Candera Guide to Using GStreamer in CGI Studio

Audience: Developers and integrators working with Candera CGI Studio who want to decode and display video streams (e.g., H.264) via GStreamer inside their applications.

Goal: Install and configure GStreamer, enable streaming in CGI Studio projects, add the VideoStream control, diagnose common issues, and follow best practices for reliable playback.

Overview

CGI Studio integrates a sample implementation for GStreamer to show how to use the `VideoStreamer` interface. The VideoStreamer sample implementation uses GStreamer to decode the specified video source and provides frames to the Video Stream behavior.

This guide walks you through setup, configuration, and common diagnostics to ensure a smooth video playback experience within your CGI Studio projects and shows how to customize or add your own implementation.

Supported Platforms

- **Windows, Android, and Linux** platforms with an installed GStreamer environment are supported.
- **Android** support is limited to **aarch64** targets.
- Default modules support **H.264 Baseline profile**.

Typical Workflow (Quick Start)

1. Install GStreamer and verify with gst-inspect-1.0.
2. Ensure PATH includes GStreamer.
3. Run CMake for host build and enable/set necessary flags for streaming.
4. Build Player for Host (includes SCHost.dll for Scene Composer) and target.
5. Add **Video Stream control** in the Scene Composer Solution and set VideoSource.
6. Enable Playback in the statemachine or (optional) hook behaviors:
   - **Configure and Play Stream** to start playback.
   - **Check Video State** to loop or pause.
7. Debug with GST_DEBUG and pipeline DOT graphs.

Prerequisites

GStreamer Installation & Verification

GStreamer must be installed and available on your system before running CGI Studio projects that use streaming. The GStreamer version provided with the CGI Studio 3rd-party software (3PSW) is **1.22.12**.

Please use the following command to verify the installation.

Bash

```
gst-inspect-1.0 --version
gst-inspect-1.0
```

You should see a list of plugins and elements if GStreamer is set up correctly.

Windows

Add GStreamer's bin directory to your **PATH**:

- To set temporarily for the current session:

  PowerShell

  ```
  $env:Path = "C:\gstreamer\1.0\x86_64\bin;" + $env:Path
  gst-inspect-1.0
  ```

- To make the settings persistent (save to user environment variables):

  PowerShell

  ```
  [Environment]::SetEnvironmentVariable("Path", "C:\gstreamer\1.0\x86_64\bin;" +
  [Environment]::GetEnvironmentVariable("Path","User"), "User")
  ```

Android Evaluation Package

Check **"Install GStreamer"** during setup. Even if already installed, run **Repair** to ensure all dependencies are met. If CGI Player reports missing DLLs, ensure GStreamer's `bin` directory is in the `PATH`, or copy the DLLs next to the executable.

Target (Linux)

On Linux targets with access to a package manager like RaspberryPi OS, STM32 MPU OpenSTLinux Distribution or Torizon the package manager can be used to install GStreamer and it's plugins. To get optimal results and hardware acceleration please refer to the manufacturers manuals for installing GStreamer and it's plugins.

For most platforms Gstreamer and it's plugins should be built during YOCTO or buildroot built. Keep in mind, that specific plugins like aiurdemux may support less features or may be more restrictive but offer hardware acceleration.

For more general information, please visit the official GStreamer website:
https://gstreamer.freedesktop.org/.

How to Build the CGI Player with GStreamer

For the following steps we assume you successfully finished GStreamer Installation & Verification.

