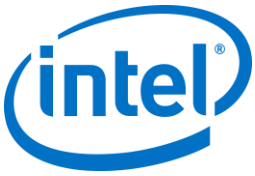


Intel® SensorViewer

Manual

March, 2018

Contents

Tool Description

The Intel® SensorViewer is a tool that enables you to view and monitor a list of available sensor and their properties, edit property values, receive sensor data and log them to CSV file. Intel® SensorViewer works with "Desktop API", "Windows RT API" and "Windows RT for WIN10 API". SensorViewer can be used by GUI or CLI.

Tool Requirements

To run the Intel® SensorViewer, you need:

- The .NET framework 4.5.1 or higher (can be downloaded from here)
- Windows 7 or higher

Tool Configuration

Application Configuration

Some settings are determined upon application startup. These settings can be edited in SensorViewer.exe.config. They cannot be changed during runtime:

- *LoggingTimestampType* – The type(s) of timestamps to log. Accepted values are:
  - *Received* (default) – date and time that the record was sent from the sensor, as reported by the sensor.
  - *System* – date and time that the record was received by the application.
  - *ReceivedAndSystem* or *All* – Both of the above timestamps will be logged.
- *LogRootPath* – The path in which the *LogRecordName* directory will be saved. Defaults to the current directory.
- *LogRecordName* – The name of the directory to create at *LogRootPath*. This is the directory in which the logs will be saved. Defaults to 'log'.
- *IsLogFilenamesWithTimestamp* – Indicates if the date and time at which logging began should be appended to the name of the directory indicated by *LogRecordName*. Defaults to false.
- *IsConsoleLogWithSensorEvents* – An Option to support printing sensor events log to the console. If it's set to 'true', then the log will be printed; otherwise, the log is disabled. The default value is 'false'.
- *SupportCustomProperties* – An option to enable/disable the custom properties for WinRT standard sensor.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0"
            sku=".NETFramework,Version=v4.5.1" />
    </startup>
  <appSettings>
    <add key="LoggingTimeStampType" value="Received"/>
    <add key="IsLogFilenamesWithTimestamp" value="true"/>
    <add key="LogRecordName" value="log"/>
    <add key="LogRootPath" value="C:\SensorViewer"/>
    <add key="IsConsoleLogWithSensorEvents" value="false"/>
```

