated by commas. Any number of sides may be specified within a single code.

See the Examples, below, for more information on these options.

# Parameters

Side specifies which side of the border the following codes modify. Side may be one of the following:

RT Right

LF Left

TP Top

BT Bottom

AL All Sides

Width is a Decimal Number in Inches (XX.xxx). Hairline borders are 0.013 inches wide.

Inside space is a Decimal Number in Inches (XX.xxx) representing the distance from the border to the interior text or object

FC:Line Color is optional. Line color is the RGB color combination for the border color. The line color must be preceded with FC (foreground color). If this parameter is blank, the line color is the system default foreground color (usually black).

# Description

Border is a Paragraph-based formatting code which provides a frame for paragraphs and table cells. Paragraphs and table cells may be bordered on any number of sides, and each side may have its own width, spacing, and color. Fill color for bordered paragraphs and table cells are handled by the Shade code.

Side specifies which side of the border is being described. Separate side information by commas. You only need to specify the sides of the border you wish to use. (For example, if you wanted a top and bottom border, you would not need to specify details for the left and right borders.) If you use the AL (all sides) option for the sides, you may not specify any other side information.

Width specifies the width of the line which borders the paragraph or table cell. Hairline borders measure 0.013 inches. 0 indicates no line.

Inside Space indicates the distance between the paragraph border and the text contained within the border. Recommended space is 0.03 to 0.1 inches. Maximum recommended space is 0.75 inches. (Use a combination of inside space and a 0 width border to extend shading beyond the edge of text without specifying a border.)

Color specifies the color of a line in the border (each line may have its own color). Specified in combinations of Red, Green, and Blue. Default Color is flagged with DC (see see Color).

See the appendix RGB Color Combinations for a list of the RGB values for the standard colors available through the Folio Views dialogs.

# Examples

<BR:AL:0.015,0.03,FC:0,255,0>This example creates a bright green border around all sides of the paragraph or table cell. The border is 0.015 inches wide and is 0.03 inches from the inside text.

<BR:TP:0.015,0.03,FC:255,0,0><BR:BT:0.1,0.03,FC:0,0,128>This example places a border on the top and bottom of the paragraph or table cell. The top border is red and is 0.015 inches wide. The bottom border is blue and is 0.1 inches wide. Note that the different side options are included in two separate border codes. This is an option (see the next example for another method of specifying borders).

<BR:LF:0.1,0.1,TP:0,0.1,RT:0,0.1,BT:0,0.1><SD:192,192,192>This example places a single black border on the right side of the paragraph or table cell and creates a gray shade for the rest of the paragraph or table cell. The Top, Right, and Bottom sides of the border have no width, but do specify inside spacing. This additional spacing is filled by the Shade code and provides a cleaner appearance for the shading. Note that no color is specified for the lines; the system default text color is used (usually black).

Carriage Return

# Syntax

 <CR>

# Description

Carriage Return is a special character formatting code used to create a new line without starting a new paragraph. Carriage Return codes are most useful for creating tabular columns as a single paragraph, or for creating lists of items as a single paragraph. Compare with the Hard Return <HR> code.

Note: Flat file 3.0 allowed you to use a CRLF (Carriage-Return-Line-Feed) in text files and would interpret them as a <CR> code. In all versions of flat file from 3.1 on, this is no longer the case. If you wish to have a carriage return (line break) in your infobase, you must include the <CR> code.

# Examples

This line is divided<CR>into five distinct<CR>lines in the infobase<CR>and are all in<CR>one paragraph.

Carriage return codes<TB>can be used with tab codes<CR>to create tabular<TB>columns in the infobase.

Character

# Syntax

 <CH:ANSI Value (Decimal)>

<CH:$ANSI Value (Hex)>

# Parameters

ANSI Value is taken from the Windows ANSI character set. Characters may be translated to another character depending on the font you are using. If the infobase is to be used on both the Windows and Macintosh platforms, ensure that that character can be displayed on both (add the character to an infobase on Windows and see if it displays on the Macintosh).

# Description

Character is a special character formatting code used to add ANSI characters to flat files. This code is supported only on import; the code is not exported from the infobase (instead of this code, the actual character is exported, although the character will not display correctly in most DOS text editors).

Use either the decimal value for the ANSI character (32 - 255) or the Hex value ($20 - $FF). Hex values must be preceded by a dollar sign ($).

# Examples

To insert an em dash, use either <CH:151> or <CH:$97>.

To insert a section symbol, use either <CH:167> or <CH:$A7>.

Character Based Formatting

# Description

 Character-based formatting affects the infobase at the character level. The appearance of single characters, words, phrases, sentences, or entire paragraphs may be changed by character-based formatting codes. Compare with Paragraph-based formatting codes, which affect the structure and positioning of entire paragraphs.

Character Based Formatting codes may only be used in the Document file or in Text, Level, or Paragraph Style Definitions, and include the following:

|  |
| --- |
| Attributes |

Bold

|  |
| --- |
| Italic  Hidden  Strikeout  Underline |

<BD+>

|  |
| --- |
| <IT+>  <HD+>  <SO+>  <UN+> |

<BD->

|  |
| --- |
| <IT->  <HD->  <SO->  <UN-> |

<BD>

|  |
| --- |
| <IT>  <HD>  <SO>  <UN> |
| Character Face | <FT:Facename,Family> | | | |
| Character Size | <PT:Size> | | | |
| Background Color | <BC:Red,Green,Blue,Default Color Option> | | | |
| Foreground Color | <FC:Red,Green,Blue,Default Color Option> | | | |
| Subscript | <SB:Offset> . . . </SS> | | | |
| Superscript | <SP:Offset> . . . </SS> | | | |

When defining styles using the Attribute codes (Bold, Italics, etc), you must specify which codes you want to have ON and which you want to have OFF for that style. See the style definition sections for more details and examples.

When paired codes (Superscript or Subscript, etc) are used in style definitions, only the first part of the code needs to be included in the definition (for example: <ST:Example,CS,**SB:3**>. (Note that the default codes, which do not use the + or -, are not allowed in level or paragraph style definitions.)

See the appendix RGB Color Combinations for a list of the RGB values for the standard colors available through the Folio Views dialogs.

Note: Windows does not allow dithered colors to be used for text foreground or background colors (it forces a pure color). Pure colors use only 255, 128, and 0 values in any of the RGB positions.

Character Style Application

# Syntax

 <CS:"Name"> . . . </CS>

# Parameters

Name may be up to 127 characters long. If non-alphanumeric characters are used, Name MUST be enclosed in quotes.

# Description

Character Styles are a paired character-based formatting code which may be used to quickly apply previously defined character-based formatting commands to text.

Character Styles are defined by name in the Definition File using Text Styles. The definition contains the character-based formatting codes which are used each time the style is applied.

Once a Character Style has been defined, it may be applied to text in the Document File. When the infobase is created, the style definition may be modified, if necessary.

Please note that Character Style is a paired code. All information to be affected by the style must be enclosed within the Beginning Character Style code <CS:"Name"> and the End Character Style code </CS>.

Character styles may NOT be nested or overlapped. Character styles, if not specifically ended with a </CS> code, are automatically ended at the end of the paragraph in which they are used.

# Examples

<CS:"Simple">This sentence has been marked with the Character Style named "Simple."</CS> The style must be defined in the Definition File.

This paragraph has <CS:"Funky">two character styles in it: "Funky" and "Nifty."</CS> Remember that <CS:"Nifty">character styles</CS> cannot overlap: when a begin character style code is found, any previously un-ended character style codes are automatically turned off.

Character Style Definition

 See Text Styles

Color

# Description

 Color is used in several different codes to change the appearance of text, links, borders, or graphical frames.

Color is always specified as a combination of Red, Green, and Blue. Color may include the Default Color option DC. This option causes the color to revert to whatever the system default foreground or background color is for that item (borders revert to the default background color, text reverts to the default foreground color, etc.)

See the appendix RGB Color Combinations for a list of the RGB values for the standard colors available through the Folio Views dialogs. If you use a color different than those specified in this list, the color will appear correctly in the infobase and will be listed as a Custom Color in the appropriate dialogs.

For more information on the specific uses of color in the infobase, refer to the following sections:

Background Color

Borders

Foreground Color

Highlighter Pens

Links

Objects

Shade

Command (Menu) Link

# Syntax

 <ML:Style name,"Command">  
 .  
 .  
 .  
</ML>

# Parameters

Style Name refers to a Link Style (LK) as specified in the Text Style <ST> code in the Definition File.

Command is the menu command to be run when the link is followed. The commands must be the same as the ones which appear in the US English version of the Customize dialog in Folio Views for adding a toolbar command or in the Command Link dialog for adding a command. (Other language versions of Folio Views can select the menu command using the specific language; however, the command will be stored in English in flat file.)

# Description

Command links allow you to run a Folio Views menu command from a link. Any toolbar command that you could access through the Customize dialog may be added to a command link, including those provided by custom extensions to Folio Views.

Please note that command link codes are paired codes. Text or other information to be marked as a command link launch point must be entirely enclosed by the begin command link code <ML> and by the end command link code </ML>

Note: When flat files are exported from Folio Views or Extract, the filter exports an <EL> code to end all links (rather than </ML>). Either code may be imported.

# Examples

This is a <ML:Command,"Advanced Query">sample command link<ML>. It opens the Advanced Query dialog when launched.

<ML:Command,"Go Back">Click here to go back.</ML>

Comments

# Syntax

 <CM> . . . </CM>

# Description

Comments are non-indexed areas of text. They may be stored with the infobase, but they are not visible from anywhere within the infobase. Comments serve as notes to yourself and to others who may work with the flat file. Comments may also be used to block off sections of the flat file which may cause errors during the creation process. Comments should be included in the Document File when exported from a finished infobase. (This may be used to allow you to store application specific codes for other applications in the infobase without corrupting the infobase.)

Comments may appear in either the Definition File or the Document File. However, only those in the Document File are stored in the infobase; Comment Codes should be used in the Definition File when building a flat file so you can remember what each code does, but these codes are not exported when an infobase is saved as a Folio flat file.

Please note that Comment codes are paired codes. Text or other information to be included in a Comment must be entirely enclosed by the Begin Comment Code <CM> and by the End Comment Code </CM>.

Comments are stored in the infobase if the Store record comments in infobase option is set in the Import Filter Options dialog. The comments may be written back to flat file if the Write comments to flat file option is set in the Export Filter Options dialog.

# Examples

<CM>This flat file is protected under copyright laws</CM>

<CM>Comments can be used to block out junk aldjfjklkj adkjohir9324908 sao05r 9904t from the rest of the infobase.</CM>

Data Link

# Syntax

 <DL:Style name,"Object Name">  
 .  
 .  
 .  
</DL>

# Parameters

Style Name refers to a Link Style (LK) as specified in the Text Style <ST> code in the Definition File.

Object Name is the name of the Data Link object. Data Link objects must be defined in the definition file using the Object Definition <OD> code and the DL object type.

# Description

Data Links allow you to link to documents or other data without specifying the application associated with the file. When the link is activated, Folio Views checks the registry for a file association (based on the file extension) and opens the appropriate application.

Please note that data link codes are paired codes. Text or other information to be marked as a data link launch point must be entirely enclosed by the begin data link code <DL> and by the end data link code </DL>

Note: When flat files are exported from Folio Views or Extract, the filter exports an <EL> code to end all links (rather than </DL>). Either code may be imported.

# Examples

This is a <DL:Data,"MyFile.doc">sample data link</DL>. It links to a file with a .DOC extension. Whatever application is associated with the .DOC extension will attempt to open the file when the link is followed.

<DL:Data,"index.htm">Click here to open an HTML document in your WWW browser.</DL>

Default Font

# Syntax

 <DF:FT:"Font Name;PT:Size">

# Parameters

Font is the name of the Font to be applied.

Size is a decimal number in POINTS (XX.xx) specifying the size of the font.

PT is an optional parameter.

# Description

Default Font allows you to set the font that will be used in dialog controls which access information in an infobase. The default font does not affect the labels in dialogs; it does affect the text you enter into a dialog (such as the Query dialog). For example, the font used in the word wheel control of the advanced query dialog can be set but the font used to render the static text labels of the advanced query dialog will not be modified.

Note: the default font setting effectively overrides the infobase view font. Thus the font set with this code will be used by the reference window, table of contents, and hit list when viewing an infobase which has a default font set with the default font code.

# Examples

<DF:FT:"Comic Sans MT">

<DF:FT:"Symbol";PT:9>

Default Partition

# Syntax

 <PR:"Level Name">

# Parameters

Level Name is a level defined in the infobase using the LN level definition code or LE Level Styles code.

# Description

Allows you to set the default search partition for all queries in the infobase (the Partition specifier in a query will override this default). The default partition is always one of the levels defined in the infobase.

The standard partition is the record. When you perform a search, the hits are reported as the number of records with hits. By changing the partition, the number of hits are reported as the number of cases, chapters, documents, or books with hits.

# Examples

<PR:"Chapter">

<PR:"Case">

<PR:"Deposition">

Default Query

# Syntax

 <DQ:"Query">

# Parameters

Query is the full query to be performed on the infobase whenever the infobase is opened. The Query may include scope settings, multiple operators, phrases, and wildcards. The Query must be included in quotes and may be up to 2000 characters in length.

Note: Full quotation marks ( " ) inside a Default Query must be doubled in flat file ( "" ).

# Description

Default Query applies the specified query to the infobase whenever the infobase is opened. This option allows you to force users to start at a particular location in the infobase, rather than at the beginning.

The default query always forces the view to display records with hits (rather than all records).

For more information on building queries, see Advanced Query.

# Examples

<DQ:"[level chapter/Chapter 12:gonzo magician">

<DQ:"me and ""the great one"">

<DQ:"one | two & three ~ (four ^ five) | run\*">

Definition Codes

# Description

 Definition Codes are those codes used to define styles, objects, levels, and fields which are used in the infobase. As all Definition Codes are placed in a separate file, two benefits are created:

1. Simple re-use of styles, objects, levels, and fields throughout multiple infobases.

2. Faster Creation times by localizing definitions.

Once a style, object, level, or field has been defined, it may be used repeatedly in the Document File.

Compare with Document Codes.

See the Flat File Code Index for a complete listing of the definition codes.

Definition Include

# Syntax

 <DI:"Filename">

# Parameters

Filename specifies a Definition File for the flat file. If the full pathname is not included, then the path of the .FFF file is assumed. If the flat file import filter is not able to locate the definition file, no styles, levels, or objects are imported into the infobase.

# Description

A Definition Include code should be placed at the TOP of the Document File. It specifies the location of the Definition File and provides a link to that file. Without this code in the Document file, none of the definitions are used and no styles, levels, fields, or objects are included in the finished infobase.

You may include several Definition Include codes in the flat file. All <DI> codes must precede the header/footer codes and the first <RD> code. You may even include additional Definition Include codes in a definition file. This allows you to create special definition files for highlighters, levels, objects, and fields separate from each other and include them wherever they are needed.

# Examples

<DI:"C:\FFF\CONFIG.DEF">

<DI:"SECURITY.DEF">

<DI:"HILITE.DEF">

Document Codes

# Description

 Document codes are those codes which may appear in the Document File.

Document codes are used to place formatting commands, links, and organizational codes (such as levels, records, and fields) into the infobase. Compare with Definition Codes.

See the Flat File Code Index for a complete listing of the document codes.

End Note Links

# Syntax

 <EN:Style Name,"Query",Width,Height,"Title"> . . . </EN>

# Parameters

Style Name refers to a Link (LK) style defined with the Text Style code <ST>. May be up to 127 characters long.

Query is the query performed when the link is followed. See Advanced Query for details on the Folio Views query syntax.

Width is optional. If used, the Height must also be specified. It is a Decimal number in inches (XX.xxx) which sets the width of the popup window in the infobase.

Height is optional. If used, the Width must also be specified. It is a Decimal number in inches (XX.xxx) which sets the height of the popup window in the infobase.

Title is optional. If used, it appears as the title for the popup. Title may be up to 127 characters long, including spaces. Title MUST be included in quotes.

If <EN> is used without the size or title parameters, default values are assigned. The default values are: Height = 2 inches; Width = 2 inches; Title = End Note.

# Description

End Note Links are used to display query results in a popup window. While they make most sense with a material often used as end notes in a book (including bibliographies, glossaries, tables of authorities, and citations), they can be used to display any searchable information in the infobase.

The End Note Link code marks the beginning of an End Note Link. All text or objects enclosed by the beginning and ending End Note Link codes serve as the link launch point.

Each link has a link style associated with it (defined using the Text Style code in the Definition File). The link style is used to mark the text selected to be the link launch point (objects are not affected by link styles), allowing you to consistently mark similar link types.

Note: When Flat files are exported from Folio Views or Extract, the filter exports an <EL> code to end all links (rather than </EN>). Either code may be imported.

