■ Summary

This document describes how to implement video/audio decoding using GStreamer when executing the video playback function of RemoteSDK.
The following slides describe the specific implementation and SampleCode.

■ Setting Environment Variables

SONY

In this document, SampleCode uses original plugin module (esvideosrc/esaudiosrc).
To make GStreamer refer to the original plugin module, set GST_PLUGIN_PATH to the path of the original plugin module in the environment variable settings.
PATH is an additional setting for the directory where GStreamer searches for plugins.
The following Example is implemented on the assumption that the original plugin module is in the current directory of the executable file.

<Example(Windows)>

```
char* gstPluginPathEnv = getenv("GST_PLUGIN_PATH");
if (gstPluginPathEnv != nullptr) {
    std::string gstPluginPath = fs::current_path().string<char>() + ";" + gstPluginPathEnv;
    _putenv_s("GST_PLUGIN_PATH", gstPluginPath.c_str());
} else {
    _putenv_s("GST_PLUGIN_PATH", fs::current_path().string<char>().c_str());
}
```

<Example(Ubuntu)>

```
char* gstPluginPathEnv = getenv("GST_PLUGIN_PATH");
if (gstPluginPathEnv != nullptr) {
    std::string gstPluginPath = fs::current_path().string<char>() + ";" + gstPluginPathEnv;
    setenv("GST_PLUGIN_PATH", gstPluginPath.c_str(), 1);
} else {
    setenv("GST_PLUGIN_PATH", fs::current_path().string<char>().c_str(), 1);
}
```

SONY

■ Initialization

Initialize GStreamer with gst_init().
g_main_loop_new() generates the main loop.

<Example>

```
gst_init(nullptr, nullptr);
GMainLoop *mainLoop = g_main_loop_new(nullptr, FALSE);
```

SONY

■ Pipeline Generation

gst_parse_launch() is used to generate the pipeline.
※ esvideosrc/esaudiosrc is the original plugin module.

<Example(Windows)>

```
char *pipeLineString = “esvideosrc name=esvdsrc ! queue ! h265parse ! d3d11h265dec ! d3d11upload ! d3d11scale ! d3d11download ! video/x-raw, width=576, height=324 ! d3d11videosink name=vdsnk esaudiosrc name=esausrc ! queue ! aacparse ! avdec_aac ! audioconvert ! directsoundsink name=ausnk";
GError *error = NULL;
GstElement *pipeline = gst_parse_launch(pipeLineString, &error);
```

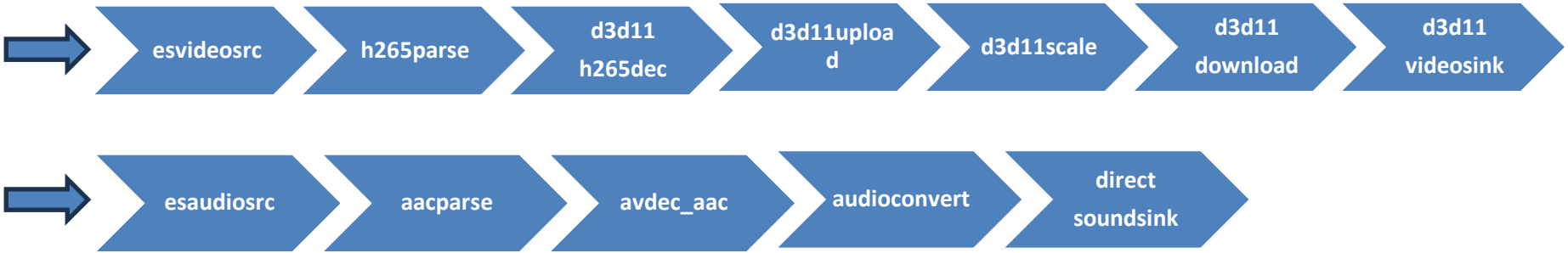
<Example(Ubuntu)>

```
char *pipeLineString = “esvideosrc name=esvdsrc ! queue ! h265parse ! vaapih265dec ! vaapipostproc ! video/x-raw, width=576, height=324, force-aspect-ratio=true ! vaapisink name=vdsnk esaudiosrc name=esausrc ! queue ! aacparse ! avdec_aac ! audioconvert ! autoaudiosink name=ausnk";
GError *error = NULL;
GstElement *pipeline = gst_parse_launch(pipeLineString, &error);
```

