IS-1500
Advanced Developer Guide

700 Technology Park Drive, Suite 102
Billerica, MA 01821 USA

Phone +1 781 541 6330 • Fax +1 781 541 6329

www.intersense.com

MNL-0025 (B)

InertiaCam Advanced Developer Guide

Compatible with IS-1500 InertiaCam Firmware 1.0, sfHub version 2.52 and sfAccess 1.16.

Contacting Thales Visionix

Please contact us if you need assistance.

Thales Visionix, Inc.
700 Technology Park Drive, Suite 102
Billerica, Massachusetts 01821 USA

Telephone: +1 781 541 6330

Fax: +1 781 541 6329

Internet: http://www.intersense.com

Technical Support:

Telephone: +1 781 541 7624
email: techsupport@thalesvisionix.com

Sales:

Telephone: +1 781 541 7650
email: sales@thalesvisionix.com

Patents

The list below identifies the protection granted by the Government of the United States and by other governments to Thales Visionix for InterSense products.

THALES

U.S. Patents

5645077, 5807284, 6162191, 6361507, 6786877, 6176837, 6409687, 7395181, 6314055 B1, 6757068 B2, 7301648, 6474159 B1, 6681629 B2, 7000469 B2, 7231063 B2, 6922632 B2, 7725253, 7900524, 8224024, 8473241, 8696458, 8762091, 8972182 and Patents Pending.

Foreign Patents

Europe 1280457 Israel: 152359 Taiwan: 162248 Europe: 1071369
China: 99807510.8 Hong Kong: 1039884 Japan: 4690546.

Trademarks

InterSense® is a registered of Thales Visionix, Inc. All other trademarks are the property of their respective owners.

Table of Contents

1. Introduction

This document describes the IS-1500 sfAccess library. You can use sfAccess instead of the InterSense Library (**isense.dll / libisense.so / libisense.dylib** files included in the product download) to incorporate advanced features of the IS-1500 and the InertiaCam into your host application. If your application only needs tracking output (orientation and position for example) or if compatibility with other InterSense tracker models is required, you should use the InterSense Library.

2. sfAccess API

2.1. Sample Program Overview

The library is distributed with sample programs written in C++ to demonstrate usage. It includes a header file (**libsfaccess.h**) with data structure definitions and function prototypes. Additional API descriptions below are also available in the header file.

`main.cpp`	Main loop of the program. All API calls are made from here.
`libsfaccess.h`	Header file containing function prototypes and definitions. This file should not be modified.
`sfosiTypes.h`	Basic type definitions.
`sfAccess.dll` **`libsfaccess.so`**	The sfAccess DLL and shared libraries. Place these files in the Windows system directory, system library directory, or in the working directory of the application (additional configuration may be required on Linux platforms).

2.2. Basics of Using sfAccess Library

To use the sfAccess library, first connect the InertiaCam to your host system and launch the sfHub application as described in the *IS-1500 User Guide*.

The API provides an extensive set of functions that can read tracker data and access tracker configuration, but in its simplest form can be limited to just a few function calls.

open()
getTrackingDataLatest()
close()

The example in main.cpp included with the sample code is slightly more sophisticated to provide feedback about the status of the connection and flow of data.

Data output functions, such as getTrackingDataLatest(), are non-blocking and output a valid flag. If no data is available, the function returns immediately with the valid flag set to false. If

data is available, buffered data is returned in the order it was received with the valid flag set to true and the data is removed from the buffer.

The sfAccess is not thread-safe. Your host application is responsible for restricting access to a single thread at a time.

2.2.1. sfaccess.ini

Settings for sfAccess are stored in a file named sfaccess.ini typically residing in the working directory. The ini file is read whenever open() is called. Default values are used if the file is not found or for any item that is not specified in the file. Item names are not case-sensitive. Commonly used settings are shown below.

Setting	Typical	Definition
Verbosity	0	Message output control bits: 0 off 1 warnings to stdlog 2 info to stdout 4 errors to stdlog
TcpSfRxEnabled	true	Enable or disable TCP connection.
TcpSfRxAddr	localhost	IP address of sfRx. Localhost can be used if sfAccess runs on the same host as sfHub. Dotted IP address is required if sfAccess is on a separate host.
UdpSfRxPort	9001	Raw data UDP port from sfHub.
UdpSfCorePort	0	Route NFT tracking output to getTrackingData() and related functions.
UdpNftPort	9007	UDP port for NFT data from sfHub.
PredictionS	0	Prediction interval in seconds.
BoresightRef	0	Boresight Euler angles in radians.
TipOffset	0	Tip offset in meters.
VirtualSyncCtrl	0	0 off 1 on

2.2.2. Streaming Mode

Streaming mode must be on in order for sfAccess to receive data from the tracking system. If the TCP connection is available, data output functions automatically start streaming mode when they are called. All other functions turn off streaming if it is on.

