ESI API

Documentation

Version 3.0 – Jul 2024

dSPACE

How to Contact dSPACE

Mail:	dSPACE GmbH Rathenaustraße 26 33102 Paderborn Germany
Tel.:	+49 5251 1638-0
Fax:	+49 5251 16198-0
E-mail:	info@dspace.de
Web:	http://www.dspace.com

How to Contact dSPACE Support

If you encounter a problem when using dSPACE products, contact your local dSPACE representative:

- Local dSPACE companies and distributors: http://www.dspace.com/go/locations
- For countries not listed, contact dSPACE GmbH in Paderborn, Germany. Tel.: +49 5251 1638-941 or e-mail: support@dspace.de

You can also use the support request form: http://www.dspace.com/go/supportrequest. If you are logged on to mydSPACE, you are automatically identified and do not need to add your contact details manually.

If possible, always provide the serial number of the hardware, the relevant dSPACE License ID, or the serial number of the CmContainer in your support request.

Contents

Confidentiality

Confidentiality Note

This document in its entirety is PUBLIC and may also be distributed to parties outside dSPACE GmbH.

Transmission of Data

No special measures are required for data transmission.

Document Revision

Objective

This section includes the document revision information.

Versions

Version	Date	Author	Remarks
0.9	25.04.2024	MaximilianSc	Initial Version
3.0	13.06.2024	MaximilianSc, CarstenMe	Update to protocol version 3

Introduction

Basics

With the dSPACE ESI API, a custom application for sending data to the dSPACE ESI Unit can be implemented. This is especially useful when it is necessary to integrate the ESI Unit in an existing replay setup.

Supported Hardware

You can use the dSPACE ESI API with any ESI Unit equipped with a recent firmware version. Please contact dSPACE for further details.