```
    <add key="SupportCustomProperties" value="false"/>
    </appSettings>
</configuration>
```

With the above settings, csv files whose logging began on 28th January 2016 at 1:38 PM will be stored in **C:\SensorViewer\log_28_01_16_13_38**.

	A	B	C
1	TimeStamp-Received	ALS.Level{0} [lux]	ALS.Timestamp received{0}
2	09-February-2016 09-26-23.811000	0.6	09-February-2016 09-26-23.811861
3	09-February-2016 09-26-23.811000	0.6	09-February-2016 09-26-23.811861
4	09-February-2016 09-26-23.811000	0.6	09-February-2016 09-26-23.811861
5	09-February-2016 09-26-23.811000	0.6	09-February-2016 09-26-23.811861
6	09-February-2016 09-26-23.811000	0.6	09-February-2016 09-26-23.811861
7	09-February-2016 09-26-23.811000	0.6	09-February-2016 09-26-23.811861
8	09-February-2016 09-26-23.811000	0.6	09-February-2016 09-26-23.811861
9	09-February-2016 09-26-23.811000	0.6	09-February-2016 09-26-23.811861
10	09-February-2016 09-26-23.811000	0.6	09-February-2016 09-26-23.811861
11	09-February-2016 09-26-23.811000	0.6	09-February-2016 09-26-23.811861
12	09-February-2016 09-26-33.797000	233.5	09-February-2016 09-26-33.797714
13	09-February-2016 09-26-33.797000	233.5	09-February-2016 09-26-33.797714
14	09-February-2016 09-26-33.797000	233.5	09-February-2016 09-26-33.797714
15			

Picture 1: Screenshot of CSV file with records for an Ambient Light Sensor with the LoggingTimestampType option set to Received.

	A	B	C
1	TimeStamp-System	ALS.Level{0} [lux]	ALS.Timestamp received{0}
2	09-February-2016 09-42-41.598334	460.4	09-February-2016 09-42-33.315849
3	09-February-2016 09-42-42.148493	460.4	09-February-2016 09-42-33.315849
4	09-February-2016 09-42-42.645650	460.4	09-February-2016 09-42-33.315849
5	09-February-2016 09-42-43.146649	460.4	09-February-2016 09-42-33.315849
6	09-February-2016 09-42-43.659660	460.4	09-February-2016 09-42-33.315849
7	09-February-2016 09-42-44.159168	460.4	09-February-2016 09-42-33.315849
8	09-February-2016 09-42-44.665478	460.4	09-February-2016 09-42-33.315849
9	09-February-2016 09-42-45.175870	460.4	09-February-2016 09-42-33.315849
10	09-February-2016 09-42-45.675912	460.4	09-February-2016 09-42-33.315849
11	09-February-2016 09-42-46.181336	0.9	09-February-2016 09-42-45.848434
12	09-February-2016 09-42-46.680264	0.9	09-February-2016 09-42-45.848434
13	09-February-2016 09-42-47.192999	0.9	09-February-2016 09-42-45.848434
14	09-February-2016 09-42-47.706569	0.9	09-February-2016 09-42-45.848434
15	09-February-2016 09-42-48.213442	0.9	09-February-2016 09-42-45.848434

Picture 2: Screenshot of a CSV file with records for an Ambient Light Sensor with the LoggingTimestampType option set to System.

	A	B	C	D
1	TimeStamp-Received	TimeStamp-System	ALS.Level{0} [lux]	ALS.Timestamp received{0}
2	09-February-2016 09-30-58.044000	09-February-2016 09-31-03.949458	285	09-February-2016 09-30-58.044727
3	09-February-2016 09-30-58.044000	09-February-2016 09-31-04.449594	285	09-February-2016 09-30-58.044727
4	09-February-2016 09-30-58.044000	09-February-2016 09-31-04.951055	285	09-February-2016 09-30-58.044727
5	09-February-2016 09-31-05.370000	09-February-2016 09-31-05.464370	3.1	09-February-2016 09-31-05.370740
6	09-February-2016 09-31-05.370000	09-February-2016 09-31-05.976450	3.1	09-February-2016 09-31-05.370740
7				

Picture 3: Screenshot of a CSV file with records for an Ambient Light Sensor with the LoggingTimestampType option set to All.

Application State

It is possible to save the current state of the application to an XML file. This file will store all of the setting and sensor configurations that can be edited in the GUI. The state can then be loaded from that file using the GUI or CLI.

By default, the GUI tries to load the application's state from SensorViewer-Default-config.xml (in the same directory as SensorViewer.exe). If the file does not exist or is corrupted, the default state will be loaded instead. This It is possible to set up the following XML elements:

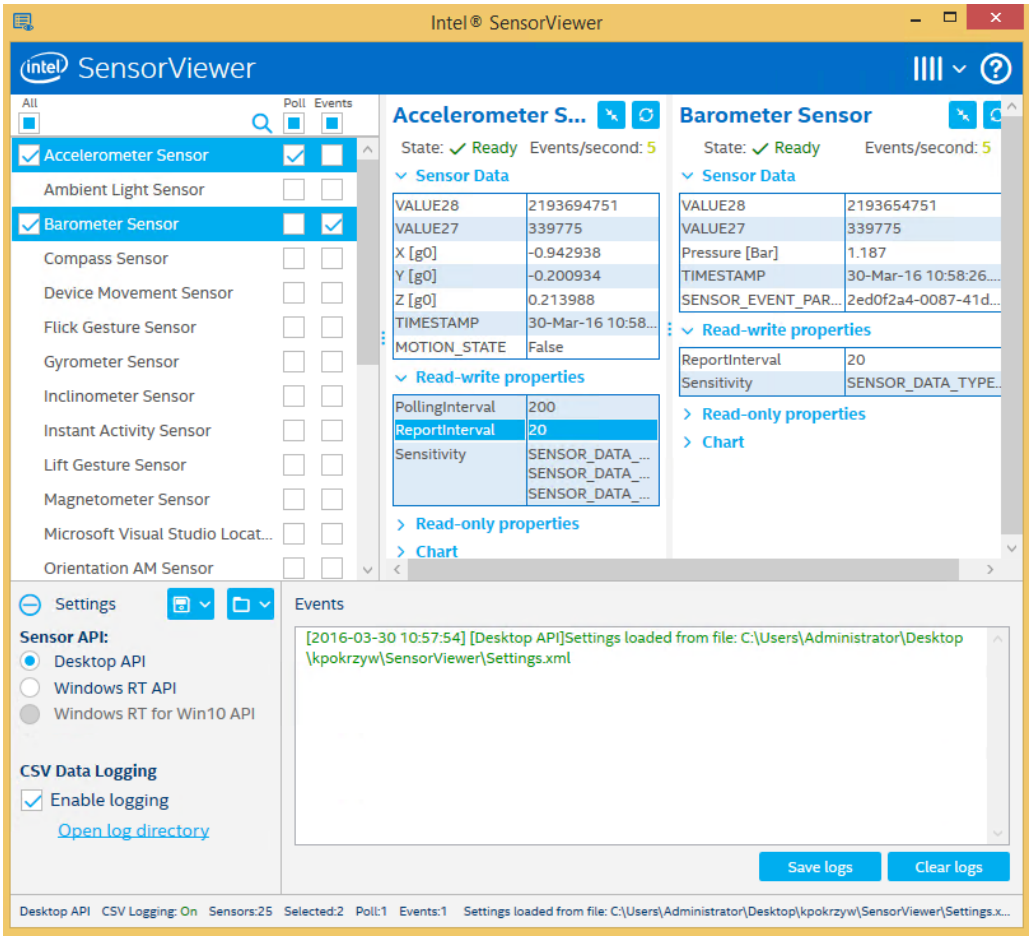
- **ApiType** – The type of sensor API to use. Accepted values are:
  - *Windows7* – The Desktop API. Uses the Sensor API Microsoft Windows assembly.
  - *Windows8* – The Windows RT API. Uses the Windows.Devices.Sensors Microsoft Windows assembly in Windows 8.x.
  - *Windows10* – The Windows RT for Win10 API. Uses the Windows.Devices.Sensors Microsoft Windows assembly in Windows 10.
- **LoggingConfig**
  - *IsLoggingEnabled* – Indicates whether to log the actual sensor data to a csv file. Accepted values are *true* and *false*.
- **Sensors** – Configures the read/write properties and settings of all sensors. Each sensor should be given its own **SensorConfig** element, which contains the following child elements:
  - *Unique Id* – A unique ID that is used by the application to identify the sensor. Do not change this value.
  - *Name* – The sensor's friendly name, as received from the sensor's firmware.
  - *IsSelected* – Indicates if the sensor is currently selected on the Sensor List panel, and is therefore visible in the Sensor Details panel. Accepted values are *true* or *false*.
  - *IsPolling* – Indicates if the sensor should be polled for data immediately after the configuration is loaded. Accepted values are *true* or *false.*
  - *IsStreaming* – Indicates if the sensor is expected to be streaming data. Accepted values are *true* or *false***.**
  - *DisabledAxes* – Axis which were hidden by user in GUI on grid view.
  - *LUID* – locally unique identifier of sensor. Some sensors don't have this value.
  - **Properties** – All editable properties of the sensor. If you are manually editing the XML file, change only the value found in "Value" elements. Do not add properties or change property names or types, or the configuration will not load properly. Accepted properties include:
    - *PollingInterval* – In polling mode, the application periodically contacts the sensor to read data from it. This setting determines the amount of time, in milliseconds, between each such poll if polling mode is enabled.
    - *ReportInterval* – When not in polling mode, the sensor streams data to the application on its own. This setting indicates the amount of time, in milliseconds, between data samples received from the sensor.

- *Sensitivity* – Only available in the Desktop API. This setting informs the sensor of the minimum difference between two data samples that is considered significant enough to report to the application. More information is available under Sensor API.
- *ReportLatency*, *ReadingTransform*, and *ReadingType* – Only available in the Windows RT for Win10 API. See Windows.Devices.Sensors for details.

File 1: Sample configuration file with application state for the Desktop API:

