ibeo
automotive

Introduction to

ibeoSDK

Ibeo Automotive Systems GmbH
Merkurring 60-62
D - 22143 Hamburg
Phone: +49 - (0) 40 298 676 - 0
Fax: +49 - (0) 40 298 676 - 10
E-mail: info@ibeo-as.com

Version History

Date	Version	Changes
14-March-2016	1.0	Initial Version of this Introduction
14-Feb-2019	2.0	Adjustments to SDK Version 6.0

Table 0.1: Version History

Contents

1 Introduction

The Ibeo's SDK (ibeoSDK) is a C++ software package compatible with Linux and Windows that allows reading data generated by Ibeo's devices (e. g., Ibeo LUX laserscanners, ScaLa laserscanners, ECU and Ibeo Evaluation Suite) and can be used for developing customer specific functionalities.

Ibeo's data are available in the binary format, either from the laserscanners via the TCP/IP data stream or as IDC files, the Ibeo Data Container, used for saving data generated by Ibeo hardware devices. In order to receive specific data types, a listener class for each data type must be registered. The listener class is notified if the required information appears in the output data of the specified hardware device. The results are provided in the onData method and can then be output according to the customer's requirements, e. g. in the CSV format.

The ibeoSDK package includes various source code demos that exemplify how each data class is structured, and thus show how the customer can develop his own applications, e. g., to assess the error probability of Devices under Test (DuT). Available data types are for example scan points, vehicle data (ego data), GPS data, or object information.

NOTE

This document is only an overview of the ibeoSDK. For further information and instructions on using the ibeoSDK, refer to the complete ibeoSDK user documentation.

2 DataContainer

DataContainers are storing and sending Ibeo data of one specific **DataType**. **DataTypes** are identified by a unique **DataTypeId**. One part of the DataType definition is a serialization specification, a definition of the attributes stored in a byte array.

- The offset in the serialization
- Type used to store the data (int8,uint16, etc)
- The type implies also the size
- The name of the attribute

There are two different DataContainers: Special- and General Containers.

Special DataContainers are meant to read and store exactly one specific serialization without altering any of the data. Special DataContainer have the id of the serialization (DataTypeId) appended to their names: e.g. **ObjectList**2281, **Scan**2205, **Image**2403.

General DataContainers are the counterpart of that. They can read and store various serializations of a DataType for the price that data might have been lost or altered irreversibly in the process of import or export. General DataContainers does not have the serialization id's in their names: e.g. **ObjectList**, **Scan** or **Image**.

3 Data

Data for processing can be received from two sources: IDC files or IbeoDevices.

3.1 Ibeo Device

Instances of IbeoDevices are used to receive an IbeoDataContainer stream from a physical device or a remote IDC stream source via TCP/IP or UDP. Ibeo Devices can also store their data in IDC files.

3.2 IDC File

IDC is the short name for Ibeo-Data-Container and is defined as a sequence of DataTypes. Each DataType is preceeded by an **IbeoDataHeader**. This header contains (amongst other data) the **size** and **DataTypeId** of the following byte array.

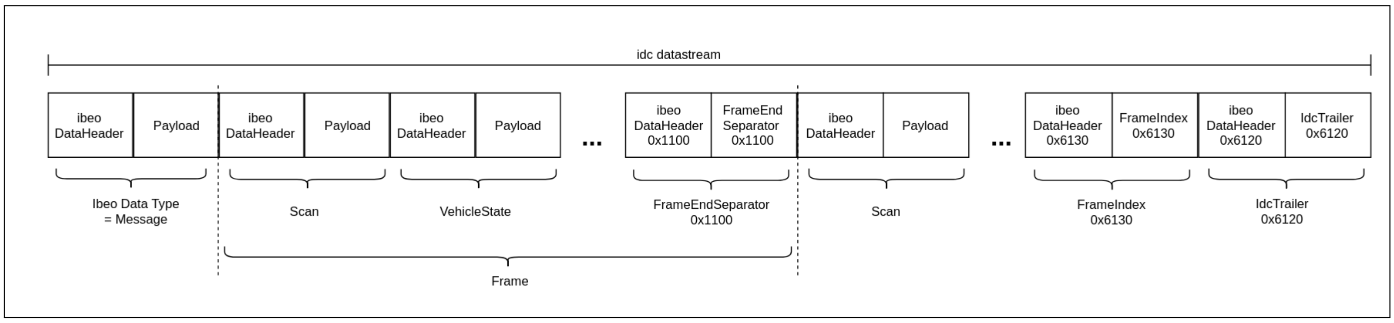

Figure 3.1: ibeoSDK Overview

3.3 Third Party Devices

The IbeoSDK is not restricted to Ibeo devices only, but also supports sensors from other manufacturers.

3.3.1 Velodyne HDL32 and HDL64

The VeldoyneDevice in the IbeoSDK is used as a proxy for these sensors. It has a number of configuration parameters, like the IP address and the UDP port, that should be set by the application to receive data from the sensor. There are also im- and exporter available (see section 4-Importer and Exporter) to convert the raw data from the sensor into IDC scans.
The VelodyneDevice can also operate in offline mode, i.e. it can be fed with UDP payload data, e.g. previously recorded by a logger, to produce IDC scans.

4 Importer and Exporter

Importers are templated classes defined for pairs of DataContainer-type and DataTypeId. An importers purpose is to deserialize the content of DataTypes with the given DataTypeId and store it into an instance of the given DataContainer type.
All importers have a method to create a DataContainer of the given type.

Exporters are templated classes defined for pairs of DataContainer-type and DataTypeId. An exporters purpose is to serialize the content of an instance of the give DataContainer type following the rules of the DataType with the given DataTypeId.

Names for Importer and Exporters follow a strict naming pattern. More precisely the naming pattern of their typedefs:

```
template<>
class Importer<Odometry9002, DataTypeId::DataType_Odometry9002>
: public RegisteredImporter<Odometry9002, DataTypeId::DataType_Odometry9002>
{
...
}; //Odometry9002Importer9002

//==================================================================================

using Odometry9002Importer9002 = Importer<Odometry9002,DataTypeId::DataType_Odometry9002>;
```

An Importers/Exporters name is defined by the class name Importer/Exporter and its two template parameters, container class (derived from DataContainerBase) and DataTypeId. The typedef-Alias (generated by 'using') and the file names are given by the container name, the word Importer/Exporter and the DataTypeId (in hex) of the Serialization to be read (for Importer) or to be written (for Exporter). In the above example this is:
Odometry9002Importer9002

- **Odometry9002** is the name of a container.

- **9002** is the DataTypeId of the serialization 0x9002.

5 Processing

The IbeoSDK provides a lot of support classes that facilitate the processing of data. These classes can be found in the include folder of *ibeo/sdk/common*.
Some examples:

5.1 Math

Math has many features and shortcuts related to mathematic functions and calculations.

5.2 Matrizes

A matrix class is to create a matrix of the size in its name e.g. Matrix2x2

5.3 Rectangle

A vector class for which can store a rectangle with center point and size (x and y extension) and a rotation.

5.4 TimeInterval

TimeInterval can store a start and end time(NTPTime).