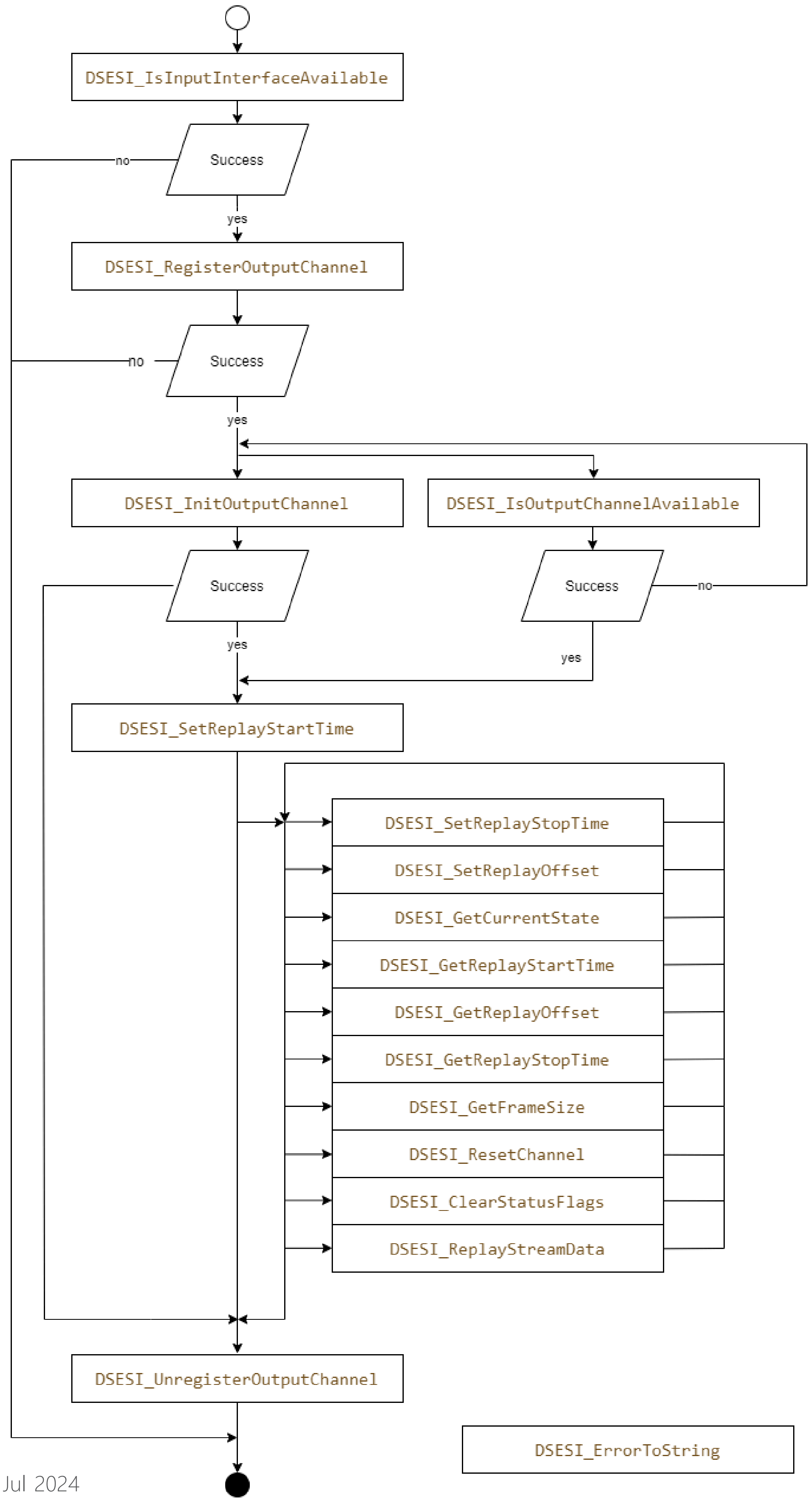
Accessing the dSPACE ESI API

The dSPACE ESI API is a functional 64-bit shared library. It is available for Ubuntu 18.04, Ubuntu 20.04, 22.04 and Windows. The interface is provided via export C function, which can be used with an application written in C or C++.

Single application support

The dSPACE ESI API allows access from a single client to the dSPACE ESI Unit. Additionally, a dSPACE SCALEXIO system can be used so send control messages to the dSPACE ESI Unit.

Overview of API functions



Files of dSPACE ESI API

The following list shows the important files of dSPACE ESI API:

```
* DSESIApi/
  * bin/
    * DSESIApi.dll (only Windows)
    * libDSESIApi.so (only Linux)
  * cmake/
  * include/
    * DSEsiApi.h
  * lib/
    * DSESIApi.lib (only Windows)
  * Demo/
    * CMakeLists.txt
    * DSESIApiDemo.cpp
```

Software Requirements

- The dSPACE ESI API supports the following operating systems and compilers:
  - Windows 10 (64bit), Visual Studio 2019
  - Ubuntu 18.04, gcc
  - Ubuntu 20.04, gcc
  - Ubuntu 22.04, gcc
- The dSPACE ESI API requires an ESI Unit featuring the V3 communication protocol. This protocol is used in firmwares which are using the Replay Core Firmware V3.0.0.

NOTICE

On Linux, the **`ping`** command must be available.

Preparation

In order to reach optimal performance, dSPACE recommends using jumbo frames for the data connections to the ESI Unit, so the jumbo frames feature needs to be enabled for each ethernet interface that is connected to a data interface from the ESI Unit. The interface that is connected to the control interface makes no use of jumbo frames.

Timestamps and Synchronicity

To ensure that all data samples are replayed at the correct time, each data sample has it's indiviudal timestamp. This timestamp is called **`Replay Timestamp`**. The data is replayed, as soon as the check

```
ESI Time > Replay Timestamp
```

becomes true. As the name suggests, **`ESI Time`** is the time of the ESI Unit, which can be the system time or the PTP time, if the ESI Unit is synchronized via PTP.

NOTICE

The ESI unit uses **`ptp4l`** for synchronization. While this tool supports different synchronization protocols, the dSPACE SCALEXIO system **only** support IEEE 802.1AS (gPTP).

As the data samples were usually recorded in a different time domain as the ESI Unit is using when replaying data, the **`Recording Timestamp`** is converted to the **`Replay Timestamp`** by the following formula:

```
Replay Timestamp = Recording Timestamp - Offset + Starting Timestamp
```

where **`Offset`** is the start of the recording (thus **`Recording Timestamp`** - **`Offset`**≈ 0 for the first sample) and **`Starting Timestamp`** is the start time of the replay.

Data is not replayed anymore, if

```
Replay Timestamp > Stop Timestamp
```

Usually, the start time of the replay is bigger than the time of the ESI Unit, when the first data sample is received, so data can be buffered at the ESI Unit. If the start time is equal or smaller than the time of the ESI Unit, when the first data sample is received, the samples are replayed directly without buffering and timing accuracy cannot be guaranteed.