# Examples

<RD>As Johnson states in his book, "No news is no news."<SP:0.139><EN:"End Note","Group Johnson",2,2,"Works Cited by Johnson">37</EN>

<RD>Popups can be as big as the infobase. <EN:"End Note","\*",4,4,"The Whole Infobase">This end note link proves it.</EN>

Field Application

# Syntax

 <FD:"Name"> . . . </FD:"Name">

# Parameters

Name may be up to 127 characters long. Name must be defined in the Definition File using the Field Definition Code <FE>.

# Description

This field code is used to mark text which is to be included in a field. The name of the field must correspond to a field which has been defined in the Definition File, and the information in the field must be of the type specified by the field (i.e. text should not be included in an integer field, nor should dates be included in a time field). For more information on defining fields, see Field <FE>.

Please note that Field codes are paired codes. Text or other information to be included in a field must be entirely enclosed by the Begin Field Code <FD:Name> and by the End Field Code </FD:Name>.

Text fields may be nested and may overlap. Text fields may overlap other field types.

# Examples

The last three words of this record are included in <FD:Example>the "Example" field.</FD:Example>

<FD:Lead><FD:Demo>This entire record is contained in the "Demo" field,</FD:Lead> and the first half is nested in the "Lead" field.</FD:Demo>

Field Definition

# Syntax

 <FE:"Name",Type,"Field Format",IX:Index Options,Character Based Formatting Codes>

# Parameters

Name may be up to 127 characters long. Note that Note and Popup are reserved names and should not be used for field names.

Type may be one of the following 5 codes:

|  |  |  |
| --- | --- | --- |
| Code | Explanation | Description |
| TX | Text | Contains alphanumeric text (letters and numbers). |
| DT | Date | Contains dates, such as July 20, 1968 or 8-3-6. |
| TM | Time | Contains time values, such as 12:34:56 PM. |
| IR | Integer | Contains integers. Decimal values are truncated for searching and comparison purposes. |
| FP | Floating Point (Real) | Contains floating point numbers (decimal numbers). Decimal numbers may be specified to 10 places. |

Field Format is optional. It is used to uniformly display field contents for query purposes. They do not affect the information display in the Document Window. See Field Formats for more information on the formats allowed for the various field types.

IX:Index Options is optional. The Index Option specifies how text within the field is indexed for searching. The index options may be a combination of the following codes, separated by commas.

|  |  |
| --- | --- |
| Code | Explanation |
| TF | Term Field indexing. Individual terms in the field are indexed separately in the Field's index. |
| PF | Phrase Field indexing. The entire contents of the field are indexed as a phrase in the Field's index. |
| TE | Term Enclosing indexing. Individual terms in the field are indexed separately in the infobase's index. |
| NO | Not indexed. Terms in the field will not be found if searched for in the infobase. (This option may not be used in conjunction with other options.) |
| PR | Proximity. All terms in a field primitive must be found within a single application of the field to register a hit. Most often used with TE and TF. |
| DT | Used only with Date fields. Controls how dates specified with a 2-number abbreviation (95, 96, 97, etc.) are handled. If this option is used, dates less than or equal to 49 are assumed to be after the year 2000; dates greater than or equal to 50 are assumed to be before the year 2000 (such as 1995). If this option is not used, all dates are assumed to be in the 1900s. |
| FP | Fast Phrase. Enables the field for fast phrase searching. |
| SW | Stop Words. Uses stop words to reduce the size of the index for fields when fast phrase is enabled. |

Bug Notice:

In versions of Folio prior to 4.5, the Folio flat file import filter would allow the TF, PF, and FP indexing options to be used together. The search results from an infobase built in this fashion would act as though the TF indexing option was not set. This would result in a corrupt infobase that would crash the Views applicaton or any Infoapp that attempted to perform maintenence on the data. Any infobase built with Folio versions 4.4 and lower using a combination of the TF, PF, and FP indexing options in any field definition is corrupt. It will have to be rebuilt from Folio Flat File using the new Folio 4.5 flat file import filter. You may also choose to simply remove any one of the options and build with versions of Folio prior to 4.5. Note that the PF and TF indexing options are not valid when used together; the TF option will be ignored.

Character Based Formatting Codes are optional. They are used to apply character-based attributes to the contents of the field. These codes include all codes as listed under Character Based Formatting. Individual codes must be separated by commas.

# Description

Use fields to organize information on the sub-record level. Information stored in fields may be searched independently of the rest of the infobase.

Information in fields may also be compared in a search. For example, you could search for all records containing dates before January 1, 1954.

Fields definitions require that you specify a name and a type. The name is used for reference and the type specifies what kind of data will be contained in applications of the field (this is used for searching only; there is no mechanism to validate data within a field application).

Fields definitions may also specify a field format (not applicable to text fields). Picture string are used to provide a common format for searching related types of information in the infobase, regardless of how the information is actually stored in the infobase. See Picture Strings, below, for additional information.

Finally, fields may also have five different indexing options associated with them. These options allow you to specify how the information in the field may be searched for in the infobase. See Indexing Options, below, for additional information.

Field Formats

Field Format allows you to specify how information in a specific field may be searched for in the infobase — regardless of how it appears in flat file. For example, you might have several different date formats in a Date field named "Accounts Receivable." By specifying a field format for the "Accounts Receivable" date field, you could search for ALL dates in the field with one query, rather than having to search for "1-5-94" and "Jan 5, 1994" (or any other possible variations).

If no Field Format is specified for a particular field, then a default field format is used for the field.

The Field Format is one of the specified templates (see the following list) for the field types which Views uses when searching the contents of those field types.

Field formats can be assigned to Time, date, integer, and real fields. The field format internally reformats the information to allow easy and consistent queries across different input formats. A default field format is assigned when you create one of these fields; you may change the format at any time.

The field format defines how the contents of a field appears in the Advanced Query dialog. Dates, times, and numbers can exist in different formats in the same infobase (such as January 12, 2002 or 01-12-02 or 12 JAN 2002). Creating a query to find all the different formats would be difficult without field formats. The field format causes the Folio Server to internally convert the contents of a field to the same format. This consistent format is then displayed in the Advanced Query dialog. The actual text within the infobase does not change.

Text

Text fields do not support field formats.

Time

Use the following key to understand the Time field formats:

|  |  |
| --- | --- |
| h | hours, measured using a 12-hour clock, not padded (1, 2, 3, etc.) |
| hh | hours, measured using a 12-hour clock, padded (01, 02, 03, etc.) |
| H | hours, measured using a 24-hour clock, not padded (1, 2, 3, etc.) |
| HH | hours, measured using a 24-hour clock, padded (01, 02, 03, etc.) |
| mm | minutes |
| ss | seconds |

The following field formats may be used for time fields:

|  |  |
| --- | --- |
| Field Formats | Example |
| h:mm | 6:19 pm |
| h:mm:ss | 6:19:11 pm |
| H:mm | 18:19 |
| H:mm:ss | 18:19:11 |
| hh:mm | 06:19 pm |
| hh:mm:ss | 06:19:11 PM |
| HH:mm | 18:19 |
| HH:mm:ss | 18:19:11 |

Date

Use the following key to understand the Date field formats:

|  |  |
| --- | --- |
| m | month, not padded (1, 2, 3, etc.) |
| d | day, not padded (1, 2, 3, etc.) |
| y | year, in four digits (1996) |
| mm | month, padded (01, 02, 03, ... 10, 11, 12) |
| dd | day, padded (01, 02, 03, ... 29, 30, 31) |
| yy | year, final two digits (96) |
| yyyy | year, in four digits (1996) |
| Mmm | abbreviation for month, mixed case (Jan, Feb, Mar, etc.) |
| Mmmm | month, mixed case (January, February, March, etc.) |

The following field formats may be used for date fields:

|  |  |
| --- | --- |
| Field Formats | Example |
| m/d/yy | 8/3/68 |
| mm/dd/yy | 08/03/68 |
| m/yy | 8/68 |
| m-d-yy | 8-3-68 |
| mm-dd-yy | 08-03-68 |
| mm-dd-yyyy | 08-03-1968 |
| Mmm-yy | Aug-68 |
| Mmm d, yy | Aug 3, 68 |
| Mmm d, yyyy | Aug 3, 1968 |
| Mmmm d, yy | August 3, 68 |
| Mmmm d, yyyy | August 3, 1968 |
| dd.mm.yy | 08.03.68 |
| d Mmmm, yyyy | 3 August, 1968 |
| yyyy | 1968 |
| yy-Mmm | 68-Aug |
| yyyy-mm-dd | 1968-08-03 |
| yyyy Mmmm d | 1968 August 3 |

Note: Not all of the formats that were available in Folio Views 3.1 are available for Folio Views 4.x.

In addition, the following date formats will be recognized if they appear in the text of the infobase and converted for display in the Advanced Query dialog (these cannot be used as field formats):

|  |  |
| --- | --- |
| m-d-yyyy | 8-3-1968 |
| mm-dd-yyyy | 08-03-1968 |
| yyyy-m-d | 1968-8-3 |
| m/d/yyyy | 8/3/1968 |
| mm/dd/yyyy | 08/03/1968 |
| yyyy/m/d | 1968/8/3 |
| yyyy/mm/dd | 1968/08/03 |
| dd.mm.yyyy | 03.08.1968 |
| d.m.yy | 3.8.68 |
| d.m.yyyy | 3.8.1968 |
| yyyy.m.d | 1968.8.3 |
| yyyy.mm.dd | 1968.08.03 |
| m-yy | 8-68 |
| m-yyyy | 8-1968 |
| m/yyyy | 8/1968 |
| Mmmm, yyyy | August, 1968 |
| Mmmm yyyy | August 1968 |
| d Mmm yyyy | 3 Aug 1968 |
| d Mmm, yyyy | 3 Aug, 1968 |
| d Mmmm yyyy | 3 August 1968 |
| d of Mmmm, yyyy | 3 of August, 1968 |
| dMMMyy | 3AUG68 |
| dMMMyyyy | 3AUG1968 |
| yyyy Mmm d | 1968 Aug 3 |
| yyyyMMMdd | 1968AUG03 |
| Mmm-yyyy | Aug-1968 |
| Mmm/yy | Aug/68 |
| Mmm/yyyy | Aug/1968 |

Integer

The following field formats may be used for integer fields:

The following field formats may be used for integer fields:

|  |  |  |
| --- | --- | --- |
| Field Formats | Example | Explanation |
|  | -1000 | Negatives represented by a minus sign. |
| , | -1,000 | Negatives represented by a minus sign. Comma used as thousands separator. |
| ( ) | (1000) | Negatives represented by parentheses. |
| ( , ) | (1,000) | Negative represented by parentheses. Comma used as thousands separator. |

Real Number

The following field formats may be used for real fields:

|  |  |  |
| --- | --- | --- |
| Field Formats | Example | Explanation |
| . | -1000.5 | Negatives represented by a minus sign. Period used as decimal separator. |
| .0 | -1001 | Rounded up or down as necessary (no decimal used). |
| ,. | -1,000.5 | Negatives represented by a minus sign. Comma used as thousands separator. Period used as decimal separator. |
| ,.0 | -1,001 | Rounded up or down as necessary (no decimal used). Negatives represented by a minus sign. Comma used as thousands separator. Period used as decimal separator. |
| ( . ) | (1000.5) | Negatives represented by parentheses. Period used as decimal separator. |
| ( .0 ) | (1001) | Rounded up or down as necessary (no decimal used). Negatives represented by parentheses. Period used as decimal separator. |
| ( ,. ) | (1,001.5) | Negatives represented by parentheses. Comma used as thousands separator. Period used as decimal separator. |
| ( ,.0 ) | (1,001) | Rounded up or down as necessary (no decimal used). Negatives represented by parentheses. Comma used as thousands separator. Period used as decimal separator. |
| $,.2 | -$10,000.50 | Negatives represented by a minus sign. Comma used as thousands separator. Period used as decimal separator. Two decimal places required (padded with zeros, if necessary). |
| ($,.2) | ($10,000.50) | Negatives represented by parentheses. Comma used as thousands separator. Period used as decimal separator. Two decimal places required (padded with zeros, if necessary). |
| .% | -46.5% | Negatives represented by a minus sign. Period used as decimal separator. Numbers are not converted to percentages (0.345 = 0.345%). |
| .0% | -47% | Rounded up or down as necessary (no decimal used). Negatives represented by a minus sign. Period used as decimal separator. Numbers are not converted to percentages (0.345 = 0.345%). |
| (.)% | (46.5)% | Negatives represented by parentheses. Period used as decimal separator. Numbers are not converted to percentages (0.345 = 0.345)%. |
| (.0)% | (47)% | Rounded up or down as necessary (no decimal used). Negatives represented by parentheses. Period used as decimal separator. Numbers are not converted to percentages (0.345 = 0.345)%. |

Note: The scientific notation formats available in Folio Views 3.1 are not supported in Folio Views 4.x.

Indexing Options

There are two sets of indexing options. One set applies to text fields only. The other set applies to date fields only.

Text Fields

Indexing options allow you to specify how information in a text field may be searched in the finished infobase. (The indexing options extend the functionality of the IX+ and IX- options provided in the 3.1 version of flat file.)

There are basically three types of indexing options: Field Indexing, Enclosing Indexing, and No Indexing. Field Indexing adds the terms in the field application to the field's index. This allows the terms to be found in a field search (such as Field X:dog). Enclosing Indexing adds the terms in the field application to the infobase index. This allows the terms to be found during a normal query (such as a search for the word dog). No Indexing turns indexing off completely; the terms in the field application cannot be found in any search.

In addition, you may specify how terms are indexed in the Field Indexing and Field Enclosing options. Terms may be indexed as individual terms and searched as individual terms (this is what is described in the preceding paragraph). Terms within a field application may also be indexed as a phrase. When terms are indexed as a phrase, then all of the terms in the field application are considered to be one word for searching purposes. For example, if International Business Machines is indexed as a phrase, the single term International Business Machines appears in the Word List in the Advanced Query dialog (rather than three separate terms).

Finally, text fields may also set a proximity for searches. If you use the Proximity (PR) option, then queries for terms in the field require that all of the terms exist in the same application of the field for a hit to register. For example, a search for [Field Presidents: Washington Adams Lincoln] would require that all three of these terms be found in the same field application. See Notes on the Proximity (PR) Indexing Option for more information.

These options may be used in conjunction for different effects. The default is Term Field (TF) and Term Enclosing (TE) (if no options are specified, these are used by default). This combination allows the terms in the field to be found both within the field and within the infobase as a whole.

Use all of the options (except the No Indexing (NO) option) to index terms both as phrases and as individual terms.

Use Term Field (TF) or Phrase Field (PF) alone to force the user to search in the field to find the term.

Using Term Enclosing (TE) alone does not make much sense but may be done. Using this alone allows users to find the information in the field in a general search of the infobase, but not when they search within the field. This option is normally used in conjunction with PF.

Note: While any combination of these options may be specified, the following are those allowed to be set from within Folio Views:  
  
 TF,TE Normal  
 PF,TE Phrase  
 No Unindexed  
 TF Field Only  
 PF Field Only as Phrase  
  
If no indexing options are set, the Normal is used (TF,TE).

Notes on the Proximity (PR) Indexing Option

Field proximity searching allows specific regions of an infobase to be marked up with a field and then to have searches restricted to individual instances of these proximity fields. Field proximity searching performs an unordered proximity search within the field. This functionality gives users the ability to find terms, phrases, etc. within an application of a field proximity. Historically, field searches have been somewhat confusing to some end users. By default, the search [Field Name: John Doe] does not necessarily find the terms John and Doe in the same application of the field Name. For example, consider the following example record:

<RD>A list of names: <FD:Name>John Smith</FD:Name>, <FD:Name>Jane Doe</FD:Name>, and <FD:Name>John Harris</FD:Name>

A normal field search for [Field Name: John Doe] would find a hit in this record, since the field search is actually resolved as [Field Name:John] & [Field Name: Doe].

With a proximity field, however, there would be no hits in this record, since the proximity requires that all terms be in the same application of the field. A search for [Field Name: John Smith] would result in a hit.

Note that overlapping applications of the same field are treated as overlapping applications rather than nested applications. For instance, the markup

<RD><FD:A>a b c<FD:A>d e f</FD:A>g h i</FD:A>

is interpreted as an application on a - f and another application on d - i. Because of this, the search [Field A:a i] would not return this record as a hit.

Date Fields

As we approach the year 2000, some dates stored in infobases may reference years after 1999. Since common date abbreviations often use just the last two letters of the year (such as 86, 87, or 88), there is a potential for confusion with dates after the year 1999. For example, does the date 10/27/29 refer to 1929 or 2029?

Using the DT indexing option for date fields, you can control how Folio Views interprets these dates for range searches.

When the DT indexing option is used, all two-digit years from 00 to 49 are assumed to be after the year 2000; all two-digit years from 50 to 99 are assumed to be in the 1900s.

If the DT indexing option is not used, all dates are assumed to be in the 1900s.

Special Note on Field Indexing Options

All fields have a default set of indexing options applied. These options appear when an infobase is exported. The default set is PF and TE. This default set may be removed but not modified for time, integer, and real number fields. The DT option may be added to date fields, if desired. The default set may be modified as necessary for text fields.