```
<?xml version="1.0"?>
<MainConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <ApiType>Windows7</ApiType>
   <Sensors>
      <SensorConfig>
         <UniqueId>3157509611</UniqueId>
         <Name>Barometer Sensor</Name>
         <IsSelected>true</IsSelected>
         <IsPolling>false</IsPolling>
         <IsStreaming>true</IsStreaming>
         <Properties>
            <PrimitivePropertyConfigOfUInt32>
               <PropertyType>PollingInterval</PropertyType>
               <Value>500</Value>
            </PrimitivePropertyConfigOfUInt32>
            <PrimitivePropertyConfigOfUInt32>
               <PropertyType>ReportInterval</PropertyType>
               <Value>20</Value>
            </PrimitivePropertyConfigOfUInt32>
            <SensitivityConfig>
               <PropertyType>Sensitivity</PropertyType>
               <SingleSensitivities>
                  <SingleSensitivityConfig>
                     <PropertyType>None</PropertyType>
                     <Name>SENSOR_DATA_TYPE_ATMOSPHERIC_PRESSURE_BAR</Name>
                     <Value>0</Value>
                  </SingleSensitivityConfig>
               </SingleSensitivities>
            </SensitivityConfig>
         </Properties>
      </SensorConfig>
      <SensorConfig>
         <UniqueId>1443639337</UniqueId>
         <Name>Accelerometer Sensor</Name>
         <IsSelected>true</IsSelected>
         <IsPolling>true</IsPolling>
         <IsStreaming>false</IsStreaming>
         <Properties>
            <PrimitivePropertyConfigOfUInt32>
               <PropertyType>PollingInterval</PropertyType>
               <Value>200</Value>
            </PrimitivePropertyConfigOfUInt32>
            <PrimitivePropertyConfigOfUInt32>
               <PropertyType>ReportInterval</PropertyType>
               <Value>20</Value>
            </PrimitivePropertyConfigOfUInt32>
            <SensitivityConfig>
               <PropertyType>Sensitivity</PropertyType>
               <SingleSensitivities>
                  <SingleSensitivityConfig>
                     <PropertyType>None</PropertyType>
                     <Name>SENSOR_DATA_TYPE_ACCELERATION_X_G</Name>
                     <Value>0</Value>
                  </SingleSensitivityConfig>
                  <SingleSensitivityConfig>
```

```
                  <PropertyType>None</PropertyType>
                  <Name>SENSOR_DATA_TYPE_ACCELERATION_Y_G</Name>
                  <Value>0</Value>
               </SingleSensitivityConfig>
               <SingleSensitivityConfig>
                  <PropertyType>None</PropertyType>
                  <Name>SENSOR_DATA_TYPE_ACCELERATION_Z_G</Name>
                  <Value>0</Value>
               </SingleSensitivityConfig>
            </SingleSensitivities>
         </SensitivityConfig>
      </Properties>
   </SensorConfig>
</Sensors>
<LoggingConfig>
   <IsLoggingEnabled>true</IsLoggingEnabled>
</LoggingConfig>
</MainConfig>
```

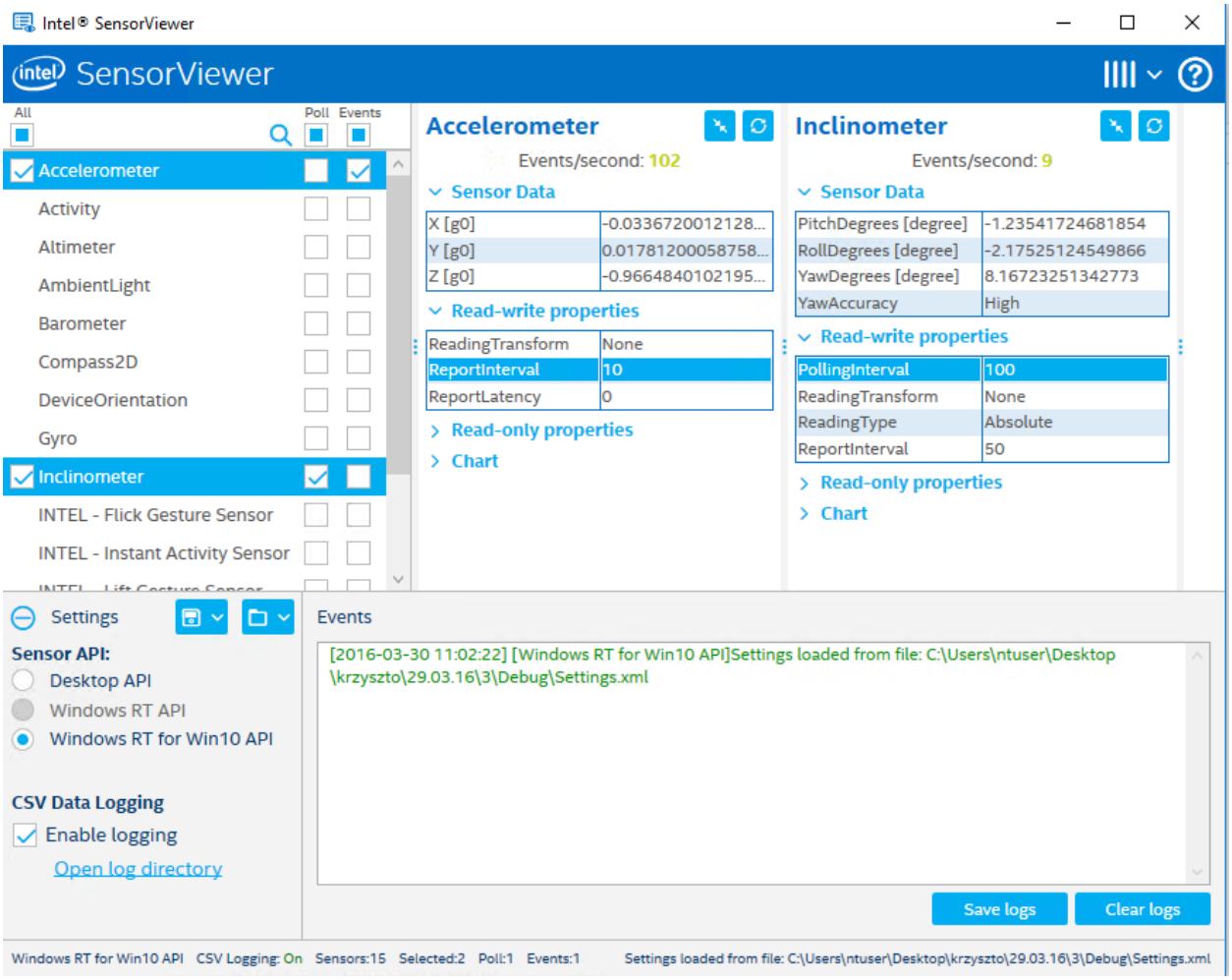


Picture 4: The SensorViewer UI as it appears when the File 1 configuration is loaded.

File 2: Sample configuration file with application state for the Windows RT for Win10 API:

```
<?xml version="1.0"?>
<MainConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <ApiType>Windows10</ApiType>
```

```
   <Sensors>
         <SensorConfig>
               <UniqueId>206862</UniqueId>
               <Name>Accelerometer</Name>
               <IsSelected>true</IsSelected>
               <IsPolling>false</IsPolling>
               <IsStreaming>true</IsStreaming>
               <Properties>
               <PrimitivePropertyConfigOfUInt32>
                     <PropertyType>PollingInterval</PropertyType>
                     <Value>500</Value>
               </PrimitivePropertyConfigOfUInt32>
               <PrimitivePropertyConfigOfString>
                     <PropertyType>ReadingTransform</PropertyType>
                     <Value>None</Value>
               </PrimitivePropertyConfigOfString>
               <PrimitivePropertyConfigOfUInt32>
                  <PropertyType>ReportInterval</PropertyType>
                     <Value>10</Value>
               </PrimitivePropertyConfigOfUInt32>
               <PrimitivePropertyConfigOfUInt32>
                     <PropertyType>ReportLatency</PropertyType>
                     <Value>0</Value>
               </PrimitivePropertyConfigOfUInt32>
               </Properties>
         </SensorConfig>
         <SensorConfig>
               <UniqueId>207046</UniqueId>
               <Name>Inclinometer</Name>
               <IsSelected>true</IsSelected>
               <IsPolling>true</IsPolling>
               <IsStreaming>false</IsStreaming>
               <Properties>
               <PrimitivePropertyConfigOfUInt32>
                  <PropertyType>PollingInterval</PropertyType>
                     <Value>100</Value>
               </PrimitivePropertyConfigOfUInt32>
               <PrimitivePropertyConfigOfString>
                  <PropertyType>ReadingTransform</PropertyType>
                     <Value>None</Value>
               </PrimitivePropertyConfigOfString>
               <PrimitivePropertyConfigOfString>
                     <PropertyType>ReadingType</PropertyType>
                     <Value>Absolute</Value>
               </PrimitivePropertyConfigOfString>
               <PrimitivePropertyConfigOfUInt32>
                     <PropertyType>ReportInterval</PropertyType>
                     <Value>50</Value>
               </PrimitivePropertyConfigOfUInt32>
               </Properties>
         </SensorConfig>
   </Sensors>
   <LoggingConfig>
         <IsLoggingEnabled>true</IsLoggingEnabled>
   </LoggingConfig>
</MainConfig>
```



Picture 5: The SensorViewer UI as it appears when the File 2 configuration is loaded.

Custom Sensor Configuration

The SensorViewer enables you to edit the labels given to the values found in your sensors' data reports. This allows you, for instance, to give unique values to each of the axes in an accelerometer that may be more descriptive than 'x', 'y', and 'z'. You can do this by editing the *CustomSensorConfig.xml* file. By default, it contains a basic collection of sensor definitions that affect the parsing of sensor data.

The file will look something like this:

```
<sensor name="Physical Accel" id="0x73" osType="Windows">
   <fields>
      <field friendly_name="X" field_raw_name="Value1" units="g" />
   </fields>
</sensor>
<sensor name="AmbientLight" osType="Windows">
   <fields>
      <field friendly_name="Level" field_raw_name="LIGHT_LEVEL_LUX" units="lux" />
```

```
    </fields>
  </sensor>
```

Each sensor type has the following attributes:

- **Name** – A friendly name indicating the sensor type. Can be used to identify a sensor type if the ID number below is not available.

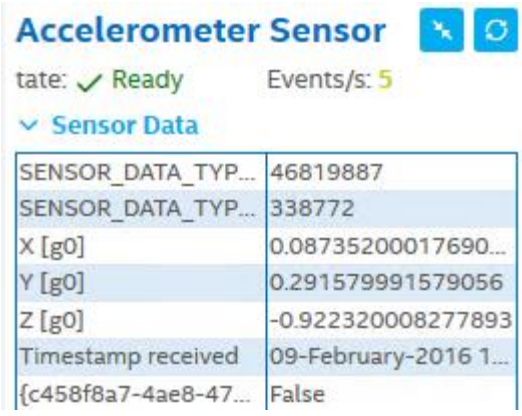
- **Id** – An ID number in hexadecimal that uniquely identifies the sensor type. This can also be used to categorize custom sensors that are not otherwise recognized. In the example above, for instance, every custom sensor with an ID number of 0x73 will have the indicated settings applied.
- **osType** – The OS type. This should always be *Windows*.

To change a label, add a field under the sensor. Each field will change the name of a single data point appearing in the sensor's data report. The field should have the following attributes:

- **field_raw_name** – The name of the field as the sensor originally reports it. This is how the field is identified, so its name can be changed; if a sensor data report contains this a field of this name, it will be displayed instead as the value listed in the friendly_name attribute, and the units (if any) will be displayed.
- **friendly_name** – The custom label that you want to be displayed on this field in the GUI and in CSV files.
- **units** – (optional) The units used by the field. Some data points in a sensor's data report do not have units.

The following example shows how to change the names of each of the axes reported by an accelerometer. The result is visible in Picture 6:

```
<sensor name="Accelerometer" id="0x79" osType="Windows">
  <fields>
    <field friendly_name="X" field_raw_name="Value1" units="g0" />
    <field friendly_name="Y" field_raw_name="Value2" units="g0" />
    <field friendly_name="Z" field_raw_name="Value3" units="g0" />
    <field friendly_name="X" field_raw_name="ACCELERATION_X_G" units="g0" />
    <field friendly_name="Y" field_raw_name="ACCELERATION_Y_G" units="g0" />
    <field friendly_name="Z" field_raw_name="ACCELERATION_Z_G" units="g0" />
  </fields>
</sensor>
```



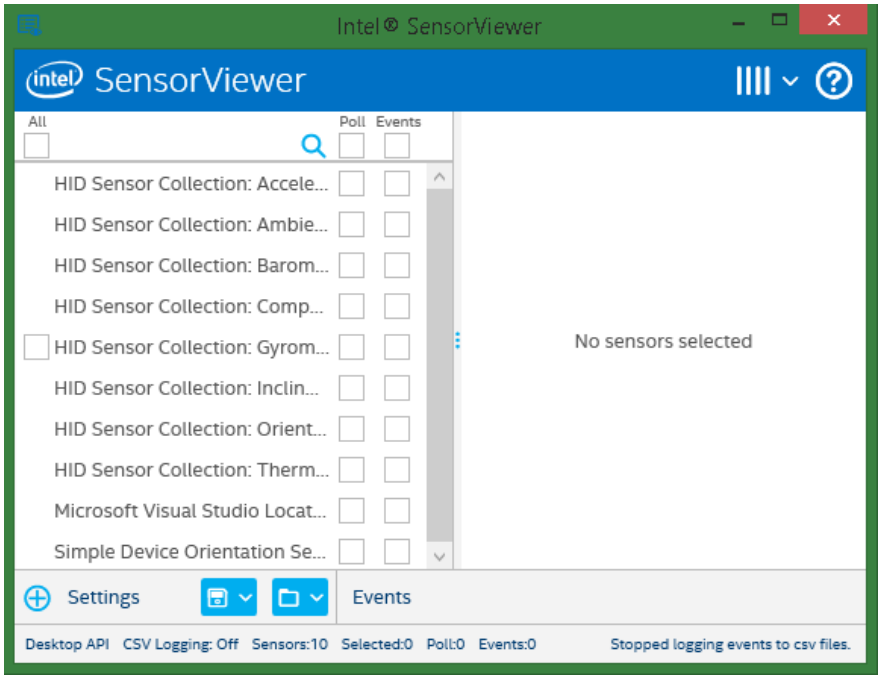
Picture 6: Sensor data received from an accelerometer, with customized field names.

GUI Manual

Run the SensorViewer GUI application by opening **SensorViewer.exe.**

The application will try to load the default configuration file (*SensorViewer-Default-config.xml*), and load the default API if that file is not present or otherwise invalid. The default API is the Windows RT for Win10 API for Windows 10, or the Desktop API for any other version of Windows.

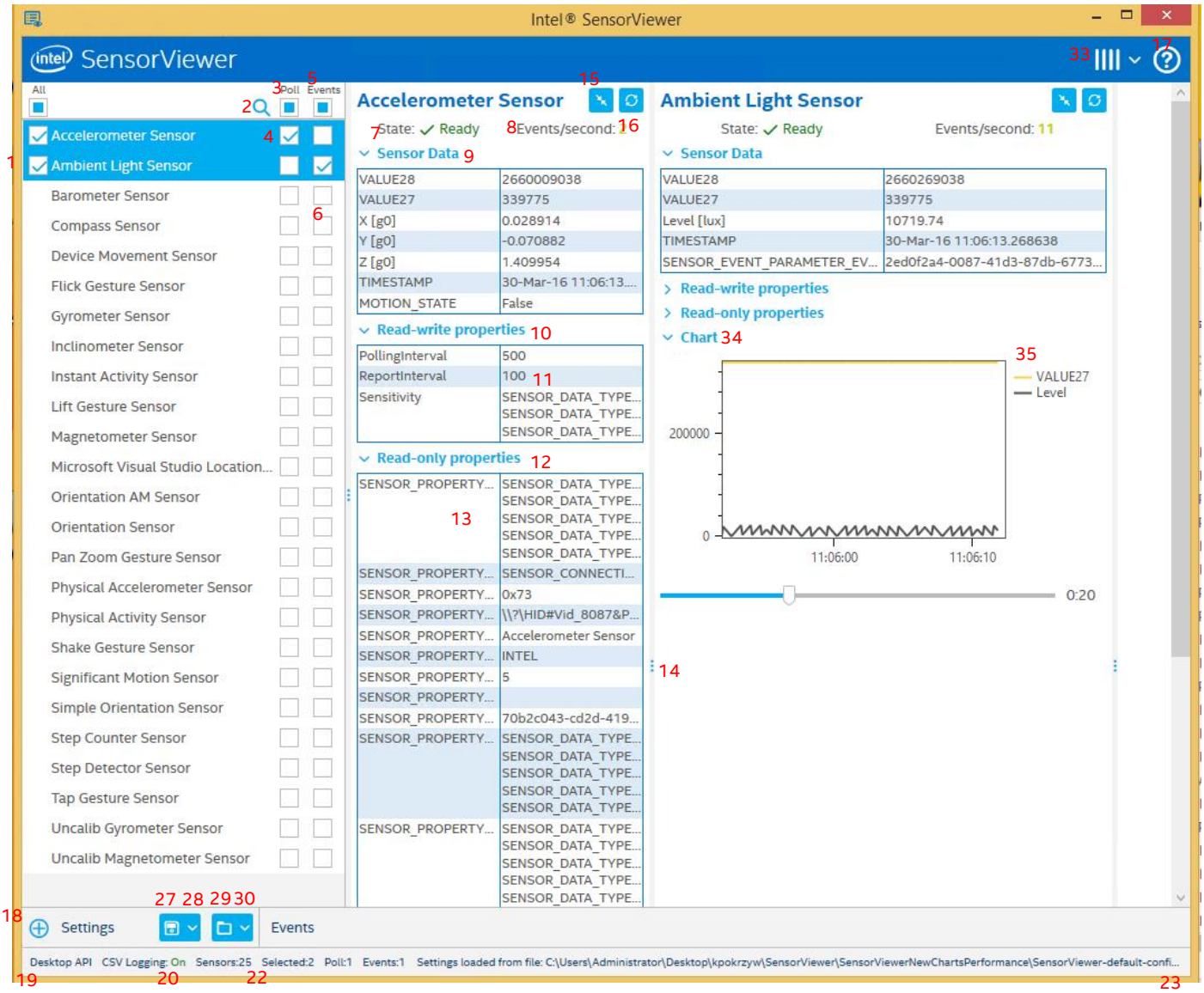
After the configuration is loaded, all sensors connected to the platform and available for the selected API will load and appear in the panel on the left.



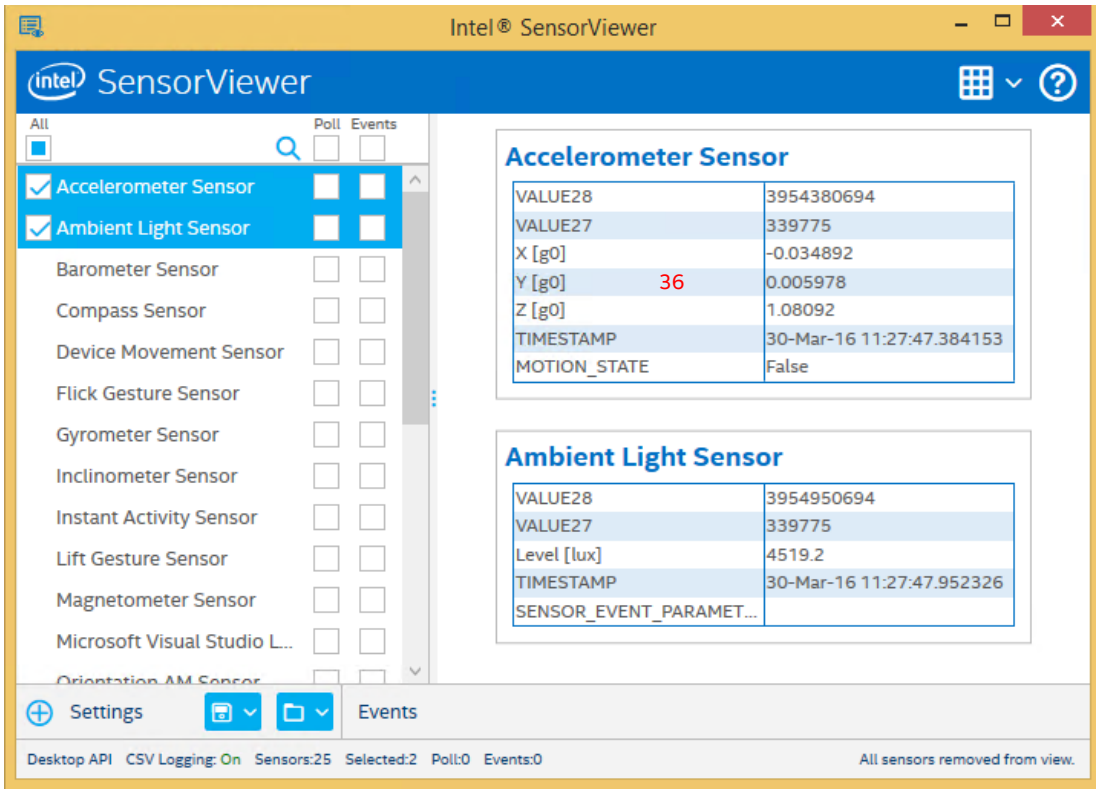
Picture 7: Application window on startup, after sensors are loaded using the Desktop API.

Click on any sensor in the list to select it. Its information will appear in the Sensor Details panel.

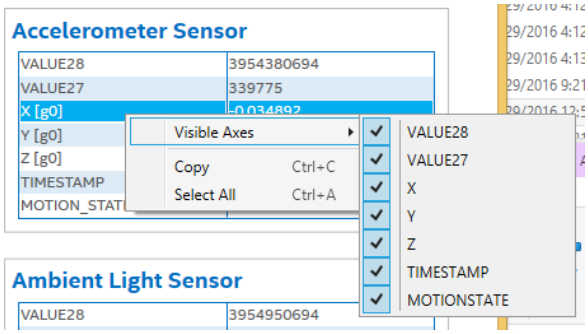
Pictures 8-13 display the GUI with each of its elements marked with red numbers. See below the images for a detailed treatment of each GUI element.



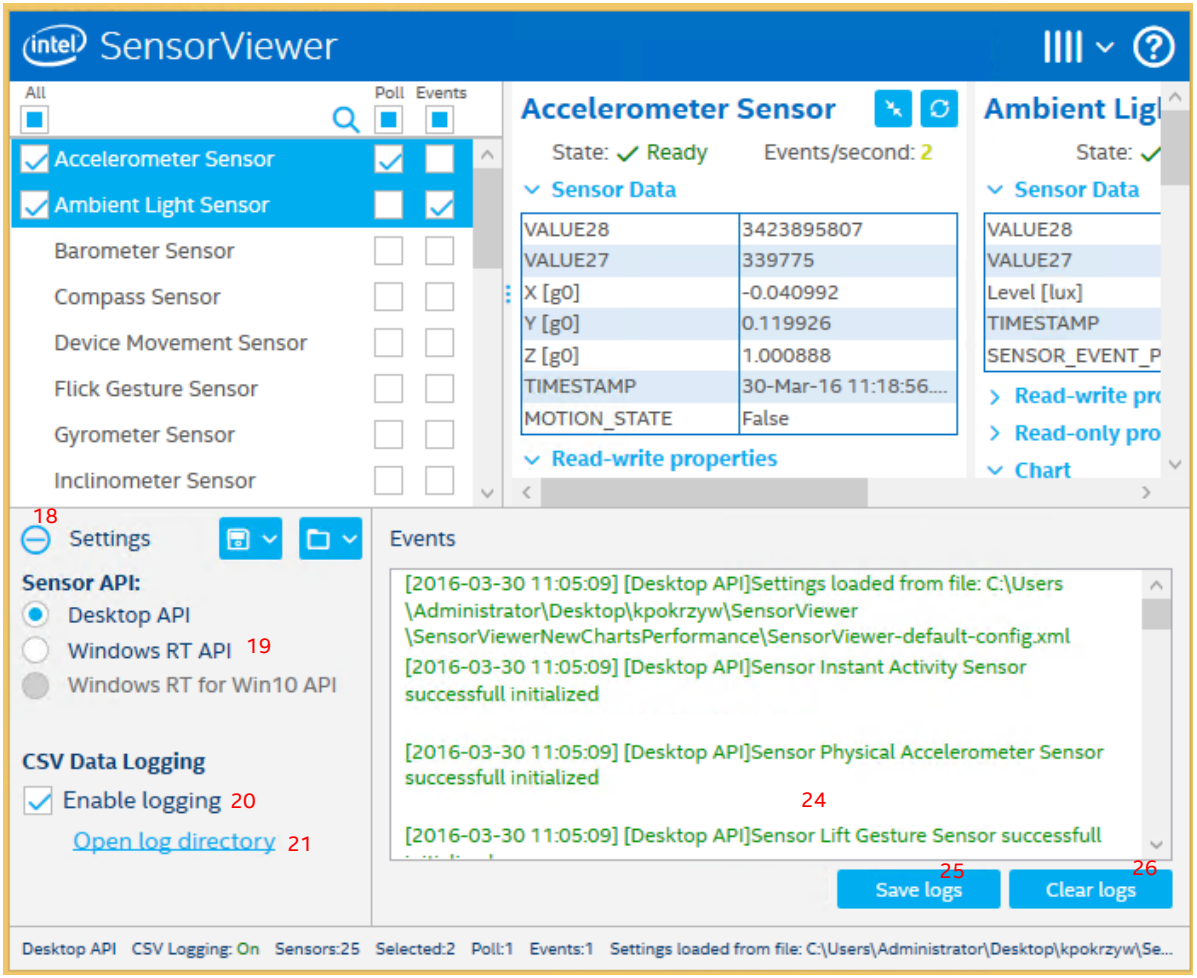
Picture 8: Application window with two selected sensors on column view.



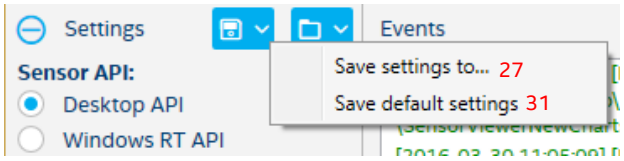
Picture 9: Application window with two selected sensors on grid view.



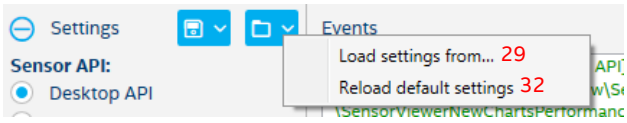
Picture 10: Expanded context menu for grid view.



Picture 11: Application window with expanded settings panel.



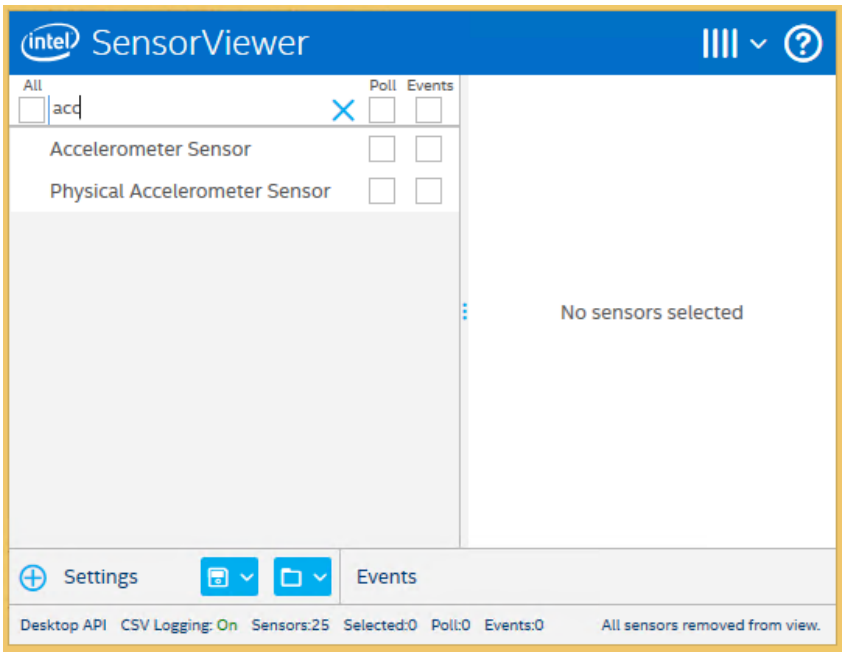
Picture 12: Expanded menu with save operations.



Picture 13: Expanded menu with load operations.

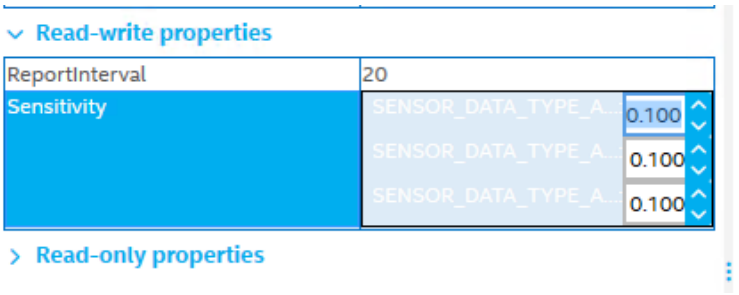
1. This checkbox indicates that the sensor beside it appears in the Sensor Details pane.

2. Search icon. Click to search for a sensor by name.