- Make sure you have a working Microsoft Visual Studio installation.
- Close Microsoft Visual Studio.
- Install GStreamer for Windows developer and runtime:
  - Go to
    https://gstreamer.freedesktop.org/data/pkg/windows/1.22.12/msvc/gstreamer-1.0-devel-msvc-x86-1.22.12.msi
    If you skipped GStreamer Installation & Verification also install:
    https://gstreamer.freedesktop.org/data/pkg/windows/1.22.12/msvc/gstreamer-1.0-msvc-x86-1.22.12.msi
- If you do not have GStreamer directories in the folder cgi_studio_3psw/lib and cgi_studio_3psw/src you
  need to copy the following files and folders from your GStreamer installation directory:
  1. Copy the following folders from your GStreamer installation directory (C:/gstreamer/1.0/msvc_x86/include) to folder cgi_studio_3psw/src/GStreamer/Win32/include
     - glib-2.0
     - gstreamer-1.0
  2. Copy the following folders from your GStreamer installation folder (C:/gstreamer/1.0/msvc_x86/lib) to folder cgi_studio_3psw/lib/GStreamer/Win32/lib
     - glib-2.0
     - gstreamer-1.0
  3. Copy the following library files from your GStreamer installation folder (C:/gstreamer/1.0/msvc_x86/lib) to folder cgi_studio_3psw/lib/GStreamer/Win32/lib
     - gio-2.0.lib
     - glib-2.0.lib

    - gobjext-2.0.lib
    - gstreamer-1.0.lib

CMake Flags

Set the platform and enable/disable streaming and mixed reality features at build time:

- Open CMake
- Input the location of your source code into the first input field "Where is the source code:" (/cmake/Candera/Player).
- Input the location where to build the binaries to into the input field "Where to build the binaries:"
- To be sure you have a clean build, go to the menu "File - Delete Cache".
- Click the "Configure" button in the middle of the CMake window on the left side.
- A dialog will be opened, choose "Optional platform for generator": Win32
- Wait for CMake to finish configuration.
- In the CMake defines, adjust the following settings:
    - COURIER_PLATFORM
      Selects the paltform to build for. e.g. AndroidPlatform
    - CGIAPP_SAMPLE_STREAMING_ENABLED
      Enables streaming and links GStreamer libraries from the project’s third-party directory.
    - CGIAPP_SAMPLE_MIXEDREALITY_ENABLED
      Enables augmented-reality related behaviors and controls. This is required to run the mixed
      reality sample solution included with Scene Composer which also contains samples for video
      streaming.
    - CGIAPP_SAMPLE_VIDEO_ROOT_PATH Sets the base folder for video files. Make sure this is
      correctly set depending on you making a host or a target build.
- Click the "Configure" button again.
- Wait for CMake to finish configuration.
- Press the "Generate" button (just right to the "Configure" button).
- Wait for the Microsoft Visual Studio Solution to be generated.
- Navigate to the Microsoft Visual Studio Solution file in the folder "Where to build the binaries".
- Open the Solution file "Player.sln" with your working Microsoft Visual Studio.
- Wait for the project to load.
- In the Microsoft Visual Studio menu press "Build - Build Solution".
- Wait for the solution to be built.
- Finally, you can start the Player application.

VideoStream Control

As part of the Augmented Reality Controls (AR Controls) the CGI Studio Player was enhanced with a sample implementation for video Streams. This is used in the MixedReality Sample which is avaliable for selected platforms.

The MixedReality sample has been modified and uses the Video Stream (Video Stream) functionalities.

Furthermore, the intro video is played in a infinite loop instead of once, which is realized with a condition behavior (Check Video State) and action behavior (Configure and Play Stream).

Properties

Property	Type	Description
TargetNode	Candera::AbstractNodePointer	Node where the video will be displayed. If empty, the anchor node is used.
VideoSource	FeatStd::String	URI to the video (local path, RTSP, or asset).
ScaleToVideoSource	bool	If enabled, resizes texture to match video dimensions.
FlipVertically	bool	Flips video vertically (2D only).
Timestamp	FeatStd::UInt32	Bind video timestamp to external reference for sync.