# Examples

<FE:"Weapons",TX,IX:PF,PE,UN+,FT:"Times New Roman",BC:0,128,255>

<FE:"Section Numbers",TX,IX:NO>

<FE:"Graduations",DT,"Mmm d, yyyy",IX:DT>

<FE:"Math Answers",FP,"$,.02",BD+,FC:0,255,0>

<FE:"Time",TM,"HH:mm:ss">

Flat File Include

# Syntax

 <FI:"Filename">

# Parameters

Filename specifies a flat file to be inserted into the current flat file. If the full pathname is not included, then the flat file is assumed to be in the same directory as the current flat file.

# Description

Flat File Include allows you to insert a separate flat file into an existing flat file. Use this code to create a master flat file which points to all definition files to be used, sets the header and footer for the infobase, and then lists all supplementary flat files to be included. Or, use this code to insert updated material into an existing infobase.

Note that this code will force a new record break in the infobase. The code following the Flat File Include code must be a record code (or another FI code).

Unlike previous versions of Folio flat file, definition include codes <DI> are valid within the included flat file.

# Examples

<FI:"new\_info.fff">

<FI:"F:\HOME\JOHNSON\FFF\CHARSETS.FFF">

Font Facename

# Syntax

 <FT:Font,Family,Character Set>

# Parameters

Font is the name of the Font to be applied.

Family is optional. Family may include:

|  |  |  |
| --- | --- | --- |
| Code | Family | Example |
| SR | Serif | Times New Roman |
| SN | Sans Serif | Arial |
| FX | Fixed Pitch | Courier 10 |
| SC | Script | Script (Script) |
| DV | Decorative | (no examples available) |

Character Set is optional and should only be used when accessing special characters not supported in the ANSI character set (see the appendix Character Sets). The Character set specifies what character set is to be used. The default is ANSI, but may be one of the following:

PC (IBM Character Set)

SY (Symbol Character Set)

# Description

A character-based formatting code, Font Facename (sometimes just called "Font") specifies the typeface which is used for a particular area of text. The Font Facename code changes the font to Font. The specified font is in effect until the next Font Facename code or until the end of the paragraph in which it occurs, whichever comes first.

Family refers to a two-letter code representing the generic family to which the Font belongs. This is an optional parameter in this code. If it is included, and if the specified font is not found on the system, another font currently on the system from the same font family is chosen. If the family is not specified and if the named font is not found, a close substitution is made.

Note: Most fonts (including Wingdings and Zaph Dingbats) belong either to the Serif (SR) or Sans Serif (SN) families. This may cause some problems if the font you are using in the infobase is not available on another users system. For example, a Wingdings bullet may appear as a letter rather than a bullet.

<FT> used without any Font or Family parameters is a Default Code. This code ends the current font and reverts back to the font in effect from any applied styles.

# Examples

<FT:"Times New Roman",SR>This paragraph is all in the Times New Roman font. If the font is not installed on a system using the infobase, another Serif font is chosen.

<FT:Arial,SN>This sentence is in the Arial font. <FT:Script,SC>But this sentence is in Script. Both sentences will substitute fonts from the same family if necessary. <FT>This last sentence will display in the Normal Level (which is usually Times New Roman).

Footer

# Syntax

 <FO>Footer Text and Formatting Codes</FO>

# Parameters

Footer Text and Formatting Codes may be any text or formatting codes normally found in the Document File and the special generated codes. This code (and the Header code, if used), must come after the Definition Include code <DI> and before the first record code <RD>.

# Description

The Footer Code allows you to store a Footer in the infobase. The footer appears only at the bottom of printed pages.

The Footer may contain simple text, formatted text (character and paragraph based formatting codes, including styles), and objects. If styles and objects are used in the Footer, they must be defined in the Definition File.

Multiple paragraphs may exist in a footer code, but not multiple records. Do not place any <RD> codes in the footer. Use paragraph breaks <HR> or line breaks <CR> if you need multiple lines in your footers.

Only one Footer may be used in any one Document File. The Footer Code (and, if used, the Header Code) must be placed immediately after the Definition Include <DI> code at the top of the Document File. All other Footer codes are ignored.

Please note that the Footer code is a paired code. All information (and other codes) which are to appear in the footer MUST be placed between the Begin Footer code <FO> and the End Footer Code </FO>.

# Examples

<FO><PS:Footer>This text (and the paragraph style named 'footer') should appear in the <IT+>footer<IT> for the infobase. Note that other character codes may also be <OB:Object,Bitmap,2,2>included, as well as objects.</FO>

<FO><JU:RT><BD+><FT:Arial><PT:10>Johnson Publishing Division, Inc. All Rights Reserved.<BD> <OB:"Corporate Logo",Metafile, 0.3,0.3></FO>

Foreground Color

# Syntax

 <FC:Red,Green,Blue,DC>

# Parameters

Red is an Integer between 0 and 255

Green is an Integer between 0 and 255

Blue is an Integer between 0 and 255

DC is an OPTIONAL parameter. If specified, the system default color is used.

# Description

Foreground Color is a character-based formatting code which changes the color of the text displayed on the screen.

Foreground Color codes affect all text following the code up to the next Foreground Color code or the end of the paragraph, whichever comes first.

Foreground colors are defined in combinations of Red, Green, and Blue. Values for each color component may be from 0 to 255. 0 indicates an absence of color and 255 indicates maximum color. If all three values are set to 0, the color Black is created. If all three values are set to 255, the color White is created. Additional color combinations are listed in the appendix RGB Color Combinations.

DC (Default Color Option) is an optional parameter which sets the foreground color to the current system default foreground color (as specified through the Windows' Control Panel). The two letter code for this option is DC. If this parameter is not included, the RGB color is used.

<FC> used without any RGB parameters is a Default Code. This code ends the current foreground color and reverts back to the previous foreground color. This option is provided to save space in the flat file and to permit styles to function correctly.

See RGB Color Combinations for a list of the RGB values for the standard colors available through the Folio Views dialogs.

# Examples

<FC:0,0,0>The foreground color for this text is black.

<FC:0,255,0>This text is green.<FC:100,0,100> But this text is purple.<FC:0,0,0,DC> And this text is in the system default foreground color.

Formatting

# Description

Formatting codes affect how the infobase looks. Formatting codes include:

Character Based codes

Paragraph Based codes

Styles

Tables

Please refer to the appropriate section for more information on these items.

Generated Codes

# Syntax

 The Generated Codes are as follows:

<GP> Generate Page Number

<GT> Generate Current Time

<GD> Generate Current Date

<GM> Generate Last Modification Date

<GI> Generate Infobase Title (from Infobase Information)

<GA> Generate Infobase Author (from Infobase Information)

<GF> Generate Infobase File Name

<GQ> Generate Current Query

Note: These codes may ONLY be placed inside of a Header or Footer.

# Description

Generated Codes are special character codes which are generated and updated in the infobase each time the infobase is printed. These codes allow you to include page numbers, the current time, the current date, and the infobase title, author, and file name in a printed header or footer.

# Examples

<FO>The current time is: <GT>, and the current date is <GD>. The name of the infobase is <GI>, and the full path and file name for the infobase is <GF>.</FO>

<FO><JU:CN>Page #<GP></FO>

<HE>The Query which was just performed was: <GQ></HE>

Group

# Syntax

 <GR:Name>

# Parameters

Name may be up to 127 characters long.

# Description

Groups are used to organize related records by topic. These categories may be searched independently or in conjunction with other groups, words, or phrases.

Multiple Group codes may be placed in any one record. Because of the dynamic (rather than static) approach to record groupings, records may belong to multiple topical groups simultaneously.

Group names may be from 1 to 127 characters long. If the name contains spaces or other non-alphanumeric characters, the name must be included in quotes.

Groups may be defined in the Definition file; usually, groups listed in the definition file are not applied anywhere in the document file (although defining a group in the definition file that is applied in the document will not cause any problems).

Note: Groups are record-based. If you place a group code in a record containing multiple paragraphs, all paragraphs in that record become members of the group.

# Examples

<GR:Dogs>German Shepherds have long been considered both man's best friend and man's greatest nightmare.

As this record demonstrates, Groups may <GR:Demo>be placed anywhere within a record (following any paragraph-based codes, of course!) They may also contain multiple topics. <GR:Example> <GR:"Multiple Groups">

Hard Return

# Syntax

 <HR>

# Description

Hard Return functions the same as a paragraph break in most word processors: it inserts a Carriage-Return-Line-Feed (CRLF) and starts a new paragraph without starting a new record. Use Hard Return codes to separate multiple paragraphs within a single record.

# Examples

This paragraph and the one following it have been separated by a Hard Return code.<HR>This sentence is a new paragraph, even though it does not appear to be so in the Document File.

Hard Space

# Syntax

 <HS>

# Description

Hard Space inserts a non-breaking space into the infobase. Words on either side of a hard space remain on the same line as each other, regardless of the screen size or printed page format. Note that the hard space code should replace the space character between words.

Note that this code is not supported by the Folio Views 4.x client.

# Examples

Sometimes, you want to keep some terms together, such as International<HS>Business<HS>Machines or Folio<HS>Views.

Header

# Syntax

 <HE>Header Text and Formatting Codes</HE>

# Parameters

Header Text and Formatting Codes may be any text or formatting codes normally found in the Document File and the special generated codes. This code (and the Footer code, if used), must come after the Definition Include code <DI> and before the first record code <RD>.

# Description

The Header Code allows you to store a header in the infobase. The header appears only at the top of printed pages.

The Header may contain simple text, formatted text (character and paragraph based formatting codes, including character styles & paragraph styles), objects, and generated codes. If styles and objects are used in the Header, they must be defined in the Definition File.

Multiple paragraphs may exist in a header code, but not multiple records. Do NOT place any <RD> codes in the footer. Use paragraph breaks <HR> or line breaks <CR> if you need multiple lines in your headers.

Only one Header may be used in any one Document File. The Header Code (and, if used, the Footer Code) must be placed immediately after the Definition Include <DI> code at the top of the Document File. All other Header codes are ignored.

Please note that the Header code is a paired code. All information (and other codes) which are to appear in the Header MUST be placed between the Begin Header code <HE> and the End Header Code </HE>.

# Examples

<HE><PS:Header>This text (and the paragraph style named 'header') appears in the <IT+>header<IT> for the infobase. Note that other character codes may also be <OB:Object,Bitmap,2,2>included, as well as objects.</HE>

<HE><JU:RT><BD+><FT:Arial><PT:10>Johnson Publishing Division, Inc. All Rights Reserved.<BD> <OB:"Corporate Logo",Metafile, 0.3,0.3></HE>

Hidden

 See Attributes.

Highlighter Pen Application

# Syntax

 <PN:Pen Name> . . . </PN>

# Parameters

Pen Name may be up to 127 characters long. The name must be defined using the Highlighter Pen Definition code <PD> in the Definition File.

# Description

Highlighters (pens) provide a quick and easy way to mark text with character-based formatting codes and, at the same time, place them into a searchable text field.

Highlighters are very similar to highlight pens which many readers use to mark passages of interest in books and magazines. Often similar colors are used to mark similar topics. With Highlighters, you may specify not only the color of the highlighter, but also the font type and size, as well as any attributes (such as bold) to mark the text.

The attributes associated with a Highlighter must be defined in the Highlighter Pen Definition <PD> code (see Highlighter Pen Definition for more information on defining Highlighter Pens).

Please note that Highlighter application codes are paired codes. Text or other information to be included in a Highlighter must be entirely enclosed by the Begin Highlighter Code <PN:Name> and by the End Highlighter Code </PN>. Highlighter codes may NOT be nested or overlapped. (Field codes, which may also be formatted, may be overlapped.)

See RGB Color Combinations for a list of the RGB values for the standard colors available through the Folio Views dialogs.

# Examples

<PN:"Murder Weapons">This entire sentence is marked with the Murder Weapons highlighter pen and placed in a text field of the same name.</PN>

<PN:Example>This paragraph is partially marked by two different highlighter pens.</PN> Note that they <PN:Demo>do NOT overlap</PN> each other at all.

Highlighter Pen Definition

# Syntax

 <PD:"Pen Name",Character Based Formatting Codes>

# Parameters

Pen Name may be 1 - 127 characters long, including spaces.

Character Based Formatting Codes are optional. They are used to apply character-based attributes to the contents of the highlighter pen. These codes include all codes as listed under Character Based Formatting. Individual Codes must be separated by commas.

# Description

Once a Highlighter Pen has been defined, it may be applied to text (similar to a Character Style). The text is formatted according to the character based formatting attributes in the Highlighter Pen definition and is placed in a searchable field. For more information on applying Highlighter Pens, see Highlighter Pen Application.

# Examples

<PD:Example,BD+,IT+,UN-,FT:Arial,PT:12>

Defines a Highlighter Pen named "Example." Text marked with the Highlighter Pen should display in Bold and Italic text (Underline is turned OFF), and is in the Arial 12 point font.

<PD:"Murder Weapons",UN+,FC:128,0,0>

Defines a Highlighter Pen named "Murder Weapons." Text marked with this Highlighter Pen should display in Underlined Red text.

Hit List

# Syntax

 <HL:Hit List Options>

# Parameters

Hit List Options include:

HR:Width — Hit Reference

RR:Width — Relevance Rank

PR:Width — Partition

FD:Name:Width — Field

Width for all options is measured in screen units; default values are 100.

Options may be listed in any order. Multiple fields may be specified, but only one each of the other codes are valid.

Note: The HL code must be listed after the field definitions for any fields referenced in the HL code.

# Description

Hit List is used to represent the settings for the Hit List pane in Folio Views. Settings for the Hit List pane may be saved in an infobase; this code allows those settings to be retained when the infobase is exported or imported.

The options are a direct representation of the options that may be set in the Hit List Properties dialog. See Hit List Properties for more information on these options.

# Examples

<HL: HR:122; FD:Number,122; FD:Abstract,122; RR:57; PR:123>

<HL: HR; FD:Number,122; FD:Severity; FD:Priority,122; FD:Feature,122; FD:Abstract; RR:57; PR>

Hypergraphic File Format

Although the hypergraphic file format is not technically a flat file code, it is generated when a file is exported to flat file. This file format also allows you to build hypergraphic files programmatically, if desired.

Note that this file format is only supported with the Hypergraphic Object Handler, version 2. The object handler reads version 1 hypergraphics and converts them automatically to the version 2 format if the object is edited and saved.

# Syntax

FOLIOHG2  
<IMG SRC="name of graphic image">  
<AREA SHAPE=shape COORDS=coords HREF=link information>  
...  
<AREA SHAPE=shape COORDS=coords HREF=link type?link data>

# Parameters

FOLIOHG2 is a signature that defines the file as a hypergraphic file. It must be the first 8 characters of the file (no leading whitespace).

IMG identifies the name of the graphic to be used as a hypergraphic. May include the full path to the image. Note that the name must be included in quotes.

AREA defines a single link region on the graphic. Multiple AREA parameters may exist in the same hypergraphic file. AREA has three primary components: SHAPE, COORDS, and HREF.

SHAPE may be RECT, ELLIPSE, ROUNDRECT, or POLYGON.

COORDS represent the coordinates on the graphic for the link region. RECT, ELLIPSE, and ROUNDRECT require four coordinates (two for upper left vertex, two for lower right vertex). POLYGON takes a variable number of coordinates (one pair for each vertex). COORDS is measured in pixels, twips (100t), or percentages (1.23%). The first coordinate in a pair is the number of pixels from the left; the second is the number of pixels from the top.

HREF specifies the link information. The format for the link information is link type?link data. HREF information must be enclosed in quotes. See the following table for more information:

|  |  |  |
| --- | --- | --- |
| Link Type | Link Data | Example |
| Jump | name of jump destination | "Jump?appendix" |
| Query |

optional records with hits flag

|  |  |
| --- | --- |
| query to perform | "Query?(RecordsWithHits=TRUE)[Group Animals]" |
| Object |

optional fit to window flag

|  |  |
| --- | --- |
| name of object to display | "Object?(FitToWindow=TRUE)Cat Picture" |
| Program | program to run | "Program?notepad" |
| Data | data object in infobase to display | "Data?MyZipFile" |
| Command |

menu command to execute

|  |  |
| --- | --- |
| (see Command Names for more information) | "Command?Advanced Query" |
| Web | URL to execute | "Web?www.nextpage.com" |
| Named Popup | name of popup to display | "Named Popup?My Definition" |
| Inter Infobase Jump | path to infobase#destination name | "Inter Infobase Jump?c\:\\myfiles\\myinfo.nfo#Appendix A" |
| Inter Infobase Query | path to infobase#(optional records with hits)query to perform | "Inter Infobase Jump?c\:\\myfiles\\myinfo.nfo#(RecordsWithHits=TRUE)[Group Dogs]" |

Note: The last line of the .hgx file must be followed by a line break. If the line break is not used, then the last link in the .hgx file is ignored.

