



i-net
Crystal-Clear**X** version 10

Fonts

Understanding problems and solutions

Table of Contents

1 Content.....	3
2 Terms.....	4
2.1 Native Fonts.....	4
2.2 Logical Fonts.....	4
2.3 Dynamic Font Names.....	4
2.4 Font Embedding.....	5
2.5 Font Auto-Scaling.....	5
3 Problems.....	6
3.1 Internationalization.....	6
3.2 Font Variants.....	6
3.3 Missing Fonts on the Client.....	6
3.4 Floating Font Metrics and Fixed Point Font Metrics.....	7
3.5 Reports for the Internet.....	8
3.6 Reports for your Intranet.....	8
3.7 Existing Reports with Native Fonts.....	8
3.8 Larger Fonts after Migrating Templates from Crystal Reports.....	8

1 Content

On the surface, fonts seem to be a simple thing. However, they often can cause great problems. i-net Crystal-Clear offers a number of possibilities to avoid these problems. Before creating a report, it is good to understand the problems and to take them into consideration while creating reports.

2 Terms

2.1 Native Fonts

A native font is a TrueType font which is stored in a *.ttf or *.ttc file. So that i-net DesignerXML can use the font, it must either be installed on the system or must be in the folder of embedded fonts (see: [Font Path](#)). A native font can only display the characters stored in the font. All other characters will either be shown as little boxes, as question marks or not at all.

2.2 Logical Fonts

A logical font is the combining of multiple different native fonts of a local computer to a new font. The logical fonts in Java contain all typical Unicode symbols. If exotic signs (like Klingon, for example) are needed and you have a font which contains them, they can be added to the font configuration.

See <http://java.sun.com/j2se/1.5.0/docs/guide/intl/fontconfig.html> for more information on this.

The following logical fonts exist: Serif, SansSerif, Monospaced, Dialog, and DialogInput. How these logical fonts look can vary from system to system. This especially is noticeable when comparing Asian to European computers. Through the combining of multiple fonts, the line height can vary. For example, under Windows, SansSerif is mapped to Arial, but has a different line height than Arial

2.3 Dynamic Font Names

The font name of a given element can be set via formula at run time. The font name could even be stored in the same table as the data itself. If the element contains HTML formatted text (see: [Text Interpretation](#)), the font name can also be directly in the data itself.

2.4 Font Embedding

The fonts are embedded with the report output at run time, not at the time of the report design. Font embedding is currently only possible for the output formats Java Viewer, PDF, and Postscript 3. Only fonts which allow embedding can be embedded. The license information is in the fonts themselves. The embedding of fonts must be activated and the font path must be set. Font embedding greatly increases the size of the report output. With font embedding, more bandwidth, memory, and CPU are required. The payoff is that the fonts used in the report no longer need to be installed on the client when the Java Viewer shows the reports.

2.5 Font Auto-Scaling

Font Auto-scaling is an algorithm used by the Java Viewer in which it tries to compensate for the differing Font Metrics on the Report Server and the Client. The Font Metrics can vary between different systems like Windows and Linux by 30%. With font auto-scaling, a piece of text is re-scaled on the client until it has the size expected for it by the server. There can be negative effects, if the server or the client does not have a character and it has a width of 0.

3 Problems

3.1 Internationalization

Say you want to distribute your report internationally. Your database is Unicode-compliant. Can your fonts show all required characters? The font Arial can display all European languages, for example. But it does not contain Asian symbols like Chinese or Japanese. The possible solutions to this problem:

- ▷ Use logical fonts.
- ▷ Use dynamic font names.

3.2 Font Variants

On various client computers it could be that various versions of the same font are installed. Even the well-known font Arial differs from Windows version to Windows version. This not only is true in regards to the number of characters, but also in regards to minor differences in character sizes.

The possible solutions to this problem are:

- ▷ Font embedding - if you want to embed the font itself in the report (Note: take the font's license into consideration)
- ▷ Font auto-scaling
- ▷ Leave enough room for the text labels, since a variation of 1-2 pixels in the width of text labels on different computers can be normal.

3.3 Missing Fonts on the Client

If the server uses a font which does not exist on the client, then Java tries to map this font to an existing font. This can lead to a whole number of problems. The mapped font can have a different size (which can often be as much as 30% different). Individual characters may be missing and of course the font itself looks different. The possible solutions to this problem are:

- ▷ Font embedding
- ▷ Use logical fonts (usually they include all needed characters and at least look similar)
- ▷ Font auto-scaling - however, this will only compensate for the differing sizes.

3.4 Floating Font Metrics and Fixed Point Font Metrics

The width of a character is not a whole number but a floating point decimal number. But since only whole pixels are displayed on the screen or by the printer, these must be converted. With Fixed Point Font Metrics, a pixel is either black or white, and a character will always have a whole number of pixels as its width. This causes a higher contrast, a more “pixeled” look, and some characters can seem to “stick” to each other.

With Floating Font Metrics, the transitions are interpolated. This means a character can have a width with fractions of pixels. This may cause a slightly blurry look, but the characters will look smoother and will no longer “stick” to each other.

On a printer with 300 DPI, it is hardly possible to tell the difference between the two methods. Both types create a printed page with the look of Floating Font Metrics. This is why i-net Crystal-Clear (and PDF) uses Floating Font Metrics to render the reports, achieving a better printing image. On a screen with 96 DPI, there are major differences, however.

The problem occurs because i-net DesignerXML still works with Fixed Point Font Metrics. This can especially be seen when using many identical letters in a row (look for example at 50 small “i”s in a row). These can cause a different width in the design view and in the report preview. This can especially be aggravating with text fields.

Design View in i-net DesignerXML (Fixed Point Metrics)



Preview in Report Viewer (Floating Font Metrics)



The solution for this is:

- ▷ Do not design text fields as too “tight”, and always take a look at the preview.

3.5 Reports for the Internet

Since a high number of greatly differing systems can be expected, you should only be working with

- ▷ Font Embedding OR
- ▷ Logical fonts with font auto-scaling

3.6 Reports for your Intranet

Only in homogeneous intranet environments can you work with

- ▷ Native fonts AND
- ▷ Without font embedding

3.7 Existing Reports with Native Fonts

Say you have a large amount of reports with native fonts and want to show these with logical fonts without having to manually edit each of your reports.

The solution to this is to:

- ▷ Set the property “UseNativeFonts” to false and delete the value of the property “FontPath”. This will cause logical fonts to be used for all reports.

3.8 Larger Fonts after Migrating Templates from Crystal Reports

Certain versions of Crystal Reports seem to display fonts somewhat smaller than the actual font size, this is especially the case when exporting to PDF. For example, setting a text label to Arial size 12 and then exporting to PDF with Crystal Reports seems to produce a text with the font Arial, size 10.6. To remedy having your reports seem too large when exporting to i-net Crystal-Clear, the ReportReader tool uses a file called “fontmappings.xml” to automatically map fonts encountered in the templates to slightly smaller fonts.

- ▷ In order for this XML file to be used, simply make sure it is located in the same folder as your ReportReaderNET.exe.
- ▷ See the Crystal Reports Migration Guide for further information