Using Video Stream in CGI Studio

Add the Control

1. Open your scene in CGI Studio. This can be a 2D or 3D scene. The process is similar for both.
2. Add the **VideoStream** control to the scene.
3. Assign the target node (mesh/rectangle where the video texture should appear).
4. Add a video source. The video source property is a string. Opening the property editor for the video source does NOT take you to the "Choose Item" dialog. The language table selection opens instead where you can create strings for this property based on the selected locale. The tooltip says to provide the URI of the video. This is done as follows:
   - Video stream from raw resource in the asset: `asset://SampleKit_MixedReality##Resources#Videos#SampleVideo_Intro.mp4` which is the "Candera Name" property of the video resource prefixed with `asset://`
   - Video stream from RTSP: Use the URL e.g. `http://www.cgistudio.at/mixedreality_sample.mp4`
   - Video stream from file: As video source string provide just the file name. The base folder e.g. `../../cgi_studio_content/html/video/` is set in the CMake configuration and therefore hardcoded in the application. Make sure to set the

CGIAPP_SAMPLE_VIDEO_ROOT_PATH correctly for both host builds and target builds.

5. Set an image that is shown if the stream cannot be decoded.
   - For a 2D scene the image is set on the brush assigned to the RenderNode of the StreamTarget.
   - For a 3D scene the image is set as texture of the Appearance assigned to the Mesh that is set as StreamTarget.
6. Now choose a method to start playback from following:
   - State Machine - Open the state machine and add a new OnEntry on the state of your choice. Set
     the Action to "Configure and Play Stream" and set the VideoControl in your Scene as TargetNode.
   - Action and Condition Behaviours
     - Action: Configure and Play Stream
     - Condition: Check Video State

See Candera AR Controls documentation:

cgi_studio_docs\offlineDoc Page: Augmented Reality Controls

The VideoStreamer Interface

VideoStreamer provides an Interface towards the VideoSteamBehavior. The header can be found here:
/cgi_studio_controls/src/Behaviors/Streaming/VideoStreamer.h

GStreamer Sample Implementation

The Player contains a non optimized sample implementation of the VideoStreamer for GStreamer. The implementation targets to be close to the GStreamer samples and easy to understand. There are no platform specific implementations like specialized pipelines for the iMX 8 series.

The video source string is analyzed and according to the protocol the right source implementation of gstreamer chosen. For video files from assetit is giostreamsrc, for udp it's udpsrc, uridecodebin for http and rtsp and for files uridecodebin is used after concatenating the CGIAPP_SAMPLE_VIDEO_ROOT_PATH set in CMake with the file name and converting the file name to an URI.

For Android all the GStreamer plugins are treated as static plugins and must be added manually. This is
already included in the sample implementation. As for now only a prebuilt GStreamer shared library for
Android on aarch64 architectures is in the 3psw folder. For all other architectures the customer must build this
shared libray.

Debugging GStreamer

GStreamer supports setting the loglevel over environment variables or in code.

Enable GStreamer Logs (GST_DEBUG)

Setting GST_DEBUG to a value between 1 and 9 enables debug output.

`export GST_DEBUG=<level>` or `GST_DEBUG=<level> ./Player AssetFileCff.bin`

Level	Name	Description
0	none	No debug information is output.
1	ERROR	Logs all fatal errors. These are errors that do not allow the core or elements to perform the requested action. The application can still recover if programmed to handle the conditions that triggered the error.
2	WARNING	Logs all warnings. Typically these are non-fatal, but user-visible problems are expected to happen.
3	FIXME	Logs all "fixme" messages. Those typically show that a codepath that is known to be incomplete has been triggered. It may work in most cases, but may cause problems in specific instances.
4	INFO	Logs all informational messages. These are typically used for events in the system that only happen once, or are important and rare enough to be logged at this level.
5	DEBUG	Logs all debug messages. These are general debug messages for events that happen only a limited number of times during anobject's lifetime; these include setup, teardown, change of parameters etc.
6	LOG	Logs all log messages. These are messages for events that happen repeatedly during an object's lifetime; these includestreaming and steady-state conditions. This is used for logmessages that happen on every buffer in an element for example.