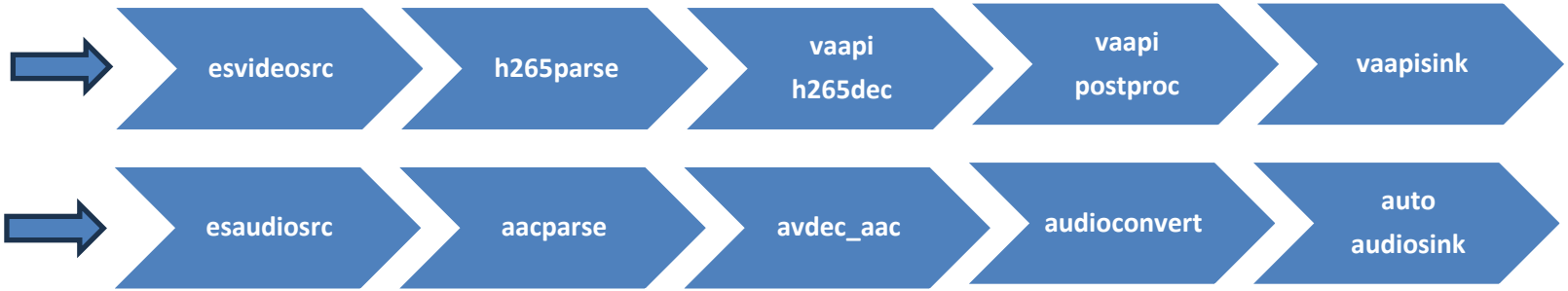
■ Composition of Elements

SONY

Pipeline(Windows)



Pipeline(Ubuntu)



■ Callback function registration

SONY

Obtain a bus from the generated pipeline with gst_element_get_bus().
Register a callback function to the bus with gst_bus_add_watch().
The callback function is registered and executed when EOS, ERROR, etc. are detected by the GStreamer.

<Example>

```
static gboolean bus_callback(GstBus* bus, GstMessage* msg, gpointer data){
    switch (GST_MESSAGE_TYPE(msg)) {
    case GST_MESSAGE_EOS:
        printf("GST_MESSAGE_EOS¥n);
        break;
    case GST_MESSAGE_ERROR:
        printf("GST_MESSAGE_ERROR¥n);
        break;
    default:
        break;
    }
    return true;
}

void register_bus_callback(GstElement *pipeline){
    GstBus* bus = gst_element_get_bus(pipeline);
    unsigned int watchId = gst_bus_add_watch(bus, bus_callback, nullptr);
}
```

SONY

■ Set pipeline status to regenerate

Set the pipeline state to PLAYING with gst_element_set_state().

<Example>

```
GstStateChangeReturn ret = gst_element_set_state(pipeline, GST_STATE_PAUSED);
ret = gst_element_set_state(pipeline, GST_STATE_PLAYING);
```

SONY

■ Execution of main loop

Execute the main loop with g_main_loop_run().
g_main_loop_run() is should be in a separate thread because the function returns at the end of the main loop.

```
<Example>
    std::thread t([]() {
        g_main_loop_run(mainLoop);
    });
```

■ Video Data Reception

SONY

After executing the main loop of GStreamer, the video playback function of RemoteSDK is executed to get video data from the camera.
Store video/audio data in a ring buffer when OnReceivePlaybackData() is executed.
The data stored in the ring buffer is retrieved by esvideosrc/esaudiosrc, and the video is played back by flowing the data to the elements in the latter stage.
In addition, video and audio can be synchronized using PTS.

<Example>

```
    void OnReceivePlaybackData(CrInt8u mediaType, CrInt32 dataSize, CrInt8u* data, CrInt64 pts,
CrInt64 dts, CrInt32 param1, CrInt32 param2) {
        if (mediaType == SDK::CrMoviePlaybackDataType_Video) {
            // Store data in ring buffer
        }else{
            // Store data in ring buffer
        }
    }
```

■ Overall image when video playback preview is executed