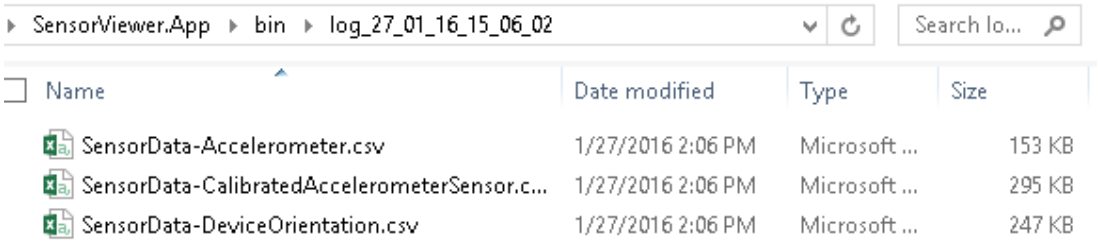
Picture 14: Application with active search box.

3. If this checkbox is selected, the application will begin polling for all available sensors that support polling. If it is deselected, polling will stop for all sensors.
4. Each sensor has its own checkbox that enables you to start or stop polling that sensor alone. This checkbox is disabled if the sensor doesn't support polling. Only sensors in the Ready state can be polled.
5. If this checkbox is selected, the application will subscribe to streaming data for all sensors.
6. Each sensor has its own checkbox that enables you to subscribe to streaming data for that sensor alone. Only sensors in the Ready state can stream data.
7. The current state of sensor. Possible states are "Ready", "Not Available", "No Data", "Initializing", "Access Denied", "Error", and "Removed".
8. The number of times per second this sensor is polled for data and/or streams a data report.
9. This sensor's latest data report.
10. A list of read/writable sensor properties.
11. Each read/writable sensor property can be clicked on and edited directly in the GUI.



Picture 15: Read-write property Sensitivity for accelerometer in edit mode.

12. Read-only sensor properties.
13. Adjustable column size. Drag the line to the left or to the right to expand and contract the columns in the table.
14. Drag this line to the left or to the right to change the amount of space given to this sensor's details.
15. Minimize button. Removes this sensor from the Sensor Details pane.
16. Refresh button. Updates the sensor's latest data report and properties.
17. Help button. Opens the SensorViewer manual.
18. Expandable panel with settings and event console.
19. The Windows sensor API currently in use. Possible values are "Desktop API", "Windows RT API", and "Windows RT for Win 10 API". This last API, only available in Windows 10, contains all the features and sensors available in the Windows RT API in addition to Windows 10-only features. When the Settings panel is expanded, you can select the API to use by clicking on the appropriate radio button.
20. Indicates if CSV logging is currently in progress. When the Settings panel is expanded, the "Log data to CSV" checkbox enables you to log all sensor data to CSV files. Each sensor that has polling or streaming enabled will have its sensor data written to a different CSV file in the logging directory.