7	TRACE	Logs all trace messages. Those are message that happen very very often. This is for example is each time the reference count of a GstMiniObject, such as a GstBuffer or GstEvent, ismodified.
9	MEMDUMP	Logs all memory dump messages. This is the heaviest logging and may include dumping the content of blocks of memory.

See GStreamer Basic tutorial 11, 2025 for details.

Dumping Pipeline Graphs

Sometimes a more visual representation is very helpful to understand what is going on. GStreamer offers built
in support for that.

1. Initial Configuration
   Add the following code to your initialization process.
   (e.g., within bool InitImpl(...) in
   /cgi_studio_player/src/Player/Features/Streamer/GStreamer.cpp)

   C++

   ```
   static bool init = false;
   if (!init) {
       init = true;
       g_setenv("GST_DEBUG_DUMP_DOT_DIR", "/tmp", TRUE);
   }
   ```

   Alternatively, set the environment variable before launching the application:
   export GST_DEBUG_DUMP_DOT_DIR=/tmp

2. Executing the Dump
   Add the following code at the point where you want to capture the pipeline data.
   (e.g., within bool StopImpl() in
   /cgi_studio_player/src/Player/Features/Streamer/GStreamer.cpp)

   C++

```
gst_debug_bin_to_dot_file(GST_BIN(m_pipeline), GST_DEBUG_GRAPH_SHOW_ALL,
"gstreamerpipe");
```

3. Viewing the File
   After dumping the .dot file, you can use Graphviz to view the file or convert it to SVG format.

   Bash

   ```
   dot -Tsvg /tmp/gstreamerpipe.dot -o /tmp/gstreamerpipe.svg
   ```

Graph Structure and Legend

Examples of Pipeline Elements

- GstPipeline (gstreamer-pipeline)
- GstAppSink (app_sink): Final data output destination
- GstVideoConvert (video_convert): Format conversion
- GstQueue (app_queue): Data buffering
- GstVideoFlip (video_flip): Video flipping and rotation
- GstDecodeBin (decoder): Automatic decoding process
- GstAiurDemux / GstGioStreamSrc: Source input and demuxer

Legend

Element States

- [~] void-pending: Transition pending
- [0] null: Initial state
- [-] ready: Ready
- [=] paused: Paused
- [>] playing: Playing

Pad Activation

- [-] none: None
- [>] push: Push mode
- [<] pull: Pull mode

Pad Flags

- [b] locked: Locked
- [f] flushing: Flushing
- [b] blocking: Blocking
- [E] OS (EOS): End of Stream

> Uppercase letters indicate that the flag is set.

Pad Task

- [T] Task started
- [t] Task paused

Troubleshooting & Best Practices

- DLL Missing: Add GStreamer to PATH or copy DLLs.
- Android support is limited to aarch64 targets.
- iMX8 aiurdemux causes problems with some video files when using giostreamsrc.
  - The same videos work fine if filesrc is used.
  - Use less aggressive compression, try other framerates or compression formats.
  - See the iMX8 GStreamer User Guide for details on the iMX8 series. Try using fragmented MP4 for progressive playback which does not require seek.
- Selecting the decoder, mux etc. is done by GStreamer to keep the sample code understandable.
  - If you want to use a specific pipeline please adapt /cgi_studio_player/src/Player/Features/Streamer/GStreamer.cpp for your needs or use your own implementation.

Best Practices & Performance Tips

- Use streams compatible with H.264 Baseline.
- Start with moderate resolutions.
- For optimal performance platform specific implementations are required.
  - Leverage hardware acceleration: G2D compositor, videoconvert, VP DSP. Most SoCs of the iMX 6 and 8 series have specialized hardware for decoding, color format conversion, scaling etc.

Version Compatibility

- GStreamer 1.22.12

References:

- GStreamer Official Site
- Windows Installation Guide

- Debugging Tools Tutorial
- Augmented Reality Controls

Revision #8
Created 2026-03-27 06:00:39 UTC by Kanai Tomoaki
Updated 2026-09-01 08:37:10 UTC by Kanai Tomoaki