Introduction

The SDK supplied with Infix Server onwards enables programs to be written which incorporate the functionality of Infix Server. Documents can be created or edited using the SDK in just the same way as the command line version of Infix Server.
The SDK has been kept deliberately sparse. It contains only a handful of calls, which to a large extent mimic the behaviour of the command line version. This means that most of the general configuration of Infix Server is still performed using the XML command file used by the command line version.
This document provides a brief and technical description of the components of the API. It should be read in conjunction with the User Reference Manual supplied with Infix Server and makes frequent references to information contained therein.

Supported Platforms

The SDK is provided for Windows (98/2000/XP), Linux (x86), MacOSX and Solaris (Sparc). Under Windows, the SDK is a DLL, which can be linked into any C/C++ application. Under Solaris, MacOSX and Linux, the SDK is a static link library.
Build environments
The SDK (& Infix Server) was built using the following environments:

```
Solaris: gcc 3.3.2, 32-bit, Solaris 8 (SunOS 5.8)
            Requires dynamic stdc++ library (included in distribution)
Linux: gcc 3.2, 32 bit, Mandrake Linux 9.0 3.2-1mdk
            Statically linked - no special dynamic libraries required.
MacOSX: gcc 3.3, MacOSX 10.3.7
            Statically linked - no special dynamic libraries required.
Windows: Microsoft Visual Studio.Net
            Requires ICU dynamic libraries - included in distribution
```

Licensing

The SDK shares the same license as Infix Server. A single license entitles the use of both Infix Server and the SDK on a single host.
Like Infix Server, the SDK requires the presence of a valid hardware or software key, otherwise it operates in demo mode only. In demo mode, a watermark is added to edited pages. This watermark will not invalidate the output but will render it unsuitable for use by an end user.
Contact Iceni at **www.iceni.com** to purchase additional hardware or software keys should you need them. Discounts can be arranged for bulk purchases or for OEM licensing. Iceni can also arrange the supply of special builds of the SDK, which do not require keys.

Using Infix Server SDK

To build a program based upon the SDK only the main SDK library is required. This library includes the auxiliary libraries jpeg, png, flate decompression, tiff and icu (except MacOSX). These freely available additional sub-libraries are also included in the distribution in source code form.

License Keys

This version of the SDK requires a software-based license key in order to operate. This key should have been provided with the Infix Server distribution when it was purchased.
Infix Server & the SDK will attempt to load the license file from 'infixserver.key' in the current working directory. This default may be changed using the SDK to any desired pathname.

Example Applications

InfixServerDLLTest

This application shows how to run an XML command file to produce a PDF using the Infix Server API.

The XML command file uses the search and replace and image insertion functionality to produce an output PDF from an input template PDF.

“C” Language API

The “C” language declarations of the API are contained in the header file **isinterface.h**.

void* isInterfaceCreate(char* configFile, char* keyFName, char** results)

This must be the first call to the API. It initialises the SDK’s data structures, loads the configuration file and validates the host key file.

On success returns an abstract pointer. This pointer must be passed to all subsequent calls in this part of the API since it identifies the instance of Infix Server.

Throughout the rest of this section, this unique pointer is referred to as “vstate”.

void isInterfaceDestroy(void* vstate)

This must be the final call to the API. It destroys the vstate and all its data structures.

ICError isRunXMLFile(void* vstate, char* fname, char** results)

This function is used to run an XML command file.

The results that are returned to the caller are in XML format and must be destroyed using the API function **isResultsDestroy**.

A non-zero return indicates that an error has occurred. In this case, processing will have stopped at the document that caused the error.

ICError isRunXML(void* vstate, char* xmlData, char** results)

This function runs xml commands from a buffer.

The results that are returned to the caller are in XML format and must be destroyed using the API function **isResultsDestroy**.

A non-zero return indicates that an error has occurred. In this case, processing will have stopped at the document that caused the error.

void isResultsDestroy(char* results)

This function is called to destroy the results buffer that is returned to the user in call to **isRunXML** and **isRunXMLFile**.