If the TCP connection is not available, your application can still receive data using the get data functions if streaming mode is already enabled in sfHub.

2.3. Data Type Definitions

The following data type definitions apply to all IS-1500 API functions for both C++ and C.

Data Type	Definition
`bool`	Boolean
`string`	char string

`byte`	unsigned char
`int16`	16-bit signed integer
`uint16`	16-bit unsigned integer
`int32`	32-bit signed integer
`uint32`	32-bit unsigned integer
`int64`	64-bit signed integer
`uint64`	64-bit unsigned integer
`float`	32-bit floating point
`double`	64-bit floating point
`size`	unsigned integer
`enum`	integer enumeration

3. API Reference

The sfAccess API provides functions for accessing various kinds of data and for performing various operations. The most useful subset of those functions are included in this section.

Unless otherwise indicated, all functions return boolean status set to true on success or false on failure.

The sfAccess library includes both C++ and C APIs. The C API versions of the functions are prefixed with SfAccess_. They are equivalent to the C++ versions and take an additional handle parameter returned by SfAccess_create(), which should be called once before any other function. SfAccess_destroy() can be called later to close the connection to the tracker and release the handle.

The following functions always return the latest data available.

getTrackingDataLatest()
getTrackingDataAtTime()

Note that the functions listed below store data in internal queues and output data in the order it is received. To ensure that you are reading the most recent data, you need to call these functions until the valid flag is false.

getTrackingData()
getImuData()
getDiagData()

3.1. General

open()

```
bool
open()
```

This function reads the sfaccess.ini file and opens a connection to the tracker. The getStatus() function can be called afterward to check the connection details.

This function should be called before any other function with the exception of getVersion().

When you finish working with the tracker, call close() to close your connection to it.

C API function: SfAccess_open().

getStatus()

```
bool
getStatus(...)
```

Retrieves the status of internal conditions of the InertiaCam sensor and sends the information to the four outputs you pass it. This function has no input arguments.

The portStatus output changes only when your application calls open() or close(). The other outputs, receiveStatus, errorStatus, and commStatus refresh each time your application calls this function. Each status is `0` when no new data is available.

portStatus	Port status flags: 0 Port closed. 1 Port open. Port status flags: 01h TcpSfRxPort 02h UdpSfRxPort 04h UdpSfRximgPort 08h UdpSfRxInputPort 10h UdpSfCorePort 20h UdpSfHubPort
receiveStatus	Receive status flags to indicate whether or not data has been received on a particular port since the last call of this function: 0 No new data. 1 New data. Receive status flags: 01h Reserved 02h UdpSfRxPort 04h UdpSfRximgPort 08h Reserved 10h UdpSfCorePort 20h UdpSfHubPort
errorStatus	Error status flag to indicate whether or not an error has occurred since the last time this function was called: 0 No new errors. 1 At least one new error.
commStatus	sfHub communication status flags to indicate status of communication with InertiaCam: 0 Not available. 1 Searching for sensor. 2 Streaming data from sensor. 3 Sensor paused. 4 Sensor disconnected.

C API function: SfAccess_getStatus().

close()

```
bool
close()
```

Closes the connection to the tracker opened when you called open() and releases all resources. This function has no arguments.

C API function: SfAccess_close().

getVersion()

```
string
getVersion()
```

Returns a string containing the sfAccess version number in the format *x.yy.zz*.

You do not have to call open()before calling this function. The version numbers are also available in the library header file at compile time. This function has no arguments.

C API function: SfAccess_getVersion().

getLastError()

```
string
getLastError()
```

Returns an error message after any function call returns a failed status. If no error has occurred as a result of the last function call and the previous error has been cleared, returns an empty string. The error message does not clear on its own and persists until the next error occurs or until the application calls clearLastError().

C API function: SfAccess_getLastError().

clearLastError()

```
bool
clearLastError()
```

Clears error message returned by getLastError().

This function has no arguments.

C API function: SfAccess_clearLastError().

