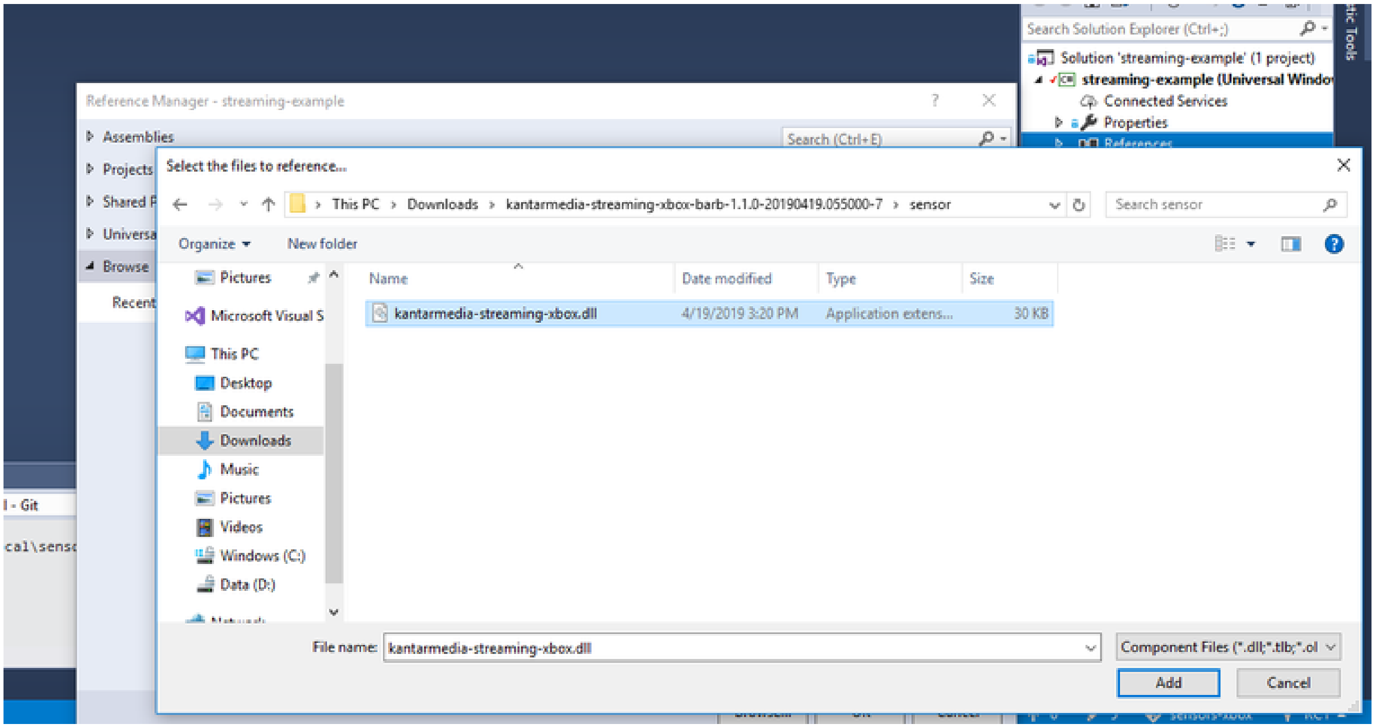
streaming-xbox/Implementation

Streaming Measurement in streaming-xbox

This chapter describes the basic use of the sensor for the measurement of streaming content.

- Right-Click on "References"
- Click on "Add Reference"
- Click on "Browse..." and open "**kantarmedia-streaming-xbox.dll**"



This PC › Downloads › kantarmedia-streaming-xbox-barb-1.1.0-20190419.055000-7 › sensor

Name	Date modified	Type	Size
kantarmedia-streaming-xbox.dll	4/19/2019 3:20 PM	Application extens...	30 KB

File name: kantarmedia-streaming-xbox.dll

Component Files (*.dll;*.tlb;*.ol

Lifecycle of the Measurement

This chapter explains the use of the sensor for measuring the streaming content . The following are necessary things to know when using the sensor.

1. Create instance of sensor
2. Implementation of the adapter
3. Beginning of the measurement
4. End of the measurement

Special Cases

Before getting into the measurement , Below are certain cases/events, which will pause/stop measurement.

Stopping of all streams

- the app is closed
- the app is not visible (minimized)

This will stop all measurement and finalize all running streams.

Stopping of certain stream

- the window containing the media which is played is minimized (send to background)
- the window containing the media which is played is turned invisible

This will only stop the stream in the concerning window, other streams are not affected.