# Description

The hypergraphics file format describes the link regions on a specified graphic file. Additional information about the format includes:

• Notes on Inter-Infobase Links

• Coordinates saved as percentages to allow for resizing

• Polygon areas not supported in hypergraphics editor

Notes on Inter-Infobase Links

When hypergraphics containing inter-infobase links are exported, they list the infobase name followed by the infobase ID. The ID is a set of four long integers separated by periods. This ID is not required when importing the hypergraphic or when creating a hypergraphic outside the editor.

For example, exporting a hypergraphic might produce the following line in the HGX file:

<AREA SHAPE=RECT COORDS=10%,10%,20%,50% HREF="Inter Infobase Jump?c\:\\temp\\myfile.nfo:B9030A8D.67B12A1D.0.0#Destination 1">

This may be modified for import to be:

<AREA SHAPE=RECT COORDS=10%,10%,20%,50% HREF="Inter Infobase Jump?c\:\\temp\\myfile.nfo#Destination 1">

Also note that the full path to the infobase does not need to be stored in the link. You may store only the name of the infobase and rely on the search path used by Folio Views to find and open the infobase (typically, you will want to modify the NfoPath registry entry).

Coordinates saved as percentages to allow for resizing

When you edit and save a hypergraphic object in the hypergraphic editor, the all coordinates are written out as a percentage (accurate to two decimal places). The percentage value allows the hypergraphic image to resized in Folio Views without distorting the link areas.

Polygon areas not supported in hypergraphics editor

Polygon areas are not supported in the hypergraphics editor. You cannot change the link type, link data, or link region for polygon areas in the editor. These changes may only be made in the hypergraphics file (\*.hgx).

Polygons must have at least three pairs of coordinates. The first and last coordinates do not have to be the same (the region is automatically closed if the first and last points are not the same).

Indents

# Syntax

 <IN:LF:Left,RT:Right,FI:First>

# Parameters

Left is a Decimal number in inches (XX.xxx) specifying the Left indent for the entire paragraph.

Right is a Decimal number in inches (XX.xxx) specifying the Right indent for the entire paragraph.

First is a Decimal number in inches (XX.xxx) specifying the indent for the First Line of the paragraph.

Indent is a paragraph-based formatting code which allows you to set Left, Right, and First Line indents.

# Description

Left indents shift the entire paragraph a specified distance away from the left margins.

Right indents shift the entire paragraph a specified distance away from the right margins, whether fixed or not.

First line indents shift only the first line of the paragraph. First Line indents may be negative (less than 0) to create hanging indents. (The absolute value of a negative first line indent may not exceed the absolute value of the left indent. For example, a Left indent of 0.5 inches and a First line indent of -1 inch would cause an error since you would be trying to shift the First line indent behind the left margin.)

# Examples

<IN:LF:0,RT:0,FI:0.25>This is a common indent: the left and right indents are set to 0, and the first line is set to 0.25 inches, creating an appearance similar to most books.

<IN:LF:0.5,RT:0,FI:-0.5>This creates a "hanging indent," where the first line of text hangs out to the left of the rest of the paragraph.

Infobase Information

# Description

 Infobase Information is a set of six fields designed to hold infobase-specific information. All Infobase Information codes should be placed in the Document File (for backwards compatibility with the Folio 3.x flat file, these codes may also appear in the Definition file). Text contained in the Infobase Information fields are not included in the main body of the infobase, but are available from the Folio Views Properties dialog for all infobase users to see.

The following Infobase Information codes may be used to store infobase-specific information. After each code is the maximum number of characters which may appear in each infobase information field.

|  |  |  |
| --- | --- | --- |
| Abstract | <AS>Abstract text</AS> | No limit |
| Author | <AU>Author text</AU> | No limit |
| Remark | <RM>Remark text</RM> | No limit |
| Revision Date | <RE:Date text> | 45 |
| Subject | <SU>Subject text</SU> | No limit |
| Title | <TT:"Title text"> | 127 |

Only one code for each field may be used per infobase. Additional codes are ignored.

Infobase Information codes may be placed anywhere in the Definition File. It is recommended, though, that all be placed in a similar location (such as the top or bottom of the file).

Italic

 See Attributes.

Jump Destination

# Syntax

 <JD:Destination Name>

# Parameters

Destination Name may be up to 127 characters long and MUST coincide with a Destination Name in Jump Link code <JL> in order for the link to function. Destination Name must be unique to the infobase.

# Description

Jump Destination codes mark the end-point of a Jump Link. The position of the Jump Destination code within text or before or after graphics or other objects is the position of the cursor after the link is performed.

The Destination ID must match a Destination Name specified in a corresponding Jump Link code (either in the current infobase or in another infobase) for the link to function.

Two Destination Names may not be placed immediately next to each other (at least one character must separate the two). If you need to link to the same location, use the same Jump Destination Name.

# Examples

The middle of this sentence marks <JD:"End Link One">the end of a Jump Link.

<JD:Example>The cursor will land at the beginning of this sentence when the link is executed.

Jump Link

# Syntax

 <JL:Style name,Destination name,"Infobase name">  
 .   
 .   
 .   
 </JL>

# Parameters

Style Name refers to a Link (LK) style defined with the Text Style code <ST>. May be up to 127 characters long.

Destination Name identifies the end point of the link. It MUST correspond with a Destination ID in a Jump Destination code <JD>.

Infobase Name is optional. It specifies a value UIL for the infobase. If not specified, the current infobase is assumed.

# Description

Jump Link codes mark the beginning of a Jump Link. All text or objects enclosed by the beginning and ending Jump Link codes serve as the link launch point. Not only can jump links link to destinations in the same infobase as the link, but also to destinations in different infobases.

Each link has a link style associated with it (defined using the Text Style code in the Definition File). The link style is used to mark the text selected to be the link launch point (objects are not affected by link styles), allowing you to consistently mark similar link types.

The Destination Name must be a unique alphanumeric string from 1 to 127 characters long. If non-alphanumeric characters (such as spaces or punctuation) are used in the Destination Name, the entire name must be included in quotes.

Jump Links can exist between information in two infobases. If you wish to link to information in a separate infobase, include the Infobase Name in the Jump Link code. Infobase name must be a valid UIL (universal infobase locator), which is generally just the infobase file name. See the appendix Understanding the Folio Search Path for more information on the UIL.

Note: When flat files are exported from Folio Views or Extract, the filter exports an <EL> code to end all links (rather than </JL>). Either code may be imported.

# Examples

<JL:Jump,"End Link One">This entire paragraph functions as a jump link within the current infobase. The paragraph is also marked with the link style named "Jump," and should jump to the Jump Destination code containing the destination ID "End Link One."</JL>

Only the word <JL:Test,Demo,"c:\Folio4\nfo\demo.nfo">LINK</JL> functions as a jump link. This link connects to a different infobase (demo.nfo).

This also links out to a <JL:Test,Demo,"\Folio4\samples\legal.nfo"> different infobase</JL> using a partial path to the infobase.

Justification

# Syntax

 <JU:Justification>

# Parameters

Justification may be one of the following (examples of each type of justification are also provided. The vertical lines represent the margins):

Justification Description

LF Left

Keeps the text aligned with the left margin. The right side of the text appears ragged (this paragraph is left justified).

CN Center

Aligns the text around a point half-way between each margin. All text in the paragraph is centered on the screen and the printed page (this paragraph is center justified).

RT Right

Keeps the text aligned with the right margin. The left side of the text appears ragged (this paragraph is right justified).

FL Full

Keeps the text aligned with both the left and right margin. Neither side of the text appears ragged (this paragraph is full justified).

# Description

Justification is a Paragraph-based formatting code which allows you to adjust the alignment of text with the margins. Justification for any one paragraph may be set to any one of four settings. Each setting has its own two-letter code.

Justification is also used for paragraphs within table cells.

# Examples

<JU:RT>This paragraph has right justification.

<JU:CN>This paragraph is center justified. (Center justification can be good for headings and titles.)

Keep Together

# Syntax

 <KT>

# Description

Use Keep Together to affect how paragraph text is printed.

Keep Together forces all of the text in the paragraph to print on the same page (provided the page is large enough to hold the paragraph).

Keep Together is a paragraph formatting code. It may be used alone or as part of a paragraph or level style.

# Examples

<RD><KT>All of the text in this paragraph will print on the same page. This option does not affect how the information is displayed on the screen.

Keep with Next

# Syntax

 <KN>

# Description

Use Keep Next to affect how paragraph text is printed.

Keep Next forces the paragraph to be printed with the paragraph that follows it. Both of the paragraphs will print on the same page (provided the page is large enough to hold the paragraph).

Keep Next is a paragraph formatting code. It may be used alone or as part of a paragraph or level style.

# Examples

<RD><KN>The text of this paragraph will print with the text of the following paragraph. This option does not affect how the information is displayed on the screen.

Level Application

# Syntax

 <LV:"Level Name">

# Parameters

Level Name must be one of the levels defined in the definition file.

# Description

Use the level application code to set the levels used in the infobase. The level application code allows conversion routines to specify the level for a record at any point in the record, rather than in the record code (which is the standard method for applying levels).

Only one level application code is allowed in any single record. It may not be used if the record code <RD> also specifies the level.

Note also that this code is supported only on import. It is never exported (the levels are specified in the <RD> code on export).

Note also that this code, if used, must precede any custom heading codes. See Records for more information on custom headings.

For information on applying levels, see Records.

# Examples

<RD>The level application code may <LV:"Chapter">appear anywhere in a record.

<RD>It may not be used if the record code sets the level.<LV:"Heading">

Level Definition

# Syntax

 <LN:Name1,Name2,Name3,...>

# Parameters

Name may be up to 127 characters long. Up to 255 names may be specified in hierarchical order (including Normal Level).

Note: There are three codes used to define and apply levels. The <LN> code defines the names of the levels and specifies their hierarchical order. The <LE> code defines optional formatting styles for the levels specified in the <LN> code. The <RD> code is used to actually apply the levels to the finished infobase.

Only one Level Definition code may exist per infobase.

# Description

The Level Definition code lists, in hierarchical order (highest to lowest), the levels which are used in the infobase. The "Normal Text" level does not need to be included in this code.

Each level in the infobase is named. The names may be 1 to 127 characters long (including spaces). If a name contains spaces or other non-alphanumeric characters, the name must be enclosed in quotes. Up to 255 levels may be specified by name (including Normal Level).

For convenience, each level should have a meaningful name which relates to the type of information which is contained on that level.

For information on applying levels, see Records.

Note: This was a required code for Folio Flat File 3.1. It is an optional code for Folio flat file 4.x. If this code is not used, then the order of the LE codes in the definition file will set the hierarchy.

# Examples

<LN:Volume,Book,Chapter>

Old Testament, Genesis, Chapter Three

<LN:Author,Play,Act,Scene>

Shakespeare, Hamlet, Act 3, Scene 2

<LN:Title,Heading,"Sub Heading">

Common Mistakes, Driving, Going too slow in the fast lane

Level Style Definition

# Syntax

 <LE:Level Name;Paragraph Based Formatting Codes,Character Based Formatting Codes>

# Parameters

Level Name must be one of the Names defined in the <LN> code if the <LN> code is used. If the LN code is not used to set the hierarchy of the levels in the infobase, the order of the LE codes is used to set the hierarchy. Name may be up to 127 characters long, including spaces.

Paragraph Based Formatting Codes are optional. They are used to apply paragraph-based attributes to the paragraphs in the level. These codes include all codes as listed under Paragraph Based Formatting. Individual codes must be separated by commas.

Character Based Formatting Codes are optional. They are used to apply character-based attributes to the paragraphs in the level. These codes include all codes as listed under Character Based Formatting. Individual codes must be separated by commas.

# Description

Level Styles are used to mark all paragraphs within a level which are not otherwise marked. Level Styles are useful for formatting large areas of similar text quickly (such as all the chapter headings or all the paragraphs within a chapter). Character, Paragraph, or Link styles which are found within in the level override the Level Style.

The order that the level styles are defined can set the hierarchy of the levels in the infobase. If the Level Definition code is not used, then the order of the LE codes is used. For example, if you define the style for the Chapter level before the Heading level, then the hierarchy in the infobase will be Chapter, Heading.

For information on applying levels, see Records.

# Examples

<LE:Volume; JU:CN, BD+, IT-, ST-, UN->

Defines the style for the Volume Level. This level is Center justified and is in Bold text. (The other character attributes — with the exception of 'Hidden' — are turned off. When this style is applied over any other text which may have those attributes, they are turned off until this style is ended.)

<LE:Book; JU:LF, FC:255,0,0>

Defines the style for the Book Level. The level is left justified, in red (255,0,00) text.

<LE:Chapter; LH:0.25, BC:0,0,255, UN+>

Defines the style for the Chapter Level. The level has a fixed line height of 0.25 inches, has a blue (0,0,255) background color, and is underlined.

Line Height

# Syntax

 <LH:Height>

# Parameters

Height is a Decimal number in inches (XX.xxx) specifying the height of the line.

# Description

Line Height is a Paragraphed-based code representing the distance between lines of text (when text is single spaced). By default, Folio Views automatically adjusts the line height to accommodate the current fonts, attributes, and positions of the text. However, you may change the line height for any specified paragraphs to any fixed number. Setting the line height increases or decreases the amount of white space added to the bottom of a line.

The height of the line is measured in inches. 0 represents automatic line height.

If <LH> is used by itself, with no parameters, it functions exactly as if 0 had been entered as a parameter and returns the Line Height to automatic.

Line Height may be used when you know that an infobase is to be printed and when you want to generate a specific number of lines per printed page.

# Examples

<LH:0.25>This paragraph has a line height of one-quarter inch.

<LH:0>This paragraph has automatic line height generated by Folio Views.

Line Spacing

# Syntax

 <LS:Spacing>

# Parameters

Spacing is a Decimal number (XX.xxx) representing a multiple of the Line Height.

# Description

Line Spacing is a Paragraph-based code which is used to specify the distance between lines. Default setting for Line Spacing is 1; however, Spacing may be set to any value greater than 1 (including fractional values in decimal form: 1.5, 2.25, etc).

Line Spacing is measured as a multiple of the Line Height. Line Spacing equal to 1 results in the paragraph being Single Spaced. 2 is equivalent to Double Spacing (twice the line height).

# Examples

<LS:1.5>This paragraph is 1½ spaced.

<LS:3>This paragraph is triple spaced, representing three times the line height between lines.

Line Width

# Syntax

 <LW:Width>

# Parameters

Width is a Decimal number in inches (XX.xxx) specifying the fixed width of a paragraph. A zero (0) value specifies automatic width.

# Description

Line Width is a Paragraph-based code which is used to specify the width of lines in a paragraph. Width may be set to any value between 0 and 22 inches (including fractional values in decimal form: 1.5, 2.25, etc). 0 indicates auto-width, which allows the paragraph to size itself to the window, no matter how large or small the window is. Any other value represents a fixed width.

If <LW> is used by itself, with no parameters, it functions exactly as if 0 had been entered as a parameter and returns the Line Width to automatic.

# Examples

<LW:5>This paragraph has a fixed width of 5 inches.

<LW:0>This paragraph has an auto-width, which allows it to wrap to the size of the current window.

Links

# Description

 Links allow you to move quickly from one area of the infobase to another or from an infobase to an external program. Links can also be used to perform queries, pull up an object within the infobase, or create a pop-up window containing textual information.

Folio Views software supports the following link types:

Command Link

Data Link

End Note Link

Jump Link

Named Popup Link

Object Link

Popup Link

Program Link

Query Link

Web Link

Note: When Flat files are exported from Folio Views or from Extract, the filter exports an <EL> code to end all links (rather than </JL>, </OL>, </QL>, </PL>, </PW>, </DL>, </PX>, or </EN>). Either the <EL> or the Standard End Link codes may be imported.

Link Styles

 See Text Styles.

Named Popup Link

# Syntax

 <PX:Style,"Title"> . . . </PX>

# Parameters

Style Name refers to a Link (LK) style defined with the Text Style code <ST>. May be up to 127 characters long.

Title is required. Refers to the Named Popup Definition <DP> which stores the text for the popup. Title may be up to 127 characters long, including spaces. Title MUST be included in quotes.

# Description

Named Popup Links are used to display non-searchable text in a popup window. The text in the named popup link is only stored once in the infobase, but may be referenced (by name) in many popups.

Named Popup Link codes mark the beginning of a Named Popup Link. All text or objects enclosed by the beginning and ending Named Popup Link codes serve as the link launch point.

Each link has a link style associated with it (defined using the Text Style code in the Definition File). The link style is used to mark the text selected to be the link launch point (objects are not affected by link styles), allowing you to consistently mark similar link types.

Note: When Flat files are exported from Folio Views or Extract, the filter exports an <EL> code to end all links (rather than </PX>). Either code may be imported.

# Examples

<RD>Move the <PX:"Named Popup","Insertion point definition">insertion point</PX> to the beginning of the record.

<RD>Join <PX:"Named Popup","NIRVANA">Nirvana</PX> today to spice up your life!

Named Popup Link Definition

# Syntax

 <DP:Width,Height,"Title"> . . . </DP>

# Parameters

Width is optional. If used, the Height must also be specified. It is a Decimal number in inches (XX.xxx) which sets the width of the popup window in the infobase.