21. A link to the logging directory currently in use. Clicking this link will open Windows Explorer at the appropriate location; the directory path will appear as a tooltip over the link. By default, the directory name is "log_[Date of log]_[Time of log]".



Picture 16: Screenshot of a logging directory with CSV logs for three sensors.

Picture 17: Screenshot of the tooltip indicating the logging directory.

22. The next item on the status bar, indicating the total number of sensors found, total number selected, and total number with polling or streaming activated.
23. The final item on the status bar, indicating the most recent change made to the application.
24. Console listing all application events. It displays the following:
    - Configuration loading and saving
    - API events:
        - Sensors added
        - Sensor states changed
        - Sensors removed
        - Other sensor events, such as Shake events for accelerometers
    - Errors
25. Save logs. Opens the standard Windows save file dialog window, in which you can save a .txt file with the content of the console.
26. Clear logs. Clears the contents of the console.
27. Save current settings to a file. Opens the standard Windows save file dialog window in which you can save the application's current state (API type, whether logging is active, details on sensor properties, etc.).
28. Arrow that opens a context menu that lists commands [27] and [31].
29. Load settings from a file. Opens the standard Windows load file dialog window, in which you can choose a file from which to load your desired application state.
30. Arrow that opens a context menu that lists commands [29] and [32].
31. Save default settings. Saves the application's current state to the SensorViewer-Default-config.xml file, which is loaded on startup by default.
32. Reload default settings. Loads the application state from the default SensorViewer-Default-config.xml file. If the file does not exist or is corrupted, the default state will be loaded instead.
33. Choose sensor view. Opens context menu, in which you can choose one of views: Column View or Grid View.