Demo

The demo for dSPACE ESI API shows the following steps:

1. Checking if the ESI Unit's input and output are available
2. Registering an output channel
3. Initializing the output channel
4. Set the start time
5. Send some replay data
6. Querying the current state
7. Unregister the output channel

For details and more information see DSEsiApiDemo.cpp.

API Description

Data Types

The dSPACE API makes uses of the following data types:

Data Type	Description
`DSTEsiChannelNumber`	The channel number of the ESI Unit
`DSTEsiChannelHandle`	A handle to uniquely identify the channel
`DSTEsiInputInterfaceT`	A struct to configure the channel. See description below
`DSTEsiStatusT`	A struct which contains the state of the ESI Unit. See description below
`DSTEsiError`	An enum for the error returned by the functions. See description below

Where to go from here

Information in this section

DSTEsiInputInterfaceT

Member	Data Type	Description
controlIP	**`const char*`**	Control IP of the ESI Unit
controlPort	**`uint16_t`**	Control port of the ESI Unit
dataIP	**`const char*`**	Data IP of the ESI Unit
dataBasePort	**`uint16_t`**	Data base port of the ESI Unit
localIP	**`const char*`**	Optional: The local ip the ESI API binds to to receive messages from the ESI Unit. The value NULL defaults to 0.0.0.0
localPort	**`uint16_t`**	Local port to receive messages from the ESI Unit
maxPayloadSize	**`uint16_t`**	Maximum payload size.
timeoutGet_ms	**`uint32_t`**	Timeout after which the Get functions are returning

Member	Data Type	Description
timeoutStream_ms	`uint32_t`	Timeout after which the ReplayStreamData function is returning (e.g. no free buffer on the ESI Unit)

NOTICE

MTU
The default MTU size in most of the Ethernet networks is 1500 bytes. For using the ESI API, an MTU of 9000 Bytes (also called Jumbo Frames) is needed.

NOTICE

Maximum payload size
The maximum payload size is calculated by
`MTU - (sizeof(ReplaySensorMessageHeader) + IP_HEADER_SIZE + UDP_HEADER_SIZE)`,
where `sizeof(ReplaySensorMessageHeader)` is 24 Byte, `IP_HEADER_SIZE` is 20 Byte and `UDP_HEADER_SIZE` is 8 Byte.

DSTEsiStatusT

Member	Data Type	Description
enabled	`bool`	Channel is enabled
loss	`bool`	Loss detected
ptpSynced	`bool`	PTP is actually synchronized
ptpActive	`bool`	PTP activated on the ESI Unit
freeFifoEntries	`unit32_t`	Number of free fifo entries on the ESI Unit
freeBufferBytes	`uint64_t`	Free buffer in bytes on the ESI Unit
currentTime_ns	`uint64_t`	The current time in nano seconds on the ESI Unit

DSTEsiPacketPropertiesT

Member	Data Type	Description
timestamp	`uint64_t`	Timestamp in nanoseconds
frameNumber	`uint16_t`	Frame number of the packet
lineNumber	`uint16_t`	Line number of the packet
streamID	`uint8_t`	Stream ID of the packet

NOTICE

Stream ID
The Stream ID is usually 0. Change only after consulting dSPACE.

DSTEsiError

The **`DSTEsiError`** is an enum which is returned by most of the functions of the dSPACE ESI API. Use the **`DSESI_ErrorToString`** function to convert it to a human readable string.

Error Code	Value	Description
NoError	0	No error
InternalError	1	Internal error
ControlInterfaceNotAvailable	2	Control interface not available
DataInterfaceNotAvailable	3	Data interface not available
ChannelNotAvailable	4	Channel not available
InvalidChannelHandle	5	Invalid channel handle
InvalidChannelIndex	6	Invalid channel index
InvalidTiming	7	Invalid timing
InvalidPointer	8	Invalid pointer
ControlSocketError	9	Error in control socket
DataSocketError	10	Error in data socket
Timeout	11	Timeout
InvalidInterfaceConfig	12	Invalid inputInterface configuration
ChannelAlreadyRegistered	13	Channel already registered
DataNotAvailable	14	Data not available

Functions

The following chapter presents an overview of the functions of the ESI API.

Where to go from here

Information in this section

DSESI_IsInputInterfaceAvailable

Purpose

Pings the control ip address and the data ip address of the given inputInterface. Allows the user to check if the ESI Unit is available. The ping is not sent from a specific interface, and therefore the function might indicate the ESI Unit is available, but the actual config might force an specific interface which is properly connected.