Height is optional. If used, the Width must also be specified. It is a Decimal number in inches (XX.xxx) which sets the height of the popup window in the infobase.

Title is required. All named popups must have a name. Title may be up to 127 characters long, including spaces. Title MUST be included in quotes.

If <DP> is used without width or height parameters, default values are assigned. The default values are: Height = 2 inches; Width = 2 inches.

# Description

Named popups links display non-searchable text in a popup window in the infobase. The definition provides the text and formatting for the information in the link.

Named popup links refer to the definition by the title. All named popup links must have a title.

Between the <DP> and </DP> codes, any text, object definition, link, character formatting, paragraph formatting, or table codes may appear.

Note: Named popup definitions appear in the document file (not the definition file). They must appear before the first record code in the flat file.

# Examples

<DP:"NIRVANA">NIRVANA - National Institute for Reducing Vocal And Noisome Apples</DP>

<DP:1.5,2.3,"Insertion point definition"><PS:"Popup Definitions><CS:Emphasis>Insertion Point</CS> -- Point at which text may be entered into a document. Generally represented by a flashing vertical bar. Sometimes referred to as the <IT+>cursor<IT->.</DP>

Note

# Syntax

 <NT:Width,Height,"Title"> . . . </NT>

# Parameters

Width is optional. If used, the Height must also be specified. It is a Decimal number in inches (XX.xxx) which sets the width of the note window in the infobase.

Height is optional. If used, the Width must also be specified. It is a Decimal number in inches (XX.xxx) which sets the height of the note window in the infobase.

Title is optional. If used, it appears as the title for the note. Title may be up to 127 characters long, including spaces. Title MUST be included in quotes.

If <NT> is used without any parameters, default values are assigned. The default values are: Height = 2 inches; Width = 2 inches; Title = Note.

# Description

Notesare a personalization feature of Folio Views. They allow users to store notes in the infobase without corrupting the appearance of the text.

Notes are displayed in Folio Views as a yellow icon next to the paragraph in which they are placed. When the note is activated, the text contained between the Note codes appears in the note window.

Text contained in the Note may have any of the Paragraph and Character attributes applied to it (including styles). Objects may also be placed in notes, as may links.

Only one Note code may be used per paragraph (not per record). Additional Note codes are ignored.

For consistency, Note codes should be placed at the beginning of the paragraph. However, they may be placed anywhere in the paragraph.

# Examples

<HR><NT:3,2,"My Note"><PS:Funky>This is the text of a marginal note. It has the paragraph style "Funky" applied to it.</NT> This paragraph has a marginal Note created for it in the infobase. The text between the Note codes appears only in the note. Notice that this note is titled "My Note."

<RD><NT:4,4><JU:CN><OB:Car,Bitmap,1,1><HR><BP:0.12 5> Don't you just <IT+>love<IT> this car?</NT> The note in this paragraph contains an Object and text.

<RD><NT>Nothing fancy in this note. It uses all of the default values (2" x 2", title = Note).</NT>Just a simple note.

Objects

# Description

Objects are graphics, animation sequences, video clips, audio clips, and other non-textual information which are placed inside of the infobase. Objects are defined using the Object Definition code and are placed in the infobase by using either the Object Application code or the Object Link code.

Object Application

# Syntax

 <OB:Object Type:"Object Name",Width,Height,BR:Border,SD:Shade>

# Parameters

Object Type must be one of the supported general object types (either FO for Folio object or OL for OLE object).

Object Name refers to an Object as defined with the Object Definition <OD> code in the Definition File.

Width is optional. It is a Decimal number in inches (XX.xxx) which sets the width of the object in the infobase.

Height is optional. It is a Decimal number in inches (XX.xxx) which sets the height of the object in the infobase.

Border is optional. It refers to the Border code and places a border around the object in the infobase.

Shade is optional. It refers to the Shade code and provides the background fill color for the object.

# Description

The Object Application Code is used in the Document File to position graphics and other objects within the infobase.

To place an Object in the Document File, the object must be defined in the Definition File using the Object Definition <OD> code. Once it has been defined, an object may be referred to in multiple <OB> codes.

Object Type must be either FO (for Folio objects) or OL (for OLE objects). Folio objects are the only types of objects that can be displayed in the Picture dialog.

If both the width and height are set to 0, the object is auto-sized to match the original size of the graphic.

# Examples

<OB:FO:"Cool Cars",3,0;BR:AL:0.015,0.03,FC:255,0,0;SD:128,128,128>This Object code places a Folio object named "Cool Cars" in a 3 inch wide frame (the height is determined by the original size of the graphic). The object has a red border and a gray fill color.

<OB:OL:"Book1",4,3>This object code places an OLE object named "Book1" in a 4 inch by 3 inch frame. No border is applied to the object.

Object Definition

# Syntax

There are several slightly different object definitions, depending on the type of object being defined. The general object types are Folio, Data Link, OLE, and Class. The definitions for the these types are:

Folio Objects

 <OD:FO:"Object name",Object Handler,"File Name",RP,File Type>

Data Link Objects

 <OD:DL:"Object name","MIME Type","File Name",RP>

OLE Objects

 <OD:OL:"Object name",Version,"File Name",LI,IC,RP>

Class Objects

 <OD:CL:"Object name",Class Name,"File Name",RP>

# Parameters

All parameters for the various definitions are listed below.

Object Name is the name of the object as it will appear in the infobase. May be up to 127 characters long. The object name is referenced in the Object Application code, Object Link code, Title Page code, or a Data Link code. Note that the object name must be unique within the general object type (no two Folio objects can have the same name in the infobase). Note also that, for OLE objects, the name is generated by the OLE server when the object is exported.

File Name is the name of the file which holds the object. May use a full or relative path.

Object Handler is the name of the object handler to use to display the image (required for Folio objects). The default object handlers included with Folio Views are Bitmap, Metafile, HyperGraphic, and Picture. Third-party developers may create other compatible object types. (The object handlers are listed in the Windows Registry under HKEY\_LOCAL\_MACHINE\SOFTWARE\FolioViewsEnu\4\ObjectHandlers.)

MIME Type is required for Data Link objects. It refers to the MIME (Multipurpose Internet Mail Extensions) type for the type of file you wish to include in a data link. The MIME types registered on your computer are listed in the Data Link dialog and in the Windows Registry under HKEY\_CLASSES\_ROOT.

Version is required for OLE objects. It is always 2.0.

LI is an option for OLE objects. It indicates that the object is linked (not embedded).

IC is an option for OLE objects. It indicates that the object is displayed as an icon.

Class Name is required for Class objects. It refers to a custom object class type created by a third-party developer.

RP is optional. See Replace Definition for more information.

# Description

Object Definition codes are used to specify objects which are included in the infobase. Objects may be graphics, animation sequences, video, audio, or other non-textual information.

There are four general object classifications: Folio, Data Link, OLE, and Class.

Folio objects are those which are created using the Folio Object Handler API included with the Folio Integrator (software developers kit). Default Folio object types include Bitmap, Metafile, HyperGraphic, and Picture.

Data Link objects are used in Data Links only. They reference a file (such as a word processor document or spreadsheet file) that will be stored in the infobase.

OLE objects are OLE (object linking & embedding) object embedded or linked in the infobase. These can only be created on export; the files created by the export filter may be imported into an infobase through any flat file.

Class objects are custom object types for any data that can be stored in the infobase. These object types must be defined and handled using the Infobase API (NFOAPI) in the Folio Integrator. A developer might use this code to embed a custom form in the infobase that is referenced by a custom extension or to embed a query template created with a better query template editor. Basically, a class object can be any type of file; how it is used in the infobase is dependent on the developer (the data link object type and the query template code used in flat file are Folio-specific implementations of this type of object).

All Folio objects require an object handler. The object handler manages the display of the object within the infobase. Additional object handlers may be created using the Object Handler API included with the Folio Integrator.

Note that the bitmap and metafile object handlers attempt to autodetect file types for conversion to bitmap or metafile. If you have raster images that are not bitmap, use the Bitmap object handler to convert it in the infobase. Use the Metafile object handler for vector images. (Not all formats are supported.)

Note: Folio Views 4.x does not support objects linked to a file (other than OLE objects). Object definitions which use the link to file option in 3.x flat files are converted to embedded objects. To ensure that the object is not lost, copy the files being referenced by the flat file to the same directory as the flat file.

# Examples

<OD:FO:"Cool Car",HyperGraphic,"c:\files\tiff\car.tif">

Defines a Folio object named "Cool Car" as a hypergraphic image with a filename of c:\files\tiff\car.tif. The file is stored in the infobase in its native format.

<OD:DL:"Accounting Form","Word.Template",f:\forms\account.doc">

Defines a Data Link object named "Accounting Form." The object MIME type is Word.Template (a Microsoft Word template file). The file name is f:\forms\account.doc. The file will be stored in the infobase and displayed in its associated application when a data link referencing this object is followed.

<OD:OL:"OLE1fa0",2.0,"FFF1.OLE",LI,IC>

Defines an OLE object named OLE1fa0. It is stored in the file FFF1.OLE. The OLE object is linked to a file and is displayed as an icon. Remember that these files may only be created by exporting an infobase to flat file.

<OD:CL:"Custom Form","CustomClass","input.frm">

Defines a Class object named Custom Form. The object is of class type CustomClass, and the file is stored in the file input.frm.

Object Link

# Syntax

 <OL:Style name,Object Name,Class Name,Infobase Name,ZM>  
 .   
 .   
 .   
 </OL>

# Parameters

Style Name refers to a Link Style (LK) as defined with the Text Style <ST> code in the Definition File.

Object Name refers to an Object as defined with the Object Definition <OD> code in the Definition File.

Class Name is optional. It must be specified when creating an inter-infobase link. If linking to a Folio 4.x infobase, class name must be Folio Objects. If linking to a Folio 3.x infobase, it must be Bitmap or Metafile.

Infobase Name is optional. It specifies a valid UIL for the infobase. If not specified, the current infobase is assumed. Note: This option is not valid for infobases used on the Macintosh.

ZM is optional. If used, it sizes (zooms) the object to fit the Object Window.

# Description

Object Links display the object in a separate pane in Folio Views. This type of link is useful in keeping graphics out of the way of infobase text until needed.

Note that only Folio objects may be used in an object link. OLE objects are no longer supported in object links.

Object Links can refer to objects in another infobase. If you wish to link to an object in a separate infobase, include the Infobase Name in the Query Link code. Infobase name must be a valid UIL (universal infobase locator), which is generally just the infobase file name. See the appendix Understanding the Folio Search Path for more information on the UIL. Please note that this functionality is only available on the Windows platform. If your infobase will be used on the Macintosh platform, do not use this option.

ZM (zoom to fit window) may be used to proportionally size the object to the window size specified. This option ensures that the object fits within the window.

Please note that Object Link codes are paired codes. Text or other information to be marked as an Object Link must be entirely enclosed by the Begin Object Link Code <OL> and by the End Object Link Code </OL>.

Note: When flat files are exported from Folio Views or Extract, the filter exports an <EL> code to end all links (rather than </OL>). Either code may be imported.

# Examples

<OL:Objects,Car,ZM>This paragraph is marked with the link style named "Objects" and is the launching point for an Object Link. The object is sized to fit the window.</OL>

An object is used as the launch point of an Object Link in this example. <OL:Empty,Schematic1><OB:FO:MiniIcon,1,1></OL> The link uses the link style "Empty" and contains the graphic image "Schematic1." See Object Application for more information on the object used as the launch point.

Page Break

# Syntax

 <PB>

# Description

Page Break is a special character code which may be used to format documents for printing. If you wish to keep Chapter 1 separate from Chapter 2 when the infobase is printed, you should include a Page Break code between the two chapters.

The Page Break code is analogous to a Hard Page code in most word processors. Page breaks do not start new records, but only one may be used per record. The page break (in Folio Views) is shown at the end of the record and will only break the page at the end of the record when the infobase is printed.

# Examples

<RD>This paragraph is split in the middle by a page break code. This code should only be used to guarantee <PB>that two items do not occur on the same printed page (or when you wish to force two items on to the same page). In these instances, page breaks should be used.

<RD>Chapter One: The Story of Your Life, in MarvyColor<PB>  
Chapter Two: I was a Teenage BoneHead

Paragraph-Based Formatting

# Description

 Paragraph-based formatting affects the infobase at the paragraph level. The appearance and structure of entire paragraphs may be changed by Paragraph-Based Formatting codes. Compare with Character-Based Formatting codes, which affect the appearance of text within paragraphs.

Paragraph-Based Formatting codes may only be used in the Document file or in Level or Paragraph Style Definitions, and include the following:

After Spacing <AP:Value>

Before Spacing <BP:Value>

Borders <BR:Left line width,Top line width,Right line  
 width,Bottom line width,Line color,Left inside  
 space,Top inside space,Right inside space,  
 Bottom inside space,Fill color>

Indents <IN:Left,Right,First>

Justification <JU:Justification>

Keep Together <KT>

Keep with Next <KN>

Line Height <LH:Height>

Line Spacing <LS:Spacing>

Line Width <LW:Width>

Tab Set <TS:Location,Justification,Leader>

Paragraph Break

See Hard Return.

Paragraph Proximity

# Syntax

 <PP>

# Parameters

None.

# Description

Paragraph proximity codes mark regions of text in an infobase which may be searched using the Paragraph Proximity query syntax. With paragraph proximity searches, all of the terms in the query must be found between two paragraph proximity codes to count as a hit.

Proximity searches use a window metaphor. The search checks for the terms you are searching for within a specified window. If they all appear within the window, then the query reports a hit. The default window for paragraph proximity searches is the record. There are implied paragraph proximity codes at the beginning and ending of all records in the infobase. As such, you only need to insert paragraph proximity codes where you wish to create additional proximity windows. For example, if you insert a paragraph proximity code into the middle of a record, you have divided the record into two proximity windows (from the start of the record to the PP code and from the PP code to the end of the record). See the examples for more information.

Please note that paragraph proximity codes may appear anywhere in a paragraph (they do not necessarily need to go at the start or the end of a paragraph).

See Sentence Proximity for related information.

Note: The primary purpose for the paragraph proximity codes is to allow you to create larger records in your infobase while still providing a mechanism to search smaller units of the infobase. Using fewer but larger records in your infobases can reduce the overall size of the infobase.

Note: This code is designed for use with Folio Views 4.2 and later. Versions of Folio Views 4 prior to version 4.2 can open infobases which use this code. However, the text in the infobase may display incorrectly and search results may be erroneous. As an infobase creator, it is your responsibility to ensure that your users have upgraded their software in order to use this feature.

# Examples

<CM>Example 1  
If there are no PP codes in a record, then the entire record is treated as a single paragraph proximity window.</CM>  
<RD>All records have implied paragraph proximity codes at the start and end of the record.

<CM>Example 2  
The following record is divided into two proximity windows.</CM>  
<RD>It was the beast of Thames,<PP>It was the wurst of Thames.

<CM>Example 3  
One suggested use is of paragraph proximity is for lists of items within a record. The following example is divided into three proximity windows.</CM>  
<RD>When building infobases, keep in mind the following:  
<HR><PP>1. Balance compression with build speed.  
<HR>2. Use large records for better overall compression.  
<HR>3. Pull the source data from one drive and build to another drive to improve build speeds.  
<HR>4. Order pizza and root beer.  
<HR><PP>These tips should help the entire build process run smoothly, unless you order extra pepperoni.

Paragraph Spacing

# Description

 Paragraph Spacing consists of two paragraph-based formatting codes which allow you to add spacing either before or after a paragraph without inserting additional hard return codes.

Two codes are used for Paragraph Spacing:

After Paragraph  
 Before Paragraph

Paragraph Style Application

# Syntax

 <PS:Name>

# Parameters

Name refers to a style name as specified in a Paragraph Style Definition <PA> code in the Definition File.

# Description

Paragraph Style is a paragraph-based formatting code which may be used to apply previously defined paragraph-based and character-based formatting to text.

Paragraph Styles are defined by name in the Definition File using the Paragraph Styles Definition code. The definition contains the paragraph and character-based formatting codes which are used each time the style is applied. Names may be 1 to 127 characters long.

Once a Paragraph Style has been defined, it may be applied in the Document File at the beginning of paragraphs. When the infobase is created, the style definition may be modified, if necessary.

# Examples

<PS:Simple>This paragraph has been marked with the Paragraph Style named "Simple." The style must be defined in the Definition File.

<PS:Complex>This paragraph is marked with the Paragraph Style "Complex" and <CS:"Funky">two character styles: "Funky" and "Nifty." Note </CS>that only one paragraph style may be applied to<CS:"Nifty"> any one paragraph, but that multiple character style may be applied.</CS>

Paragraph Style Definition

# Syntax

 <PA:Name,Paragraph based formatting codes,Character based formatting codes>

# Parameters

Name may be up to 127 characters long. The name is referred to by the Paragraph Style Application <PS> code in the document file.

Paragraph based formatting codes are optional. They are used to apply paragraph-based attributes to the paragraphs to which the style is applied. These codes include all codes as listed under Paragraph Based Formatting. Individual codes must be separated by commas.