3.2. Tracking Output

sfHub UDP connection is required to receive data with these functions (see UdpSfNftPort).

getTrackingData()

```
bool
getTrackingData(…)
```

Reads next available tracking data set. If streaming is off, it is turned on. Constant prediction is applied if PredictionS is greater than 0. Any arguments for this function that are not listed below are reserved for future list.

`valid`	`bool`	`out`	True if data is valid, false if data is not available.
`sid`	`uint32`	`out`	Sequence ID.
`timeImu`	`uint64`	`out`	IMU data arrival timestamp (us).
`trkState`	`uint16`	`out`	Tracker state. 0 No comm. 1 Initializing. 2 Tracking. 3 Diverged.
`trkStatus`	`uint16`	`out`	Reserved.
`trkCbn[3][3]`	`float`	`out`	Tracker rotation matrix, n-frame, row-major order.*
`trkRot[3]`	`float`	`out`	Tracker orientation Euler angles, roll/pitch/yaw, n-frame (rad).*
`trkPos[3]`	`float`	`out`	Tracker X/Y/Z position, n-frame (m).
`trkVel[3]`	`float`	`out`	Tracker X/Y/Z velocity, n-frame (m/s).
`trkAcc[3]`	`float`	`out`	Tracker X/Y/Z acceleration, n-frame (m/s/s).
`trkOmega[3]`	`float`	`out`	Tracker X/Y/Z angular rate (rad/s).

*Advanced by PredictionS × trkOmega.

C API function: SfAccess_getTrackingDataExt().

getTrackingDataLatest()

```
bool
getTrackingDataLatest(…)
```

Similar to getTrackingData() except it reads the most recent tracking data set independent of buffered data sets. This function can be used to obtain tracking data while preserving the buffered data for subsequent calls to getTrackingData() or getTrackingDataAtTime(). The valid flag is always true unless no tracking data was received since open() was called.

C API function: SfAccess_getTrackingDataLatestExt().

getTrackingDataAtTime()

```
bool
getTrackingDataAtTime(…)
```

Similar to getTrackingDataLatest() but with virtual synchronization (if VirtualSyncCtrl is set to 1 in sfaccess.ini). The most recent tracking data set is propagated to the specified time. Constant prediction is also applied if PredictionS>0.

Arguments are the same as for getTrackingDataLatest() with the addition of the time parameter:

`time` `uint64` `in` Time to which to predict (us), obtained using getTimeUs().

Since coherent timestamps are required, virtual synchronization is supported only if sfHub and SfAccess are used on the same host.

Proper operation of getTrackingDataAtTime() requires that getTrackingData() not be used. This is because getTrackingDataAtTime() utilizes buffered data for the virtual synchronization algorithm and calling getTrackingData() removes the buffered data. It is safe to use getTrackingDataLatest() instead of getTrackingData().

C API function: SfAccess_getTrackingDataAtTimeExt().

3.3. Tracking Output User Settings

These functions specify settings that are only applied locally by sfAccess. They may be called at any time, but settings are overridden by the ini file when open() is called if the file specifies it.

setPrediction()

```
bool
setPrediction(float interval)
```

Set constant prediction to be applied to these orientation outputs: trkCbn and trkRot.

`interval` `float` `in` Prediction interval in seconds.

C API function: SfAccess_setPrediction().

setBoresightRef()

```
bool
setBoresightRef(float euler[])
```

Set boresight reference Euler angles to be applied to these orientation outputs: trkCbn, trkRot, trkOmega, trkRotSig.

`euler[3]` `float` `in` Roll/pitch/yaw in radians.

C API function: SfAccess_setBoresightRef().

setTipOffset()

```
bool
setTipOffset(float offset[])
```

Set tip offset to be applied to position output trkPos to change the default tip location.

`offset[3]` `float` `in` X/Y/Z offset in meters.

C API function: SfAccess_setTipOffset().

3.4. Image Output

getImageBufSize()

```
uint32
getImageBufSize()
```

Returns size of internal image buffer in bytes. Useful for allocating buffer space in application code.

C API function: SfAccess_getImageBufSize().

getImage()

```
bool
getImage(…)
```

When progress reaches 100%, copies the most recently received image into the supplied buffer. If your application calls this function again before a new image finishes arriving, progress can be up to 99% only; because progress never reaches 100%, no copying takes place.

All images this function retrieves are double-buffered. The function can read the most recently received image while the new image is in the process of arriving. If it has not completely read the older image by that time the new image finishes arriving, this function discards the older image.

getImage() writes pixels to the buffer data in row-major order. The function accompanies each image with a sequence ID to associate the image with its corresponding inertial data.

