CITRIX RECEIVER FOR LINUX PLATFORM OPTIMIZATION SDK

CONTENTS

COMPATABILITY WITH RECEIVER

This is Version 13.2 of the Platform Optimization SDK. It is intended for use with Linux Receiver builds in the 13.2 series. There are separate versions of the SDK for each of the released architectures (X86, X86_64, ARMel and ARMhf) that differ only in the included binaries.

WHO SHOULD USE THIS SDK?

The purpose of the SDK is to support developers who are creating plug-in extensions for the ICA engine component (wfica) of Citrix Receiver for Linux. Plugins are built as shareable libraries that are dynamically loaded by wfica. These plugins can perform the following functions:

- Provide accelerated decoding of JPEG and H.264 data used to draw the session image;
- Control the allocation of memory used to draw the session image;
- Improve performance by taking control of the low-level drawing of the session image;
- Provide graphics output and user input services for OS environments that do not support X11.

Plugins for decoding can be built independently of the other types, unless they need to control memory allocation.

To test the sample plugins that can be built with this SDK, they may need to be renamed, and must be copied to the Receiver's installation directory.

The Receiver can also exploit additional plugins for accelerated audio and video codecs. Receiver can be configured to use GStreamer for audio, camera and multimedia functions, so such plugins are standard GStreamer components, and are not covered in the document.

There is also a UI Dialog abstraction which allows for the implementation of core dialogs to be replaced by the reimplementation of the UI Dialog Library.

PLUGINS FOR H.264-BASED SESSION GRAPHICS

XenDesktop version 6 and later support a protocol for presenting the remote session's graphics that uses a combination of H.264 and a proprietary lossless graphics encoding. For maximum flexibility in exploiting on-chip decoders and hardware rendering support, plugins take full control of the decoding, overlay and rendering process.

The details of the interface for these plugins are documented as comments in the relevant header file, *H264_decode.h*, with an unaccelerated sample implementation in the *H264_sample* directory. The sample code (*H264_sample*) was previously that of *wfica*'s fallback implementation that is used when no accelerated plugin is available, and for that reason builds a plugin named *ctxh264_fb.so*. It requires three dynamically-loaded shared libraries that were included in previous Receiver packages with major version 13, or can be built from source code that can downloaded from citrix.com. (Look for "FFmpeg Source Package" in "Additional Components" below "Receiver for LInux".) Note that the environment variable FFMPEG_INCLUDE_DIR must be set before building the sample.

The sample code (*H264_Pi_sample*) is a Raspberry Pi implementation and is provided as a reference hardware implementation.

PLUGINS FOR ACCELERATED JPEG DECODING.

All currently supported versions of XenDesktop and XenApp for Unix can use JPEG to compress portions of the session image. Plugins that support hardware-accelerated JPEG decoding can give a large improvement in graphics performance for sessions not using H.264 session graphics. At startup, *wfica* looks for a plugin named *ctxjpeg.so* to handle JPEG decoding.

The interface JPEG decoding plugins is in the *jpeg_decode.h* header file. The sample code (*jpeg_sample*) is that of *wfica*'s fallback implementation that is used when no accelerated plugin is available, and for that reason builds a plugin named *ctxjpeg_fb.so*

PLUGINS FOR MEMORY ALLOCATION.

Hardware-accelerated plugins for H.264 or JPEG decoding may need to allocate memory buffers with special characteristics, for example using physically contiguous pages. A single plugin component, *KVMEPlugin.so* is used for both straightforward memory allocation and for taking control of drawing the session image. When memory allocation is the purpose of the plugin, only two functions need to be supplied.

The header file for these plugins is *mainloop.h* and the two entry points that must be implemented are *special_allocate()* and *special_free()*. The example code is in the *allocation_sample* directory. Before using this code as a model, pay careful attention to the comments, as parts of the code are there purely for backward compatability with decoder plugins developed for now-obsolete versions of Receiver.

PLUGINS FOR ACCELERATED DRAWING IN X11 ENVIRONMENTS.

In some environments using X11, there may be drawing methods that are faster than the calls to *XShmPutImage()* that are used by default. An implementation of *KVMEPlugin.so* can use an alternative drawing method by providing the *draw()* entry point, which is used to push the session image to the screen. The optional *draw_complete()* entry point may also be provided. When these are used, there is no requirement to also implement the memory allocation functions.