Character based formatting codes are optional. They are used to apply character-based attributes to the paragraphs to which the style is applied. These codes include all codes as listed under Character Based Formatting. Individual codes must be separated by commas.

# Description

Paragraph Styles affect only information at the paragraph level. All text within the paragraph is affected by the paragraph style (to affect smaller areas, use Character Styles). Paragraph Styles may contain both paragraph-based formatting codes and character-based formatting codes.

When paired codes (Subscript, Superscript, etc) are used in a style, only the opening code needs to be included in the style. For example:

<PS:Normal,SB:0.039>

When non-paired Character Attribute codes (Bold, Italic, Underline, Strikeout, Hidden, Font, Point Size, and Color) are used, you may need to specify whether that attribute is to be ON or OFF (if not specified, it is assumed to be default). This is especially important for Bold, Italic, Underline, Strikeout, and Hidden. When one of these attributes is ON, it affects all text to which it is applied. When one of these attributes is OFF, it turns off that attribute entirely (even if that attribute was applied in a Level Style).

Paragraph Styles are named for application of the style in the Document File. Names may be 1 to 127 characters long, including spaces.

See the appendix RGB Color Combinations for a list of the RGB values for the standard colors available through the Folio Views dialogs.

# Examples

<PA:Normal; JU:LF; IN:0,0,0.25; FT:Arial; PT:12>

Defines a paragraph style called "Normal" which left justifies all marked paragraphs, as well as creating a first line indent of 0.25 inches and using Arial 12 point as the font.

<PA:Notes;FT:Arial;PT:12;BD+,IT-, UN-,ST->

Defines a paragraph style named Notes which uses the default Font and Point size, turns Bold ON, and turns Italics, Underline, and Strikeout OFF.

<PA:Funky; IN:1,1,0; BC: 255,0,0; FC:0,255,0; FT:Script; BD+, UN->

Defines a paragraph style named "Funky" which has left and right indents of 1 inch. Text is green on a red background and in the Script font. Bold is ON, Underline is OFF, and all other character attributes are assumed to be default.

Point Size

# Syntax

 <PT:Size>

# Parameters

Size is a decimal number in POINTS (XX.xx) specifying the size of the font.

# Description

Point Size works hand-in-hand with Font Facename. It is a character based formatting code which specifies the point size of the displayed font. Point Size affects all characters following the code (no matter what the font) up to the next Point Size code or the end of the paragraph, whichever comes first.

Point Size is measured in Points, not inches, for consistency with other applications. Points may be specified out to 2 decimal places (i.e. 9.85 or 22.25)

<PT> used without a parameter is a Default Code. This code ends the current point size and reverts back to the previous point size. This option is provided to save space in the flat file and to permit styles to function correctly.

# Examples

<PT:12>This entire paragraph is in a 12 point font.

<PT:10>Part of this paragraph is in a 10 point font; <PT:14.5>part is in a 14.5 point font, <PT>and the last part reverts back to the default state (in this case, 10 point).

Popup Link

# Syntax

 <PW:Style Name,Width,Height,"Title">  
 .  
 . (Popup Window Text)   
 .  
 <LT>  
 .  
 . (Link Text)  
 .  
 </PW>

# Parameters

Style Name refers to a Link Style (LK) as specified in the Text Style <ST> code in the Definition File.

Width is optional. If used, the Height must also be specified. It is a Decimal number in inches (XX.xxx) which sets the width of the popup window in the infobase.

Height is optional. If used, the Width must also be specified. It is a Decimal number in inches (XX.xxx) which sets the height of the popup window in the infobase.

Title is optional. If used, it appears as the title for the popup. Title may be up to 127 characters long, including spaces. Title MUST be included in quotes.

If <PW> is used without the size and title parameters, default values are assigned. The default values are: Height = 2 inches; Width = 2 inches; Title = Popup.

(Popup Window Text) is the text and formatting codes that appear in the Popup Window.

(Link Text) is the text which is the link launch point.

# Description

Popup Links create a link between text or an object and a Popup Window containing notes or other information. The window appears on the screen in the dimensions specified, either directly above or below the link launch point (depending on the amount of space available).

Popup Links may contain text or objects and any paragraph or character based formatting codes (including styles). Traditionally, popup windows are used to define words or ideas used within the infobase; however, they may be used for any purpose.

Popup Links consist of three parts: 1) Popup Link code, 2) Window text (text which appears in the popup window), and 3) Link Text (text which marks the starting point of the link).

1. Popup Link Code <PW:Style name,Width,Height,"Title">

Style Name refers to a link style which has been defined using Text Styles in the Definition File. It is applied only to the Link Text (link launch point).

Width and Height refer to the fixed width and height of the popup window (measured in inches).

Title refer to the text which appears as the title of the popup window.

2. Window Text

The window text is the text and formatting that you want to appear in the pop-up window. It may contain paragraph-based formatting codes, character-based formatting commands, paragraph or character styles, and objects. All text in the note window is, by default, placed in one record, so level codes and record codes are not allowed. The Link Text code <LT> must follow the end of the window text.

There is no limit to the amount of data stored as window text (earlier versions were limited to 4K).

3. Link Text

The Link Text is the text which appears in the main body of the infobase and which serves as the link launch point. When the Link Text is selected in the infobase, the popup window comes up. Link Text is marked with the Style name specified in the Note Link code (#1 above). Link Text may be, instead of text, an object, or a combination of the two.

Note: When Flat files are exported from Folio Views or Extract, the filter exports an <EL> code to end all links (rather than </PW>). Either code may be imported.

# Examples

<PW:Popup,3,3,"My Popup"><PS:"Popup Text">This is the text which appears in the pop-up window. The window is 3x3 inches. Note that formatting codes may be used inside of <IT+>Popup Links<IT><LT> This is the text which opens the popup window. It is marked with the "Popup" link style. Notice that this popup is titled "My Note." </PW>

Single words can be used as popup links. For example, the following codes <PW:Normal,2,4,><BD+>Hamlet:<BD> 1. A play written by Shakespeare; 2. A character by the same name; 3. A small, outlying village.<LT> Hamlet</PW> calls a pop-up window with two definitions for Hamlet. In the window, "Hamlet" appears in bold text.

<PW>Nothing fancy in this popup. It uses all of the default values (2" x 2", title = Popup).<LT>Just a simple popup.</PW>

Program Link

# Syntax

 <PL:Style Name,"Command line or URL"> . . . </PL>

# Parameters

Style Name refers to a Link Style (LK) as specified in the Text Style <ST> code in the Definition File.

Command Line is the command line used to start the program being linked to. It may include the full path and any startup options for the program. It may also simply include the file name of a document to open (Folio Views will attempt to open the document using the application associated with the document type; the association is set in the Windows registry and has nothing to do with flat file). May be up to 512 characters long.

Note: Program Links may be used as Web Links. Specify a valid URL in the command line. When the link is followed, the user's default Web browser is launched and the URL followed.

To open an application:

Use the full path to the application, use a relative path to the application, include the application in the same directory as the infobase, or include the application's directory in the PATH statement in the AUTOEXEC.BAT file. Examples include:

<PL:Program,"c:\apps\games\sol.exe">

<PL:Program,"..\games\sol.exe">

<PL:Program,"sol.exe">

To open a URL:

Specify a valid URL for the command line. Note that a Web browser must be installed for the link to work. Examples include:

<PL:Program,"http://www.microsoft.com">

<PL:Program,"http://www.netscape.com">

To open a file:

Specify a file — such as a word processing document, graphic, or spreadsheet — for the command line. You may specify the full path to the file, a relative path to the file, or just the file alone. The application associated with the file type is used to open the file. Examples include:

<PL:Program,"c:\docs\mydoc.doc">

<PL:Program,"..\bmp\car.bmp">

<PL:Program,"budget.xls">

Using Path Variables in Program Links:

Program links support path variables (%% and %?) for linking to programs and documents.

The %% variable is replaced with the absolute path to the infobase from which the link is executed.

The %? variable is replaced with the absolute path to Folio Views.

Both variables include a directory delimiter (\), meaning you do not need to type in the backslash, it is automatically added. For example, if the infobase was launched from the c:\nfos directory,

%%MyApp.exe

might become

c:\nfos\MyApp.exe

# Description

Program Links are used to create links between text or graphics and an external program (such as a spreadsheet or graphic editor), URL, or file (such as a document). When the external program is exited, the cursor returns to the link launch point in the infobase.

Please note that Program Link codes are paired codes. Text or other information to be marked as a Program Link launch point must be entirely enclosed by the Begin Program Link Code <PL> and by the End Program Link Code </PL>

If you are distributing your infobase to other users, make sure that the full path to the applications are not included in the link. The paths will cause problems when used on two different systems or platforms. Use relative paths (relative to the directory containing the infobase) or rely on the Windows Registry to locate the application.

Note: When Flat files are exported from Folio Views, the filter exports an <EL> code to end all links (rather than </PL>). Either code may be imported.

# Examples

Windows Examples

<PL:Program,"sol.exe">This link assumes that the application is in the same directory as the infobase or is in the search path.</PL>

<PL:Program,"c:\windows\sol.exe">This link specifies the full path to the application.</PL>

<PL:Program,"notepad ..\files\text\readme.txt">This link assumes that the application in same directory as the infobase or is in the search path. It also uses a relative path to specify a file to open in the application.</PL>

<PL:Program,"http://folio.nextpage.com">This link specifies a URL to follow using the default web browser for the system.</PL>

Query Link

# Syntax

 <QL:Style Name,"Query",Records with Hits,"Infobase name">  
 .  
 .  
 .  
 </QL>

# Parameters

Style Name refers to a Link Style (LK) as specified in the Text Style <ST> code in the Definition File.

Query is the query you wish to perform (just as if you were entering it into the Advanced Query Dialog). Query may be up to 2048 characters long.

Records with Hits is optional. If included, the view of the infobase is narrowed to contain only those records which contain hits. The flag for this option is RH.

Infobase Name is optional. It specifies a valid UIL for the infobase. If not specified, the current infobase is assumed.

# Description

Query Links are used to perform commonly used queries from text or objects within the infobase. Query Links can be very useful in an index: when a word is selected in the index, all related occurrences of the word are highlighted in the infobase.

Query is the actual query line that would be entered in the advanced query dialog. Query may be up to 2048 characters long. For more information on the Query Syntax, see Advanced Query in the Folio Views Reference.

Records with Hits is a flag used to instruct Folio Views on how to display the query results. If included, Folio Views narrows the view of the infobase to just those records which contain hits (useful when linking to a group); if excluded, Folio Views maintains a full view of the infobase.

Query Links can exist between information in two infobases. If you wish to link to information in a separate infobase, include the Infobase Name in the Query Link code. Infobase name must be a valid UIL (universal infobase locator), which is generally just the infobase file name. See the appendix Understanding the Folio Search Path for more information on the UIL.

Please note that Query Link codes are paired codes. Text or other information to be marked as a Query Link launch point must be entirely enclosed by the Begin Query Link Code <QL> and by the End Query Link Code </QL>

Note: When Flat files are exported from Folio Views or Extract, the filter exports an <EL> code to end all links (rather than </QL>). Either code may be imported.

Note: Query links always use the default partition set for the infobase, unless a partition is explicitly set in the query link. If the default partition is changed, the results of your query may change as well.

# Examples

<QL:Query,"Russ and Kim">This query searches for all records containing both "Russ" and "Kim." The search is performed in the current infobase.</QL> The previous two sentences function as the link launch point and are marked with the link style "Query."

<QL:Slick,"Mice or Men","c:\nfo\Classics.nfo">Mice or Men</QL> This query link searches for all occurrences of either "Mice" or "Men" within the classics.nfo infobase.

This link performs a query in a <QL:Test,"group writers:Jim Mosher,RH"\work\phone.nfo"> different infobase</QL> using a partial path to the infobase.

Query Template

# Syntax

 <QT:"Template Name","File Name">

# Parameters

Template Name is the name of the template as it will appear on the Search menu in the finished infobase. Preceding a character in the name with an ampersand (&) makes that character an accelerator key for the template (DOS and Windows versions of Views only.) Template Names may be 126 characters long.

File Name is the name of the template file to be included in the infobase. If the template is not stored in the same directory as the flat file, the full path for the template must be specified.

# Description

Use this code to add query templates to your infobases. A query template is a custom-designed query dialog which takes advantage of the structure within a specific infobase and hides the complexity of the Folio query syntax from the user.

Query templates are created with the Query Template Editor (QTE).

# Examples

<QT:"&1 My Template","My Template.tmp">

<QT:"&2 Your Template","c:\views\template\tours.tem">

Reconcile Shadow File

# Syntax

 <RC:Merge Type,Merge Option>

# Parameters

Merge Type specifies the reconcile type to use. For version 4.x, the only option is append merge:

AP Append Merge

Merge Option specifies the options to use during the merge. Merge Option is an integer value which is the sum of the options you wish to set for the merge type.

The options for Append Merge are:

0 Keep nothing (not recommended)

1 Keep Original Master

2 Keep New Master

4 Keep Original Shadow

Valid values for the Append Merge option are 0–7, inclusive.

# Description

Use this code to force the shadow file reconcile options for any shadow file associated with the infobase. If the shadow file ever needs to be reconciled, these options are used. The user is never prompted and no dialogs appear. The shadow file is reconciled using the specified options.

Shadow files may be reconciled using an append merge method, in which records with changes are appended together into a single record.

The options are bit flags. The sum of the options sets the appropriate bits. For example, to keep the original master and the new master, use 3 (1 + 2 = 3).

The recommended setting is 6 for append merge.

# Examples

<RC:AP:6> Keeps the new master and original shadow in an append merge.

<RC:AP:7> Keeps everything in an append merge.

Record

# Syntax

 <RD,ID:Record ID:"Level Name">

# Parameters

Record ID is optional. If used, it indicates a persistent ID for the record. This persistent ID allows you to re-create infobases without invalidating shadow files associated with the infobase. Record IDs can be generated when an infobase is exported or specified in the flat file for import (should you choose to maintain the IDs yourself). Record IDs are hexadecimal values ranging from 1 to FFFFFFFF (do not use spaces in the hex number).

Level Name is optional. If used, the record to which the code is applied starts the level specified. The record also appears in the Table of Contents in the finished infobase. Level Name must be defined in the flat file using either the Level Definition code <LN> or the Level Style code <LE>.

Additional parameters are supported for this code. See Additional Information for details.

# Description

Infobases are built around the record structure. Queries return records with hits. Groups are collections of records. And Levels are hierarchical organizations of records. Without records, an infobase would be useless.

With that in mind, the Record code indicates the start of a new record, which may contain one or more paragraphs. Generally, a record should contain the smallest amount of usable information possible — which is usually a paragraph.

The Record <RD> code must be placed anywhere you want to start a new record. When a new record is specified, a new paragraph is created as well. The Record code, if used, MUST be used before any other text or formatting codes in the record.

Generally, you do not need to worry about the record ID until you are ready to re-create an infobase. When you re-create, set the option in the flat file export filter configuration dialog to export the persistent record IDs (select Write Record IDs). These IDs are then included automatically in your flat file. (The Folio Flat File with IDs filter uses this option.)

When you add new records to the flat file, you must either number them yourself or specify the previous master infobase in your Workbench project file. (The RN code used in 3.x flat file is not supported in 4.x). If you do not do one of these things, then it is possible to mangle the record numbers (records without IDs are assigned the next available ID, which may be used in a record that the filter hasn't encountered yet). Record numbers must be unique. (Note that you must specify the previous master in the Workbench to ensure the correct Infobase ID is transferred to the new infobase.)

You may also use record IDs to manage your information. You can create and maintain your own IDs in the flat file (rather than have the flat file filter export the IDs for you). Doing this may provide you with a method of tracking versions. For example, version one could use IDs from 0 to 100,000. Version two could use IDs from 200,000 to 300,000.

Note: Record IDs are long hexadecimal integers. They may range in value from 1 to FFFFFFFF. The 3.x flat file format for record IDs is also supported for backwards compatibility.

If the Level Name contains spaces or other non-alphanumeric text, it must be enclosed in full quotes.

Note: Persistent Record IDs are not preserved when you import a flat file in Folio Views. IDs are only preserved through the Create utility or the Folio Workbench.

# Additional Information

Custom Headings

You may set a custom heading for records that begin levels. The custom heading appears in the table of contents rather than the full text of the level heading. The syntax for a custom heading is:

<RD,ID:Record ID:"Level Name",CH>text<BH>text<EH>text

CH, BH, and EH are optional codes which allow you to specify part of a record as the heading for a level (the heading is the part that appears in the Table of Contents and Reference panes). By default, the entire record (or the first paragraph in a record) is used as the heading; you may set the heading to be a sub-set of the record.

The CH (custom heading) code lets the flat file filter know that a custom heading is being used in the record.

The BH (begin heading) code flags the start of the custom heading.

The EH (end heading) code flags the end of the custom heading.

You might use these codes to prevent hidden text from appearing in the Table of Contents, to mask out section numbers, or to reduce the amount of information displayed in the Table of Contents.

