Xara X Import/Export Filters

The family of XaraX illustration programs currently use the Xar format as its native document format (with the .xar extension) and it can also import and export size optimised files for use on the internet (with the .web extension).

Xara X supports a system of plug-in import/export file converters. These filters will work by translating to or from Xar format data. The interface is designed to be as simple as possible with both import and export requiring only two functions each. Such file converters would be used to read and / or write alternative file formats, such as PDF, SVG etc.

Export

The export process works by Xara X passing the data to the filter as Xar format data and the filter translating it as required and writing it to a file. The filter is able to define in a XML ‘capabilities’ file how objects in the document are represented in the Xar format data. This allows the filter to handle complex objects (e.g. complex fills, transparency, blends, shadows etc) by having Xara X convert them to simpler ones that are more suitable for the destination format (e.g. bitmaps or simple paths). The details are as follows:

Xara X calls the filter's PrepareExport method to retrieve information about exactly what is wanted in the Xar format data (the capabilities). The filter can display UI allowing the user to control features of the filter and can vary the capabilities to suit). So, for example, the PDF exporter that’s included with Xara Xtreme specifies that it can accept linear and radial graduated fill types, but cannot understand the more advanced fill types such as conical or three-point fills. Xara thus converts all higher level fill types to a bitmap fill which PDF can cope with. Or, another example, the user can specify whether they require PDF 1.3 standard (that allows no transparency), or the PDF 1.4 standard (that does support transparency). This is defined in the capabilities XML passed back to Xara, so that no transparency information is passed to the exporter if the user selected PDF 1.3. In this case all areas of the document containing transparency are rasterised (sometimes called flattening) to non-transparent bitmaps. This same process can be applied to the more complex vector objects. So for example, the converter can ask that all blends or mould objects are converted to simple bezier vector outlines prior to export.

This capabilities system therefore allows export converters to be implemented gradually, and that Xara X does most of the hard work, where required, of converting more complex vector object types into simpler types before being passed to the export filter. Then Xara X calls the filter's DoExport method to perform the actual export. The Xar format data is passed to the filter via a standard IStream interface.

Import

The import process works in a similar way to the export process. Xara X passes the filename of the file to be imported and the filter must read the data and translate it into Xar format data. The details are as follows:

Xara X calls the filter's HowCompatible method to discover whether the particular filter can handle the file. If the filter type was not specified in the XaraX import dialog (or a file was dragged and dropped) then Xara X will call this method on each registered import filter and will use the one that scores the highest.

Xara X calls the filter's DoImport method to perform the actual import. The filename of the file being imported is passed to the filter and the Xar format data must be written to the standard IStream interface.

Implementing a Filter Object

When using C++ (or any other language that allows linking to C++ static libraries) the easiest way to implement a Xar file Reader or Writer is to use the XarLib library. The library provides access to a Xar file at the record level for both the reading and writing of Xar format files. The library is responsible for handling the ZLib compression and certain other features of the format (e.g. atomic and essential records) leaving you to deal with interpreting or writing the records without having to worry about the low-level details.

The library is currently only available as header files and static libraries built with Microsoft Visual C++ 6 (multi-threaded C runtime versions for debug and release). Other versions, including source code that can be built on other platforms, should be made available shortly.

Download the XarLib Library (643 KB)

The XPFilter archive contains a simple example filter object (XPFStats) that uses the XarLib library to handle the reading and writing of the Xar format data and a test bed program (FilterDemo) for testing the filters.

Download the XPFilter archive (106 KB)

XPFStats

The XPFStats project is an example filter built using Microsoft VC++ 6, ATL and the XarLib library. On export, it creates a text file containing a list of how many times each record "Tag" appears in the Xar data. On import it converts a text file into a Xar document containing a text story consisting of the text from the file. This project uses the XarLib library for both reading and writing of the Xar format data making it a useful example of how to use the library.

FilterDemo

FilterDemo.exe is a simple test-bed application for developing filter objects. It uses the filter objects in the same way as XaraX but it provides the Xar data for export from a Xar file you specify and on import it writes out the Xar data the filter object creates to a file.