Syntax

```
DSTEsiError DSESI_IsInputInterfaceAvailable(const DSTEsiInputInterfaceT* inputInterface);
```

Parameters (In)

- **`inputInterface`**: Specifies the data IP and the control IP of the ESI Unit.

Parameters (Out)

- None

Return value

One of the error codes defined in DSTEsiError on page 16. If the interface is available, it returns **`DSTEsiError::NoError`**.

DSESI_RegisterOutputChannel

Purpose

Registers an output channel for further usage. Returns an handle which is necessary to use the other functions.

Syntax

```
DSTEsiError DSESI_RegisterOutputChannel(const DSTEsiInputInterfaceT* inputInterface,
 DSTEsiChannelNumber channelIndex, DSTEsiChannelHandle* channelHandle);
```

Parameters (In)

- **`inputInterface`**: The interface configuration
- **`channelIndex`**: The index of the channel

Parameters (Out)

- **`channelHandle`**: Channel handle

Return value

One of the error codes defined in DSTEsiError on page 16.

DSESI_UnregisterOutputChannel

Purpose

Unregisters an output channel after usage. If the channel that was unregistered was the last from that corresponding local ip/local port configuration, the socket is closed. Must be called after replay, as it frees resources and unregisteres from the ESI Unit.

Syntax

```
DSTEsiError DSESI_UnregisterOutputChannel(DSTEsiChannelHandle channelHandle);
```

Parameters (In)

- **`channelHandle`**: Channel handle

Parameters (Out)

- None

Return value

One of the error codes defined in DSTEsiError on page 16.

DSESI_IsOutputChannelAvailable

Purpose

Checks if the registered channel is actually available for replay.

Syntax

```
DSTEsiError DSESI_IsOutputChannelAvailable(DSTEsiChannelHandle channelHandle);
```

Parameters (In)

- **`channelHandle`**: Channel handle

Parameters (Out)

- None

Return value

One of the error codes defined in DSTEsiError on page 16. If the channel is available, it returns **`DSTEsiError::NoError`**.

DSESI_InitOutputChannel

Purpose

Initializes the output channel for replay. Sets the specified parameters, activates replay and status messages of the ESI Unit.

Syntax

```
DSTEsiError DSESI_InitOutputChannel(DSTEsiChannelHandle channelHandle, uint16_t width, uint16_t
 height);
```

Parameters (In)

- **`channelHandle`**: Channel handle
- **`width`**: Width of the images in the video stream in bytes.
- **`height`**: Height of the images in the video stream.

Parameters (Out)

- None

Return value

One of the error codes defined in DSTEsiError on page 16.

DSESI_SetReplayStartTime

Purpose

Sets the start time, the offset and optionally the stop time of the ESI Unit.

Syntax

```
DSTEsiError DSESI_SetReplayStartTime(DSTEsiChannelHandle channelHandle, uint64_t StartTime, uint64_t
 Offset = 0, uint64_t StopTime = 0);
```

Parameters (In)

- **`channelHandle`**: Channel handle
- **`StartTime`**: The start time in nano seconds.
- **`StopTime`**: The stop time in nano seconds. Optional. Is only send if **`StopTime > 0`**
- **`Offset`**: The offset in nano seconds. Optional. Is only send if **`Offset > 0`**

Parameters (Out)

- None

Return value

One of the error codes defined in DSTEsiError on page 16.

DSESI_SetReplayOffset

Purpose

Sets the offset of the ESI Unit.

Syntax

```
DSTEsiError DSESI_SetReplayOffset(DSTEsiChannelHandle channelHandle, uint64_t Offset);
```

Parameters (In)

- **`channelHandle`**: Channel handle
- **`Offset`**: The offset in nano seconds

Parameters (Out)

- None

Return value

One of the error codes defined in DSTEsiError on page 16.

DSESI_SetReplayStopTime

Purpose

Sets the offset of the ESI Unit.

Syntax

```
DSTEsiError DSESI_SetReplayStopTime(DSTEsiChannelHandle channelHandle, uint64_t StopTime);
```

Parameters (In)

- **`channelHandle`**: Channel handle
- **`StopTime`**: The stop time in nano seconds

Parameters (Out)

- None

Return value

One of the error codes defined in DSTEsiError on page 16.

DSESI_GetCurrentState

Purpose

Gets the current state of the ESI Unit as it is described DSTEsiStatusT on page 15.