34. Chart data, in which you can see all data of sensor for last 5 minutes. You can check value of point on chart by holding left mouse button. You can go left, right. Up and down on chart by holding right mouse button and moving the mouse. You can zoom in and zoom out chart by scrolling mouse wheel. You can specify some area of chart and zoom in it by holding middle mouse button and selecting the area.
35. Legend of chart. You can disable axis visible on chart using context menu.

36. Grid View shows only Sensor Data of sensor. You can hide visible axis of sensor by using context menu (as you can see on picture 10). Hidden axis can be saved to application state file.

CLI Usage

The Intel® SensorViewer can be run in CLI mode. In command line mode, the SensorViewer reads an XML file indicating the application state to load, and begins recording data from sensors and saving them to CSV files. The application will end after a specified timeout period or when the Enter key is pressed. Its exit code is 0 on a successful exit for either of these two reasons, or 1 for an error.

Note: To use command line mode, you MUST load an existing application state that was saved from GUI mode.

CLI Parameters

The SensorViewer's CLI mode uses the following parameters:

- -**createConfig** filePath (or **–x** filePath or **–xml** filePath): In case you set this parameter you need to set up also apiType parameter. Creates new default settings for specified API (ApiType parameter) and saves it to filePath.
- -**config** *filePath* (or –**c** *filePath*): In case you didn't set up –createConfig parameter, this parameter is required. The path to the file with the application state to be loaded.
- -**timeout** *n* (or –**t** *n*): Optional. After *n* seconds, the program will automatically close.
- -**help** (or -**?**) – Shows help information.
- **-apiType** API (or **–api** API or **–a** API): In case you set up createConfig parameter, this parameter is required. Type of api, for which default settings will be saved. Possible values are:
  - "Desktop API" or "Windows7"
  - "Windows RT API" or "Windows8"
  - "Windows RT for Win10 API" or "Windows10"

Sample usages:

```
SensorViewer.exe –timeout 60 –config configFile.xml
```

The application state will be loaded from configFile.xml file, and the program will run for 60 seconds.

```
SensorViewer.exe –createConfig configFile2.xml –apiType "Desktop API"
```

The application will initialize "Desktop API" and will save current state of application and of sensors which was found by API to configFile2.xml file.