See the final example, below, for an example of how these codes might be used.

# Examples

<RD:Chapter>Sample Record Codes

<RD:Paragraph><IN:0.25,0,0><FC:255,0,128>The previous record belongs to the chapter level. This record belongs to the paragraph level. It is also a sub-set of the chapter level, allowing you to search (in the finished infobase) inside of the Chapter Level to get these words. The next paragraph is also a part of the paragraph level, even though it is not explicitly stated.

<RD,ID:ABCD1234><PS:Notes><BD+>Remember<BD>, records should be kept as small as reasonably possible. While there are times that you need to have more than one paragraph in a record, try not to make a habit of it. Doing so should not cause problems for the Folio Views software, but may cause problems for users of your infobase.

<RD:Chapter,CH>Captain Kirkum Recites John Jone's <BH>Brief History of the World<EH>

<RD:ID,45BD:Chapter,CH>Captain Kirkum Recites John Jone's <BH>Brief History of the World<EH>

Remark

# Syntax

 <RM>Remark Text</RM>

# Parameters

Remark text may be any length. The text may contain carriage return codes <CR>, but no others.

# Description

Remark is one of the Infobase Information Fields and is designed to hold the miscellaneous notes or comments about the infobase. Please refer to Infobase Information for details.

Remark actually appears in the Comment field in the Infobase Information dialog in the finished infobase.

Remark is a paired code. All text to be included in the abstract must be contained between the begin code <RM> and the end code </RM>.

# Examples

<RM>This infobase was created by a special presidential commission for the study of the effects of dried banana chips on ants.</RM>

<RM>Be sure to order a large pizza with everything and a keg of root beer before using this infobase.</RM>

Replace Definition

# Syntax

 RP, RP-

# Parameters

RP Indicates that the definition code should replace any existing definitions of the same name and type.

RP- Indicates that the definition code should be ignored if a definition of the same name and type already exists.

# Description

Use Replace Definition to mark which definitions may be ignored or used when the file is created into an infobase with other files.

RP (or RP+) explicitly replaces any definitions of the same name and type (if encountered).

RP- is the default setting. You may use it to explicitly mark those that should be ignored, if desired, but the code does not need to be added to your definition codes.

Replace Definition may be added to the end of any definition code which defines an item in the infobase, including styles, levels, highlighters, fields, infobase information codes, and objects.

# Examples

<QT:"&1 My Template","My Template.tmp",RP>

<ST:Fancy,CS,FT:Arial,PT:12, BC:100,0,0,FC:0,0,255,BD+,IT+,UN-,RP+>

<TP:"Introduction",RP->

<SU:RP>This is the subject of my infobase.</SU>

Revision Date

# Syntax

 <RE:"Date text">

# Parameters

Date text may be up to 45 characters long. It may not contain any other flat file codes.

# Description

Revision Date is one of the Infobase Information fields and is designed to hold the most current revision date of the infobase. This field is a Date field and should only contain a Date (up to 45 characters long). This field automatically updates each time the infobase is changed. Please refer to Infobase Information for details.

Note: This code is write-only (it is ignored on import).

# Examples

<RE:"January 12, 1992">

<RE:"5-30-83">

Sentence Proximity

# Syntax

 <SE>

# Parameters

None.

# Description

Sentence proximity codes mark regions of text in an infobase which may be searched using the Sentence Proximity query syntax. With sentence proximity searches, all of the terms in the query must be found between two sentence proximity codes to count as a hit.

Proximity searches use a window metaphor. The search checks for the terms you are searching for within a specified window. If they all appear within the window, then the query reports a hit. The default window for sentence proximity searches is the record. There are implied sentence proximity codes at the beginning and ending of all records in the infobase. As such, you only need to insert sentence proximity codes where you wish to create additional proximity windows. For example, if you insert a sentence proximity code into the middle of a record, you have divided the record into two proximity windows (from the start of the record to the SE code and from the SE code to the end of the record). See the examples for more information.

Please note that sentence proximity codes may appear anywhere in a record (they do not necessarily need to go at the start or the end of a sentence).

Note: This code is designed for use with Folio Views 4.2 and later. Versions of Folio Views 4 prior to version 4.2 can open infobases which use this code. However, the text in the infobase may display incorrectly and search results may be erroneous. As an infobase creator, it is your responsibility to ensure that your users have upgraded their software in order to use this feature.

See Paragraph Proximity for related information.

# Examples

<CM>Example 1  
If there are no SE codes in a record, then the entire record is treated as a single sentence proximity window.</CM>  
<RD>All records have implied sentence proximity codes at the start and end of the record.

<CM>Example 2  
The following record is divided into two proximity windows.</CM>  
<RD>It was the beast of Thames,<SE>It was the wurst of Thames.

<CM>Example 3  
One suggested use is of sentence proximity is for lists of items within a record. The following example is divided into six proximity windows.</CM>  
<RD>When building infobases, keep in mind the following:  
<HR><SE>1. Balance compression with build speed.  
<HR><SE>2. Use large records for better overall compression.  
<HR><SE>3. Pull the source data from one drive and build to another drive to improve build speeds.  
<HR><SE>4. Order pizza and root beer.  
<HR><SE>These tips should help the entire build process run smoothly, unless you order extra pepperoni.

Shade

 <SD:Color>

# Parameters

Color is the RGB color combination for the shade fill. Color may also be set to NO for none.

# Description

Shade sets the fill color (or shading) for paragraphs, objects, and table cells. You may specify an RGB color combination for a specific shade color, or you may set the shade to NO (for none). Setting the shade to NO returns the shading to whatever the default background color is for your system.

See the appendix RGB Color Combinations for a list of the RGB values for the standard colors available through the Folio Views dialogs.

# Examples

<SD:192,192,192>The paragraph is shaded with a light gray color.

<SD:NO>This paragraph has no background shading.

<BR:LF:0.1,0.1,TP:0,0.1,RT:0,0.1,BT:0,0.1><SD:192,192,192>This example places a single black border on the right side of the paragraph or table cell and creates a gray shade for the rest of the paragraph or table cell. The Top, Right, and Bottom sides of the border have no width, but do specify inside spacing. This additional spacing is filled by the Shade code and provides a cleaner appearance for the shading. Note that no color is specified for the lines; the system default text color is used (usually black).

Spacing

 See After Paragraph or Before Paragraph for more information on paragraph spacing. See Line Spacing for more information on line spacing.

Special Characters

# Description

 Special Characters allow you to display, in the finished infobase, the Extended ANSI character set (128-255) and to set Tabs, Hard Returns, and Page Breaks in the infobase.

The available special character codes are:

Carriage Return <CR>

Hard Return <HR>

Page Break <PB>

Tab <TB>

To access the ANSI set, press ALT+ any number between 128 and 255 on the keypad. The character is translated which appears on the screen in your editor to the appropriate ANSI character.

Note: Most DOS-based editors use the ASCII character set, which forces a translation of the ANSI text generated by the Flat File export filter. However, even though ASCII characters may appear in your flat file, the correct ANSI character appears in the finished infobase. For example, if you export a file containing an em dash (ANSI 151), you will see an ASCII 151 in your flat file in a DOS-based editor. This would still import as an em dash.

See the appendix Character Sets for the ANSI and ASCII character sets.

Strikeout

 See Attributes.

Styles

# Description

Styles allow you to apply frequently used formatting codes to characters, paragraphs, links, fields, levels, and highlighters. Styles must be named and defined in the Definition File; once named, they may be applied by name in the Document File.

Styles which may be defined in the Definition File are:

Character Styles

Highlighter Styles

Field Styles

Level Styles

Link Styles

Paragraph Styles

Set Definition of Content Building

# Syntax

<STPDEF:LE,TargetLevelName,SourceType,StyleSource,STP,format>

# Parameters

Level name is name of level as defined in tag LE.

SourceType is PA or LE, and is type of style defined in parameter StyleSource

StyleSource is name of style or level. While traversing FFF file by builder, if we are processing record with this style or level, then we will add content or currently processed record to last content item of level defined in TargetLevelName.

Format is used to set font properties like size, bold and italic propertioes. Possible values are:

* BM (bold, medium size)
* M (medium/normal size)
* S (small size)
* G (normal size, green color)
* T (big size, brown color)
* R (normal size, brown-green color)

# Description

* For BG level is "Verse Number"
* Title will be created by adding plain text from subparagraph PS:Textnum
* Also added translations from PS:Translation
* Title will look like <STP:B,I,12pt>TEXT 9<STP:10pt>This is translation text
* So tags for content building will be
* <STPDEF:LE,"Verse Number",PA,"Textnum",STP,BI12>
* <STPDEF:LE,"Verse Number",PA,"Translation",STP,10>
* These two should be placed after that level record, which should be parent for all converted content items
* So scope of these tags is only for subitems of given level record
* <STPDEF:LV,"Verse Number">
* This is removing all content building settings for level "Verse Number"
* <STPDEF>
* This is removing all content building settings

Set Last Content Level

# Syntax

<STPLAST:LevelName>

# Parameters

Level name is name of level as defined in tag LE or LV.

# Description

This tag will set last level which is visible in the content for given subtree. It is valid only forcurrent content level and all sublevels.

# Examples

<STPLAST:”Verse Number”>

<STPLAST>

Subject

# Syntax

 <SU>Subject text</SU>

# Parameters

Subject text may be any length. The text may contain carriage return codes <CR>, but no others.

# Description

Subject is one of the Infobase Information Fields and is designed to hold a summary of the main contents of the infobase. Please refer to Infobase Information for details.

Subject is a paired code. All text to be included in the abstract must be contained between the begin code <SU> and the end code </SU>.

# Examples

<SU>The life and times of Rabid Rabbit<CR>Expanded Version</SU>

<SU>Cajun cooking made simple</SU>

Subscript

# Syntax

 <SB:Offset> . . . </SS>

# Parameters

Offset is a decimal number in points or inches. Standard subscript is 3 points (0.039 inches).

# Description

Subscript is a character-based formatting code which allows you to shift text a specified distance below the baseline.

If points are used (rather than inches), you must follow the offset with p (for example, 3p).

Please note that Subscript is a paired code. All text to be subscripted must be included between the Begin Subscript Code <SB> and the End Subscript Code </SS>.

# Examples

The word <SB:3p>bananas</SS> is subscripted 3 points below the baseline.

The word <SB:0.039>bananas</SS> is subscripted 0.039 inches (3 points) below the baseline.

Superscript

# Syntax

 <SP:Offset> . . . </SS>

# Parameters

Offset is a decimal number in points or inches. Standard subscript is 3 points (0.039 inches).

# Description

Superscript is a character-based formatting code which allows you to shift sections of text a specified distance above the baseline.

If points are used (rather than inches), you must follow the offset with p (for example, 3p).

Please note that Superscript is a paired code. All text to be subscripted must be included between the Begin Superscript Code <SP> and the End Superscript Code </SS>.

# Examples

The word <SP:3p>cheesecake</SS> is superscripted 3 points below the baseline.

The word <SP:0.039>cheesecake</SS> is superscripted 0.039 inches (3 points) above the baseline.

Tab

# Syntax

 <TB>

# Parameters

None.

# Description

Tab functions identically to a Tab in a word processor or on a typewriter. It tabs the text over to the next tab stop (as defined in the Tab Set code or default every ½ inch).

You must use the <TB> code in your flat files if you want a tab in your infobases. The ASCII 9 code, generated in most text editors by pressing the TAB key, is ignored when the flat file is imported into an infobase.

# Examples

<TB>This paragraph is tabbed over to the first tab stop in the finished infobase.

Tab Set

# Syntax

 <TS:Location,Justification,Leader>

# Parameters

Location is a decimal number in inches (XX.xxx) or Center or Right.

Justification sets the justification for the tab and may be:

Code Explanation

NM Normal (or Left)

CN Center

RT Right

CA Character Aligned (Decimal Tab)

Leader specifies the dot leader for the tab and may be:

Code Explanation

NO None

DS Dots & Spaces

DO Dots

DA Dashes & Spaces

UN Underline

# Description

Tab Set is a paragraph-based code which allows you to set and define the attributes of tab settings. If no Tab Sets are defined, default tabs are set every ½ inch on the ½ inch.

One tab set consists of three items: the location of the tab set (measured in inches away from the left margin), the justification of the tab set, and the dot leader (if any) associated with the tab set. One Tab Set code may contain up to 40 tab sets, with each tab set containing those three items. Each tab set in a tab set code should be separated by semi-colons. See the Examples for more information.

Location specifies the position of the tab relative to the left margin. Location is measured in inches.

The last two Locations specified in a Tab Set code may be the following:

Center Maintains a center-aligned tab in relation to the margins rather than a fixed point. Similar to the Center Line function in many word processors. Usually uses a CN (center) justification and NO (none) dot leader.

Right Maintains a right-aligned tab in relation to the right margin rather than a fixed point. Usually uses a RT (right) justification and NO (none) dot leader.

Justification is a two-letter code which specifies the justification of the tab set.

Leader is a two-letter code which specifies the type of dot leader associated with the tab set.

# Examples

<TS:0.25,NM,NO;0.75,NM,NO>This code creates two tab sets: one at 0.25 inches and the other at 0.75 inches. Both have normal justification and no dot leader. Additional tabs default every ½ inch on the ½ inch past the last tab set. Note that there is a semi-colon between the two specified tab sets.

<TS:1.5,CN,NO;5.5,RT,DS>This code creates two tab sets (up to 40 may be set) which could be used in a Table of Contents. The first set is at 1.5 inches, is center justified, and has no dot leader. The second set is at 5.5 inches, is right justified, and has a Dots & Spaces leader (. . . . ). Additional tabs default every ½ inch on the ½ inch past the last tab set.

<TS:0.25,NM,NO,Center,CN,NO,Right,RT,NO>This code uses the special Center and Right tabs, which stay centered or right justified relative to the window size (not to a fixed point, as other tab sets do).

Tables

# Syntax

 <TA:Number of Columns,Widths for Each Column,Table Attributes><RO:Row Attributes><CE:Cell Attributes>Cell Text</TA>

# Parameters

Tables must be contained within a single record. It is recommended that any single record contain only one table.

Tables are comprised of three primary codes: the Table Definition, the Row Delimiters, and the Cell Delimiters.

<TA> is the Table Definition. TA has two optional parameters and set of optional attributes:

Number of Columns — Columns is an integer and specifies the number of columns for the table. (One column is required for each cell; cells may be merged if necessary.) Number of defined columns may not exceed 32. Note: Use this parameter only if you need fixed width cells.

Widths for Each Column — Width of each column in the table, separated by commas. Widths may be specified in [[1]](#endnote-1)inches or by [[2]](#endnote-2)percentage of the window. The total number of widths specified must equal the number of columns (if you have 3 columns, you must have 3 widths). Total width for columns may not exceed 22 inches (including the horizontal gap between columns). Note: Use this parameter only if you need fixed width cells.

Note: The number of columns and the widths for the columns are optional. If these parameters are not included, the table will be generated from the number of cells actually used in the table. Each cell will have the same width. (22 inch total maximum width; up to 32 columns.)

Table Attributes — Tables may contain several attributes. These attributes affect the entire table and set default borders for the cells in the table. See Table Attributes for more information.

<RO> is the Row Delimiter. <RO> defines the start of a row. The current row is ended when a new row begins (or you may explicitly end the row with </RO>). A table may have any number of rows. Any row in the table may be a header row. See Row Attributes for more information.

<CE> is the Cell Delimiter. <CE> defines the start of cell in a row. The current cell is ended when a new cell begins (or you may explicitly end the cell with </CE>). The number of cells in a row cannot exceed the number of columns defined in the Table Definition. A cell may contain several optional attributes. See Cell Attributes for more information.

Text within a cell may use any of the standard character and paragraph formatting options, as well as links.

Tables must be ended with the </TA> code.

Table Attributes

Table attributes affect the entire table. Some options may be overwritten in individual cells. Table attributes may be listed in any order and may include:

JU:Justification — Sets the justification (horizontal alignment) for the table. Justification may be one of the following:

LF — Left. This is the default setting. If no justification is specified, the table is left justified.

CN — Center.

RT — Right.

IN:Left,Right,First — Sets the left and right indent for the table (first line indents are ignored). The entire table indented; indents for paragraphs within individual cells are specified by using the Indent code within the cell. Uses the standard Indent code.

VG:Value — Sets the vertical gap for text within the table. This option must be set to use borders for the table (left and right borders can be no wider than the vertical gap). Value is a decimal number in inches (XX.xxx). This roughly corresponds to the Cell Margins in Folio Views.

HG:Value — Sets the horizontal gap for text within the table. This option must be set to use borders for the table (top and bottom borders can be no wider than the horizontal gap). Value is a decimal number in inches (XX.xxx). This roughly corresponds to the Cell Margins in Folio Views.

BR:Border Options — Sets the border for the table. Uses the standard Border code options for the outside borders for the table and adds the following codes for the default cell borders:

HZ:Width,Inside Space,Color — Sets the default horizontal border for cells within the table.

VT:Width,Inside Space,Color — Sets the default vertical border for cells within the table.