If streaming is off, this function turns it on.

Images are received via shared memory so to use getImage(), sfHub and SfAccess must be used on the same host.

`progress` `int32` `out` Percentage of progress on receiving image:

100	Image is output to supplied image buffer.
<100	No image available (other outputs are invalid).
0	Waiting for new image to start arriving.
1 to 99	New image is in the process of arriving.

`format` `enum` `out` Image format type, signified by a number from those defined in the table below:

Image Format	Resolution	Color	Bits Per Pixel
40	1280 × 960	Gray	8
50	640 × 480	Color	24

`sid` `uint32` `out` Sequence ID that associates the image with its inertial data.

`buf` `byte` `out` Image buffer (by reference).

`bufsize` `size` `in` Image buffer size in bytes.

C API function: SfAccess_getImage().

getImageInfo()

```
bool
getImageInfo(…)
```

Reports image information given its format.

format	enum in	Image format - see getImage().
size	int32 out	Image size in bytes.
w	int32 out	Image width in pixels.
h	int32 out	Image height in pixels.
bpp	int32 out	Bytes per pixel.

C API function: SfAccess_getImageInfo().

3.5. Low-Level Output

The sfHub UDP connection is necessary to receive data from these functions (see UdpSfRxPort).

getImuData()

```
bool
getImuData(…)
```

Reads next available set of inertial data from the IMU. If streaming is off, this function turns it on.

sel	enum in	Use CAMERA.
valid	bool out	True if data is valid, false if data is not available.
sid	uint32 out	Sequence ID.
pid	uint32 out	Packet ID.
dt	float out	Delta time or sample period in seconds.
flags	int32 out	Discrete flags.*
dv[3]	int16 out	Delta V (x, y, z).*
dtheta[3]	int16 out	Delta theta (x, y, z).*
a[3]	float out	Acceleration (m/s/s).
w[3]	float out	Angular rate (rad/s).

Timestamp in seconds can be computed as follows: T = sid × dt.

*Refer to NavChip ICD for additional details.

C API function: None.

getImuCrs()

```
bool
```

```
getImuCrs(enum ImuSelect sel,
          bool *valid,
          byte crs[])
```

Reads most recently received IMU configuration register set. If streaming is off, this function turns it on.

sel	enum	in	Use CAMERA.
valid	bool	out	True if data is valid, False if data is not available.
crs[32]	byte	out	CRS data.*

*Refer to NavChip ICD for additional details.

C API function: None.

getDiagData()

```
bool
getDiagData(…)
```

Reads next available diagnostic data set. If streaming is off, it is turned on. Any arguments for this function that are not listed below are reserved for future list.

valid	bool	out	True if data is valid, False if data is not available.
sid	uint32	out	Sequence ID.
exposure	uint32	out	Camera exposure (µs).
ncTemp	float	out	NavChip temperature (°C).
commStatus	uint32	out	Communication status. 0 Not available. 1 Searching. 2 Streaming. 3 Paused.
trkType	uint8	out	Tracker type. 0 Invalid. 1 IS-1200 2 IS-1500 3 InertiaCubeNC
trkQuality	uint8	out	Tracking quality percentage (0-100).
trkState	uint8	out	Tracking state 0 No communication 1 Tracking, not locked (relative). 2 Tracking, locked (fixed). 3 Lost
trkMode	uint8	out	Tracking Mode 1 sfCore 3-DOF 2 sfCore 6-DOF 3 NFT+Fiducial

4 NFT+GPS
5 NFT+GPS+Fiducial

C API function: SfAccess_ getDiagData().

3.6. TCP Access

The TCP connection is required for these functions.

setStreaming()

```
bool
setStreaming(bool state)
```

Sets streaming state. Normally, streaming state is managed automatically but it can be set manually if necessary for diagnostic purposes.

state	bool	in	Streaming state. true On. false Off.

C API function: SfAccess_setStreaming().

getStreaming()

```
bool
getStreaming()
```

Returns streaming state. No connections required.

C API function: SfAccess_getStreaming().

4. Thales Visionix, Inc. End User License Agreement

Complete text of the End User License Agreement you agree to with purchase of this software is provided below:

This License Agreement (“Agreement”) is an agreement between you (“Licensee”) and Thales Visionix, Inc. (“Licensor”). It governs your use of the software supplied to you by Thales Visionix, Inc. ("the Software") and related documentation. By downloading, installing or otherwise using the Software, you agree to be legally bound by the terms of this Agreement. Please read the entire contents of this agreement.