If any streams that are stopped due to special events mentioned above and should be measured again, the **Track** method has to be called again

Creating Instance of Sensor

When you start the app the sensor can only be instantiated once. It is **not possible** to change the site name or the application name during the measurement.

```
String site = ...;    // will delivered by the measurement provider
KMStreamingSensor<IStreamAdapter> sensor =
KMStreamingSensor<IStreamAdapter>.getInstance(site);
```

The site name is specified by the operator of the measurement system.

Implementation of the Adapter

In principle it is possible to measure any media library with an adapter, that is available within an app to use streaming content .

Therefore the protocol `Meta` and the interface `IStreamAdapter` must be implemented. The meta object supplies the information regarding the player(pl,plv) and the screen(sx,sy).

The library must be able to read continuously the current position on a stream/player in seconds.

```
//Implement Imeta && IStreamAdapter interfaces


 class ExampleAdapter : IStreamAdapater
    {
        class TestMeta : IMeta
        {
            public string getPlayerName()
            {
                return "ThePlayer";
            }

            public string getPlayerVersion()
            {
                return "1.0";
            }
            public int getScreenHeight()
            {
                return getScreenHeightAsyncTask().Result;
            }

            public async Task<int> getScreenHeightAsyncTask()
            {
                int returnValue = 0;
                await ThreadHelper.ExecuteOnUIThread(() =>
                {
                    returnValue =
unchecked((int)DisplayInformation.GetForCurrentView().ScreenHeightInRawP
ixels);
                });

                return returnValue;
            }

            public int getScreenWidth()
            {
                return getScreenWidthAsyncTask().Result;
            }

            public async Task<int> getScreenWidthAsyncTask()
            {
                int returnValue = 0;
                await ThreadHelper.ExecuteOnUIThread(() =>
                {
                    returnValue =
unchecked((int)DisplayInformation.GetForCurrentView().ScreenWidthInRawPi
xels);
                });

                return returnValue;
            }
        }
        MediaPlayer mediaPlayer = null;
        IMeta meta = new TestMeta();

        public ExampleAdapter(MediaPlayer player)
        {
            mediaPlayer = player;
        }

        public int getDuration()
        {
            return
unchecked((int)mediaPlayer.PlaybackSession.NaturalDuration.TotalSeconds)
;
        }
        public int getHeight()
        {

            return
unchecked((int)mediaPlayer.PlaybackSession.NaturalVideoHeight);
        }

        public IMeta getMeta()
        {
            return meta;
        }

        public int getPosition()
        {
            int position =
unchecked((int)mediaPlayer.PlaybackSession.Position.TotalSeconds);
            return position;
        }

        public int getWidth()
        {
            return
unchecked((int)mediaPlayer.PlaybackSession.NaturalVideoWidth);
```

```
        }
    }
```

```
//Create stream adapter by implementing IStreamAdapter
var playerData = new ExampleAdapter(mediaElement.MediaPlayer);
```

Beginning of the Measurement

Create a Dictionary<String> to provide information of stream to be measured .

Below are the values that can be added inside dictionary

stream(mandatory),cq,ct,sx,sy

```
Dictionary<String, Object> dict = new Dictionary<string, object>();
dict.Add("stream","mystream");
dict.Add("cq", "4711"); //Add optional properties
```

Stream is mandatory to start the measurement.

Call track method

```
//Start tracking the stream
streams.Track(playerData, dict);
```

This finalizes the Stream and should be called any time the video is interrupted (i.e. application goes to background) or the video is finished.

```
//when finished stop the adapter
streams.Unload();
```

The above code in one block:

```
KMStreamingSensor<IStreamAdapter> streams =
KMStreamingSensor<IStreamAdapter>.getInstance("test")
mediaElement.Source = MediaSource.CreateFromUri(anUri);
//Create stream adapter
var playerData = new ExampleAdapter(mediaElement.MediaPlayer);
Dictionary<String, Object> dict = new Dictionary<string, object>();
//Add optional properties
dict.Add("cq", aName);
//Start tracking the stream
streams.Track(playerData, dict);
```

For a full example see streaming-example app

End of the Measurement

Call the unload method whenever app is closed. This call sends the current state of the measurements to the measuring system and then terminates all measurements.

```
KMStreamingSensor<IStreamAdapter> streams =
KMStreamingSensor<IStreamAdapter>.getInstance("test")
...
streamingSensor.Unload();
```

stop()

measurement of streams is stopped automatically by library in the special cases mentioned above . If the user needs to stop/pause the measurement in any other case, use the stop method on stream object

```
// start streaming measurement
var stream = streamingSensor.Track(playerData, dict);
...
// stop measurement programmatically
stream.Stop();
```

If the stream should be measured again after the method stop has been called, the method track must be called again.