SD:Shade — Sets the shading for the table. Shade is an RGB color combination. Uses the standard Shade code.

AP:Value — Sets the space after the table. Value is a decimal number in inches (XX.xxx). Uses the standard After Paragraph code.

BP:Value — Sets the space before the table. Value is a decimal number in inches (XX.xxx). Uses the standard Before Paragraph code.

Row Attributes

Row attributes affect single rows in the table. Row attributes include:

HE — Marks the row as a Header Row. Tables may have more than one header row.

Cell Attributes

Cell attributes affect individual cells in a table and may override the default table attributes. Cell attributes may be listed in any order and may include:

MD:Integer — Number of cells to merge down.

MR:Integer — Number of cells to merge right.

VA:Alignment — Sets the Vertical Alignment for the cell. Alignment may be one of the following:

CN — Center. This is the default setting. If no alignment is specified, the text in the cell will be centered top to bottom.

TO — Top. Keeps the text in the cell as close to the top as possible.

BO — Bottom. Keeps the text in the cell as close to the bottom as possible.

HI:Height — Sets the cell height. Height is a decimal number (XX.xxx). Set HI:0 for auto height.

BR:Border Options — Sets the border for the cell. This option overrides the default cell borders set in the table definition. Uses the standard Border code options.

SD:Shade — Sets the shading for the table. Shade is an RGB color combination. This option overrides the default table shading set in the table definition. Uses the standard Shade code.

HC — Marks the cell as a header cell. If this code is used in any cell, all cells in the column become header cells. (For consistency and readability, this code should be used in the first cell in a column; however, this is not required.)

KN:Option — Keep Next. This code affects how the table is displayed and printed. This option keeps the current cell and the next cell down together. Option may either be + or ñ (on or off). Note that this functionality is not supported in the current Folio Views product. Other InfoApp developers may wish to support it; Folio may support it in the future.

KT:Option — Keep Together. This code affects how the table is displayed and printed. Text within the cell is not broken across screen or page boundaries (the entire cell is kept together). Option may either be + or ñ (on or off). Note that this functionality is not supported in the current Folio Views product. Other InfoApp developers may wish to support it; Folio may support it in the future.

# Description

Table is a formatting code which allows you to create tables in an infobase. Cells in a table may contain other character formatting and paragraph formatting codes, and certain paragraph formatting codes affect the entire table (such as justification).

Within a table, cells may be set to a specific size, merged, shaded, aligned, and bordered. Heading rows and heading columns may also be specified. A heading row stays on the screen as the table is scrolled down. A heading column stays on the screen as the table is scrolled to the right. These heading options allow for the display of labels for rows or columns on long tables.

Please note that Table is a paired code. Information in the table must be included between the Table Definition Code <TA> and the End Table Code </TA>.

See the appendix RGB Color Combinations for a list of the RGB values for the standard colors available through the Folio Views dialogs.

Note: Tables must be contained within a single record. Using Before Paragraph and After Paragraph spacing, it is possible to simulate a table spanning record boundaries. (You may wish to divide table information in this way for searching purposes.)

Note: For those converting SGML documents to flat file (or any other document markup language), you should be aware of the following:  
  
•The only required codes are the <TA>...</TA> pair, <RO>, and <CE>. All other codes used in tables are optional.  
•Highly desired codes include table borders (which also require setting the Horizontal and Vertical gap to have borders within the table), table justification, and row and column headers.  
•Useful codes include the merge codes, table indent codes, and before and after spacing codes.  
•Cells may contain any formatting codes, text, and objects allowed within the main body of an infobase, with the exception of levels and record codes. Cells may contain multiple paragraphs.  
  
As you can see, you do not need to support everything in your SGML file or add a huge attribute list to every table you create. Use the attributes that are necessary for your tables, and then map them to the appropriate Folio flat file codes (using OmniMark or some other pattern matching utility).

Note: If you export an infobase containing a table with no fixed width columns, an extra semi-colon is added to the export code. This does not cause problems on import. The code comes out as <TA:; BR:...(rest of table definition)>

# Examples

 <RD>A basic table (4 rows, 4 columns, no formatting, auto-defined column widths)  
<RD><TA>  
<RO><CE>row 1 column 1<CE>row 1 column 2<CE>row 1 column 3<CE>row 1 column 4  
<RO><CE>row 2 column 1<CE>row 2 column 2<CE>row 2 column 3<CE>row 2 column 4  
<RO><CE>row 3 column 1<CE>row 3 column 2<CE>row 3 column 3<CE>row 3 column 4  
<RO><CE>row 4 column 1<CE>row 4 column 2<CE>row 4 column 3<CE>row 4 column 4  
</TA>

 <RD>Same table as the first example, with borders around the table and between cells.  
<RD><TA:VG:0.075;HG:0.075;BR:AL:0.022,0.03,FC:0,0,128,HZ:0.015,0.03,FC:0,128,0,VT:0.015,0.03,FC:0,128,0>  
<RO><CE>row 1 column 1<CE>row 1 column 2<CE>row 1 column 3<CE>row 1 column 4  
<RO><CE>row 2 column 1<CE>row 2 column 2<CE>row 2 column 3<CE>row 2 column 4  
<RO><CE>row 3 column 1<CE>row 3 column 2<CE>row 3 column 3<CE>row 3 column 4  
<RO><CE>row 4 column 1<CE>row 4 column 2<CE>row 4 column 3<CE>row 4 column 4  
</TA>

 <RD>Same table as the second example, with fixed column widths, a different colored cell border, and paragraph formatting codes in a set of cells.  
<RD><TA:4,1,1.25,1.25,1; VG:0.075; BR:AL:0.022,0.03,FC:0,0,128,HZ:0.015,0.03,FC:0,128,0,VT:0.015,0.03,FC:0,128,0>  
<RO><CE>row 1 column 1<CE: BR:LF:0.021,0.03,FC:255,0,0,TP:0.021,0.03,FC:0,0,128,RT:0.021,0.03,FC:255,0,0,BT:0.021,0.03,FC:255,0,0>row 1 column 2<CE>row 1 column 3<CE>row 1 column 4  
<RO><CE><JU:CN><BR:AL:0.021,0.03,FC:255,0,255><SD:192,192,192>row 2 column 1<CE><JU:CN><BR:AL:0.021,0.03,FC:255,0,255><SD:192,192,192>row 2 column 2<CE>row 2 column 3<CE>row 2 column 4  
<RO><CE>row 3 column 1<CE>row 3 column 2<CE>row 3 column 3<CE>row 3 column 4  
<RO><CE>row 4 column 1<CE>row 4 column 2<CE>row 4 column 3<CE>row 4 column 4  
</TA>

Text Styles

# Syntax

 <ST:Name,Type,Character Based formatting codes>

# Parameters

Name may be up to 127 characters long. This name is used to identify the style and to apply it in the Document File.

Type specifies whether the style is used for a Link or for standard text. May be one of the following:

Code Explanation

CS Character

LK Link

Character Based formatting codes are optional. They are used to apply character-based attributes to the text to which the style is applied. These codes include all codes as listed under Character Based Formatting. Individual codes must be separated by commas.

# Description

Text Style codes are used to define Character Styles and Link Styles. Only character-based formatting codes may be used in Text Style definitions (refer to Character Based Formatting for information on character-based codes).

Styles must be defined in the Definition file in order to function when used in the Document file.

Type is the two-letter code describing the type of style being defined. Style type MUST be specified. Incorrect style types (i.e. — character styles being used for link styles) may cause errors when the infobase is created.

When non-paired Character Based formatting codes (Bold, Italic, Underline, Strikeout, Hidden, Font, Point Size, and Color) are used, you may need to specify whether that attribute is to be ON, OFF, or to be unaffected (default). This is especially important for Bold, Italic, Underline, Strikeout, and Hidden. When one of these attributes is ON, it affects all text to which it is applied. When one of these attributes is OFF, it turns off that attribute entirely (even if that attribute was applied in a Level Style or a Paragraph Style). When one of these attributes is specified to be DEFAULT, then the text to which the style is applied is not affected by that attribute (i.e. — if a Paragraph Style turned Italics on before the Text Style was applied, then italics would remain on; if italics were off, they would remain off). In other words, unless specifically turned on or off, they remain in their current default state. See the following examples for further explanation.

See the appendix RGB Color Combinations for a list of the RGB values for the standard colors available through the Folio Views dialogs.

# Examples

<ST:Fancy,CS,FT:Arial,PT:12, BC:100,0,0,FC:0,0,255,BD+,IT+,UN->

This code defines a Character style (CS) named "Fancy" in Arial 12 point font, with a light red background (100,0,0) and a blue foreground (0,0,255). It is Bolded and Italicized. Underline is turned OFF. All other attributes remain in their default state.

<ST:Jump,LK,FT:System, BC:170,170,0,IT+>

Defines a Link style (LK) named "Jump" which is in the System font on a yellow background. The text is italicized. All other attributes l remain in their default state.

Title

# Syntax

 <TT:"Title text">

# Parameters

Title text may be up to 127 characters long. It may not contain any other flat file codes.

# Description

Title is one of the Infobase Information Fields and is designed to hold the Title of the infobase. This title appears on the Title Bar as the name of the infobase, if included. If not included, the filename of the infobase is used. Text contained within this field may only be 1 to 127 characters long (including spaces). Please refer to Infobase Information for details.

Note: If a Title is not specified, the infobase path and file name is used as the infobase title.

# Examples

<TT:"The History of Mars from Yesterday to the Present">

<TT:"HyperTrend Encyclopedia, Cat to Dog">

Title Page

# Syntax

 <TP:"Folio Object Name">

# Parameters

Object Name refers to a Folio Object as defined with the Object Definition <OD> code in the Definition File.

# Description

The title page appears briefly each time the infobase is opened. The title page generally contains the name of the infobase, the organization which created the infobase, and the copyright for the infobase. Title pages are stored as objects in the infobase, usually as a bitmap or metafile.

Note: The Title Page code MUST be placed in the Definition File after the Object Definition code in which the object is defined.

Note: Title pages must be Folio Objects (bitmap or metafile). OLE objects are not supported. As such, the Class Name, which was required in flat file 3.x is not required in flat file 4.x. Class name, if used, is ignored.

# Examples

<TP:"Introduction">

<TP:"About Us">

Underline

 See Attributes.

User Extension

# Syntax

 <UX:"Type Name",User Defined Information>

# Parameters

Type Name is used to identify the type of information contained in the code (similar to how the Class Name parameter is used in the Object <OB> code). The type name is used to identify the custom chain filter which will process the information.

User Defined Information is any type of information that you choose to process through a custom chain filter. Information that is not handled by a chain filter is thrown away.

Note: If angle brackets < and > are used within the User Defined Information, they must form perfect pairs (for every open bracket, there must be a close bracket). Brackets are not counted when they are in full quotes (as may be required for literal text). As soon as an unpaired close bracket is encountered, the UX code is ended.

# Description

The User Extension code allows you to add custom information to a flat file and process that information with a custom filter that is chained to the flat file filter.

For example, you might have several tables in a text-based typesetting format. Rather than converting all of the tables to flat file via FSR or some other search and replace language, you could create a chain filter that filters the typeset format directly into an infobase. This would allow you to include the tables in your flat files without modifying the markup of the tables.

Folio provides a sample chain filter that may be used with this code. Use TableText for the Type Name and place comma, tab, or other character-delimited data in the body of the code. Using the TableText chain filter with the Folio Flat File filter, you can create simple tables based on character delimited text. The TableText chain filter allows you to choose the character used to delimit the text for cells. Carriage-return-line-feeds (CRLFs) are used to delimit rows.

# Examples

<UX:"PseudoTable",R1,C1-Task-C2-Priority-R2,C1-Finish Assignment-C2-A1-R3,C1-Eat Cookies-C2-Z26>

<UX:"TableText",Cell 1,Cell2,Cell3,This table,is delimited,by commas>

User Link

# Syntax

 <UL:"Type Name",User Defined Information>

# Parameters

Type Name is used to identify the type of information contained in the code (similar to how the Class Name parameter is used in the Object <OB> code). The type name is used to identify the custom chain filter which will process the information.

User Defined Information is any type of information that you choose to process through a custom chain filter. Information that is not handled by a chain filter is thrown away.

Note: If angle brackets < and > are used within the User Defined Information, they must form perfect pairs (for every open bracket, there must be a close bracket). Brackets are not counted when they are in full quotes (as may be required for literal text). As soon as an unpaired close bracket is encountered, the UL code is ended.

# Description

The User Link code allows you to add custom information to a flat file and process that information with a custom filter that is chained to the flat file filter. The User Link code is distinguished in name only from the User Extension code: both provide the same basic functionality.

For example, if you did not like the way Folio Views handles URL links, you might use the User Link code to create your own URL link that opens up a Web browser and passes it the appropriate URL.

# Examples

<UL:"My URL Link",neato-browse.exe,www.nextpage.com>

Version Information

# Syntax

 <VI:Folio,File Type,Version Number>

# Parameters

Folio identifies that the file was output by the NextPage, Inc. flat file filter (or compatible).

File Type identifies the file as a flat file (FFF) or definition file (DEF).

Version Number identifies the major, minor, and build number of the filter used in creating the flat file.

# Description

Version information is used to speed up auto-detection of Folio flat files through the Workbench. By identifying the file type and version number at the top of the file, the Workbench can select the most appropriate filter, even if the extension is something other than FFF or DEF.

This code appears at the top of both the document file and definition file.

# Examples

<VI:Folio,DEF,4.3.118>

<VI:Folio,FFF,4.3.118>

Wallpaper

# Syntax

 <WP:Pane,"object name",Display Options>

# Parameters

Pane is a three letter abbreviation for the pane the wallpaper should appear within Folio Views. Pane may be set to one of the following:

|  |  |
| --- | --- |
| Abbreviation | Control |
| DOC | Document |
| HIT | Hit List |
| REF | Reference |
| TOC | Contents |

Object Name refers to an Object as defined with the Object Definition <OD> code in the Definition File.

Display Options is optional. Currently, only one option is available. Use 1 to have the infobase content scroll over the wallpaper. If no display option is set, then the wallpaper scrolls with the infobase content. Display options may be set for all panes except the reference pane.

# Description

Wallpaper displays the specified object in the specified pane in Folio Views. The wallpaper appears as a background for text and other images in the pane pane, similar to wallpaper used on web pages.

Currently, there is no user interface in Folio Views for setting the wallpaper. This option may only be set in Folio flat file.

The code may appear in either the .DEF file or in the .FFF file. If used in the .DEF file, it must come after the object definition code for the wallpaper object. If used in the .FFF file, it must come before the first record code.

Wallpaper objects are tiled as necessary to fill the pane. As such, they should have seamless borders at the top and bottom. If they are not wide enough to fill the pane, they should also have seamless borders at the left and right.

By default, wallpaper objects are scrolled with the text (as you scroll through the infobase, the wallpaper appears to move). Setting the display options to 1 creates a fixed wallpaper which does not scroll (the text in the infobase appears to scroll over the top of a motionless background). The option to not scroll the wallpaper is valid for all panes except the reference pane.

Note: Setting the display option to 1 (so the image does not scroll with the text) may cause decreased scrolling speeds.

Wallpaper generally should not contain text or or images which interfere with reading the text. For most uses, they should use light colors. If you set all the text in your infobase to light colors, then you can use a dark wallpaper in the document pane.

# Examples

<WP:DOC,"My Wallpaper",1>

<WP:HIT,"speckled lavender waves">

<WP:TOC,"funky patterns",1>

<WP:REF,"reference wallpaper">

Web Link

# Syntax

 <WW:Style name,"URL">  
 .  
 .  
 .  
</WW>

# Parameters

Style Name refers to a Link Style (LK) as specified in the Text Style <ST> code in the Definition File.

URL is the address to the web site you wish to link to. For cross-platform (Windows & Macintosh) infobases, you must use a full URL (such as http://www.nextpage.com). If the infobase will only be used on Windows, you may use an abbreviated URL (such as www.nextpage.com).

# Description

Web Links allow you to connect to your web browser and to the specified URL. When the link is followed, the default web browser is launched and the URL is passed into it.

Please note that web link codes are paired codes. Text or other information to be marked as a web link launch point must be entirely enclosed by the begin web link code <WW> and by the end web link code </WW>

Note: When flat files are exported from Folio Views or Extract, the filter exports an <EL> code to end all links (rather than </WW>). Either code may be imported.

# Examples

This is a <WW:Web,"http://www.nextpage.com">sample web link</WW>. It opens a web browser and connects to the NextPage, Inc. web site.

<WW:Web,"http://www.lotus.com">Click here to go to the Lotus web site.</WW>

1. Use inches to set a true fixed width cell (such as 1.25 inches). [↑](#endnote-ref-1)
2. Use a percentage (such as 25%) to force the cell to always occupy a minimum amount of space in the window. As the window changes size, so does the cell.

   Note that the total percentage for all cells specified may exceed 100% (in a five column table, each column could occupy 25% of the screen). This arrangement can be used to force part of the table to remain off of the screen unless specifically scrolled to. [↑](#endnote-ref-2)