IMPORTANT NOTE

You should make a backup of any important information, valuable data, or Software on your PC before installing this or any other software. Also, you should make regular backups of the Software and any data files, so that in the event of data loss, your information may be restored.

1. DEFINITION

(a) “Documentation” means the standard end user manual for the Product provided by Licensor.
(b) “Product” means all Licensor equipment including, but not limited to, the InertiaCube, IS-900, and IS-1200 product families.
(c) “Software” means all Licensor software code including sample source code, compiled code, and embedded firmware.

2. LICENSE

(a) Grant of Rights. Licensor hereby grants to Licensee a non-exclusive, royalty-free, perpetual license to use the Software solely with Licensor Product and solely for the operation of Product in accordance with its documentation. Licensee may redistribute Licensor libraries, isense.dll, libisense.so, libisense.dylib, sfAccess.dll and libsfaccess.so as required for the operation of Licensee products.

(b) Restrictions. The Software (and any copy thereof) is licensed not sold, and Licensee receives no title to or ownership of the Software and no rights other than those specifically granted in Section 2(a) above. With the exception of Licensor sample source code, Licensee may not modify, reproduce, and create derivative works of the Software and shall not attempt to decompile or otherwise reverse engineer the Software. Licensee shall not remove any proprietary or copyright notices of Licensor in the Software or any copies thereof.

3. WARRANTY

This software and accompanying written materials (including instructions for use) are provided "as is" without warranty of any kind. Further, Licensor does not warrant, guarantee, or make any representations regarding the use, or the results of use, of the software or written materials in terms of correctness, accuracy, reliability, currentness, or otherwise. The entire risk as to the results and performance of the software is assumed by Licensee. If the software or written materials are defective, Licensee, and not Licensor or its dealers, distributors, agents, or employees, assume the entire cost of all necessary servicing, repair, or correction.

The above is the only warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose, which is made by licensor on this product. No oral or written information or advice given by licensor, its dealers, distributors, agents or employees shall create a warranty or in any way increase the scope of this warranty and you may not rely on any such information or advice.

4. LIMITATION OF LIABILITY

Neither licensor nor anyone else who has been involved in the creation, production or delivery of this product shall be liable for any direct, indirect, consequential or incidental damages (including damages for loss of business profits, assets, business interruption, loss of business information, and the like) arising out of the use or inability to use such product even if licensor has been advised of the possibility of such damages.

5. TERMINATION

(a) Termination. This Agreement shall terminate automatically and immediately upon any breach of this agreement by Licensee, including but not limited to any action in violation of the license rights and restrictions set forth in Section 2.

(b) Effects of Termination. Upon termination of this Agreement, the licenses granted in Section 2(a) will terminate and Licensee will cease all use of the Product and delete all copies of any Software (including Upgrades and Updates stored on Licensee computers) in its possession or control. This Section 5(b) shall survive termination of this Agreement.

6. MISCELLANEOUS

(a) Notices. Notices to Licensor pursuant to this Agreement will be sent to 22605 Gateway Center Drive, Clarksburg, Maryland 20972, Attention: Contracts Department. Such notices will be deemed received at such addresses upon the earlier of (i) actual receipt or (ii) delivery in person, or by certified mail return receipt requested.

(b) U.S. Government Use. The Software is a “commercial item” as that term is defined at 48 C.F.R. 2.101, consisting of “commercial computer software” and “commercial computer software documentation” as such terms are used in 48 C.F.R. 12.212.

(c) Export. Licensee understands and recognizes that the Product is export-controlled and agrees to comply with all applicable export regulations. The Product may not be exported to any country outside the country where the Product was acquired without Licensor’s prior written consent.

(d) Assignment & Successors. Licensee may not assign this Agreement or any of its rights or obligations hereunder.

(e) Choice of Law & Jurisdiction. This Agreement will be governed by the laws of the United States of America.

(f) Severability. To the extent permitted by applicable law, the parties hereby waive any provision of law that would render any clause of this Agreement invalid or otherwise unenforceable in any respect. In the event that a provision of this Agreement is held to be invalid or otherwise unenforceable, such provision will be interpreted to fulfill its intended purpose to the maximum extent permitted by applicable law, and the remaining provisions of this Agreement will continue in full force and effect.

(g) Entire Agreement. This Agreement sets forth the entire agreement of the parties and may not be modified except by a written agreement signed by Licensor.