Syntax

```
DSTEsiError DSESI_GetCurrentState(DSTEsiChannelHandle channelHandle, DSTEsiStatusT* Status);
```

Parameters (In)

- **`channelHandle`**: Channel handle

Parameters (Out)

- **`Status`**: Current ESI state

Return value

One of the error codes defined in DSTEsiError on page 16.

DSESI_GetReplayStartTime

Purpose

Gets the current replay times of the ESI Unit.

Syntax

```
DSTEsiError DSESI_GetReplayStartTime(DSTEsiChannelHandle channelHandle, uint64_t* StartTime,
 uint64_t* Offset = NULL, uint64_t* StopTime = NULL);
```

Parameters (In)

- **`channelHandle`**: Channel handle

Parameters (Out)

- **`StartTime`**: The start time in nano seconds.
- **`Offset`**: The offset in nano seconds. Optional. Is only retrieved when **`Offset != NULL`**.
- **`Stoptime`**: The stop time in nano seconds. Optional. Is only retrieved when **`Stoptime != NULL`**.

Return value

One of the error codes defined in DSTEsiError on page 16.

DSESI_GetReplayOffset

Purpose

Gets the current offset of the ESI Unit.

Syntax

```
DSTEsiError DSESI_GetReplayOffset(DSTEsiChannelHandle channelHandle, uint64_t* Offset);
```

Parameters (In)

- **`channelHandle`**: Channel handle

Parameters (Out)

- **`Offset`**: The offset in nano seconds

Return value

One of the error codes defined in DSTEsiError on page 16.

DSESI_GetReplayStopTime

Purpose

Gets the current stop time of the ESI Unit.

Syntax

```
DSTEsiError DSESI_GetReplayStopTime(DSTEsiChannelHandle channelHandle, uint64_t* StopTime);
```

Parameters (In)

- **`channelHandle`**: Channel handle

Parameters (Out)

- **`Stoptime`**: The stop time in nano seconds

Return value

One of the error codes defined in DSTEsiError on page 16.

DSESI_GetFrameSize

Purpose

Gets the frame size of the ESI Unit.

Syntax

```
DSTEsiError DSESI_GetFrameSize(DSTEsiChannelHandle channelHandle, uint16_t* width, uint16_t* height);
```

Parameters (In)

- **`channelHandle`**: Channel handle

Parameters (Out)

- **`width`**: Width of the channel
- **`height`**: Height of the channel

Return value

One of the error codes defined in DSTEsiError on page 16.

DSESI_ReplayStreamData

Purpose

Sends image data to the ESI Unit. Blocks until timeoutStream_ms is reached, if no space is available in the buffer in that time. This function is used to transmit a single line of image data. The format in the buffer is project specific and needs to be specified with dSPACE.

Syntax

```
DSTEsiError DSESI_ReplayStreamData(DSTEsiChannelHandle channelHandle, const void* packetProperties,
 const void* buffer, uint16_t bufferSize);
```

Parameters (In)

- **`channelHandle`**: Channel handle
- **`packetProperties`**: Packet properties of the data to be sent
- **`buffer`**: Pointer to the data to be transmitted
- **`bufferSize`**: Size of the buffer in bytes

Parameters (Out)

- None

Return value

One of the error codes defined in DSTEsiError on page 16. If the timeout was reached, **`DSTEsiError::Timeout`** is returned.

DSESI_ResetChannel

Purpose

Resets a channel.

Syntax

```
DSTEsiError DSESI_ResetChannel(DSTEsiChannelHandle channelHandle);
```

Parameters (In)

- **`channelHandle`**: Channel handle

Parameters (Out)

- None

Return value

One of the error codes defined in DSTEsiError on page 16.

DSESI_ClearStatusFlags

Purpose

Clears status flags.

Syntax

```
DSTEsiError DSESI_ClearStatusFlags(DSTEsiChannelHandle channelHandle);
```

Parameters (In)

- **`channelHandle`**: Channel handle

Parameters (Out)

- None

Return value

One of the error codes defined in DSTEsiError on page 16.

DSESI_ErrorToString

Purpose

Converts a given error code to a humand readable string.

Syntax

```
const char* DSESI_ErrorToString(DSTEsiError error);
```

Parameters (In)

- **`error`**: The error code returned by one of the functions listed above

Parameters (Out)

- None

Return value

A human readable string containing a description of the error.