The *allocation_sample* example includes an implementation of *draw()* that is almost identical to the default drawing code for X11.

PLUGINS FOR NON-X11 ENVIRONMENTS.

This SDK includes a separate version of the Receiver engine, *wfica_for_plugins* that is not linked with any X11 libraries. The program requires a *KVMEPlugin.so* that provides video output, mouse and keyboard input, and timer and event detection services. Some features of the X11 version are not yet available: clipboard, sessions with multiple windows ("Seamless Windows"), multimedia and Flash support.

Two example plugin implementations are included: *SDL_plugin* contains an implementation based on the SDL library; and *FB_plugin* contains a version using Linux system calls and device files, using the raw frame buffer for display.

The environments supported with SDL depend on how the library is built, but usually X11 and frame buffer graphics are supported. To use frame buffer graphics run the program from a text console: it is usually necessary

to run as the superuser, or change the permissions on the */dev/fb0* and */dev/mice* files. The frame buffer plugin also requires access to those device files.

wfica_for_plugins does not work when the sound virtual channel is configured to run in a private thread (the default). Before testing, disable the channel ('*ClientAudio=Off*" under "*[ICA 3.0]*") or the thread ("*UseThread=Off*" under "*[ClientAudio]*") in the *module.ini* configuration file.

HDX MULTIMEDIA WITH PLUGINS.

The HDX Multimedia feature can be used with accelerated plugins for X11 and plugins for non-X11 environments. Support is enabled by the configuration entry *UseSubwindows=Off*, placed under the *[WFClient]* heading. The *draw_overlay()* entry point to the plugin will be called to merge video output into the session image.

The Sub-window Interface of the Virtual Channel API supports use of this mechanism in externally-developed Virtual Channels. The relevent header file, *subwindow.h*, is also included in this SDK.

UI DIALOG LIBRARY

The UI Dialog Library (UIDialogLib.so) implements an abstraction layer for the display of dialogs. A dialog's contents is dependent on the contents of the C formsElement structures passed (The formsElement structures are declared in UI_if.h in the toplevel inc directory). The dialog is built dynamically according to the given information in these structures and rendered. The formsElements themselves are essentially label and widget pairs, where the widget can be a text box, check box, set of radio buttons, combo box, multi combo box, button, expander or another label.

The UI Dialog Library is used for the majority of dialogs within the Linux Receiver processes, including the X11 based wfica. The processes storebrowse, AuthManager, PrimaryAuthManager, and ServiceRecord use it for all of their UI. The purpose is to ensure that the UI of these essential processes can be replaced with a toolkit and event loop of your choosing by re-implementing the library. The current implementation of the UI Dialog Library distributed with the Linux Receiver is GTK+. The code for this implementation is provided under the UIDialogLib/GTK directory in this SDK as an example.

The selfservice, configmgr, and X11 wfica binaries require GTK+ for other aspects of their UI other than dialogs and therefore cannot be used with a different implementation of the UI Dialog Library than the provided GTK+ implementation. However, all of their functionality is available via the storebrowse command line interface and the configuration files.

A selection of examples are provided under the UIDialogLib/test directory to showcase some of the widgets and dialogs that may be seen.

For information on the interface itself please read the UIDialogLib.h header in the UIDailogLib/inc directory.

USING THE TEST HARNESS FOR KVME PLUGINS

The SDK includes a simple test harness for testing early versions of *KVMEPlugin.so* for non-X11 environments. The program is built in the *plugin_test* directory. When run, it will load a plugin from the current directory, initialise video output and log mouse and keyboard input events to standard output.

Timer support can be verified by specifying a command argument of *1*. Larger values of the first command argument will build a chain of data-passing stages that use pipes with timers and I/O readiness detection (*select_fd()* function) to transfer data in response to events. A second optional argument sets the initial timer value, for stress testing.

CHANGE HISTORY

Version 13.2:
Re-organise H264_sample code and Makefile changes.

Version 13.1 Release 2:
Bug fixes and addition of this change history.

Version 13.1:
Add source code for the H.264 decoder plugin for Raspberry Pi.
Add support for HDX Multimedia output with non-X11 graphics.
Add additional UI elements and a interface version number to UIDialogLib.
Rename structure members from *private* to *priv* for C++ compatability.