## Book 5—System Utilities

## Part B:

# Examples of TSSA Components

TriMedia

Version 2.0 beta

# Table of Contents

# Chapter 15

# File Reader (Fread) API

## TriMedia Fread API Overview

The Fread component provides a file reader that delivers streaming data as its output. Fread continuously rereads a file by default, producing a continuous data stream like standard in. The **exolFileIO** example provides an example mechanism for reading the file a number of times. Fread accepts multi-buffered packets and sequentially fills each buffer with binary data from the file.

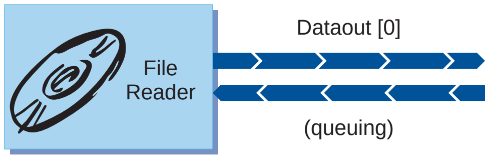

**Figure 1** Structure of the File Reader

### Inputs and Outputs

Fread takes as input the name of a file through the filename field in its instance setup structure. Its output is a stream of TSSA packets filled with binary data read from the input file. The data read from the file is not interpreted in any way by the Fread component, and is delivered to the output without any change.

The header of each packet and the number of buffers in each packet is set up by the application during **tsaDefaultInOutDescriptorCreate**. The component receiving data from Fread can interpret the data by looking at the format description in the header of each packet.

### Errors

Errors from Fread come in the form of non-zero return values from its API. In addition, Fread used in debugging mode will also use the assert mechanism to flag invalid uses of the library. Errors can result from the setup, run-time, or clean up phases. All possible errors are described in *Fread API Functions* starting on page 12.

### Progress

When Fread is used in streaming mode (OL layer), it gives the application an opportunity to stop the streaming at each End-Of-File read from the file, by calling the progress callback function. Returning a non-TMLIBAPP_OK value in the progress report allows the application to stop the Fread instance at this point, instead of allowing it to seek the beginning of the file and continue reading.

## Fread API Data Structures

This section presents the data structures used in the TriMedia Fread library. Since the Fread API is used in both the application layer and the operating system layer, each of the data structures below has a tmal and tmol counterpart.

| Name | Page |
|---|---|
| tmalFreadCapabilities_t, tmolFreadCapabilities_t | 10 |
| tmalFreadInstanceSetup_t, tmolFreadInstanceSetup_t | 11 |

## tmalFreadCapabilities_t, tmolFreadCapabilities_t

```
typedef struct {
   ptsaDefaultCapabilities_t  defaultCapabilities;
} tmalFreadCapabilities_t, *ptmalFreadCapabilities_t;

typedef  tmalFreadCapabilities_t  tmolFreadCapabilities_t;
typedef ptmalFreadCapabilities_t ptmolFreadCapabilities_t;
```

### Fields

| | |
|---|---|
| `defaultCapabilities` | Pointer to default capabilities struct (see tsa.h). |

### Description

Describes the capabilities of Fread. Fread does not have capabilities fields other than those in **tsaDefaultCapabilities_t**.

## tmalFreadInstanceSetup_t, tmolFreadInstanceSetup_t

```
typedef struct {
   ptsaDefaultInstanceSetup_t   defaultSetup;
   String                       fileName;
   Bool                         errorInsertion;
   UInt32                       errorAverageDistance;
   Int                          errorSkipByte;
} tmalFreadInstanceSetup_t, *ptmalFreadInstanceSetup_t;

typedef  tmalFreadInstanceSetup_t  tmolFreadInstanceSetup_t;
typedef ptmalFreadInstanceSetup_t ptmolFreadInstanceSetup_t;
```

### Fields

| | |
|---|---|
| `defaultSetup` | Pointer to default instance setup struct (see tsa.h). |
| `fileName` | Pathname of the file to read. |
| `errorInsertion` | turn on error insertion, this will insert random errors in the data that is send out to the data out queue. Useful for bitstream error testing. |
| `errorAverageDistance` | The average distances in bits between two errors. |
| `errorSkipByte` | Whenever a byte has this value, do not introduce any errors. Useful for omitting sync byte error insertion. |

### Description

Describes the variables used by Fread to set up. Used by the application to pass initial information to Fread.

## Fread API Functions

This section presents the functions used in the TriMedia Fread library. Since the Fread API is used in both the application layer and the operating system layer, each of the functions below, except for **tmalFreadReadBuffer**, has a tmal and tmol counterpart.

| Name | Page |
|---|---|
| tmolFreadGetCapabilities | 13 |
| tmalFreadGetCapabilities | 13 |
| tmolFreadOpen | 14 |
| tmalFreadOpen | 14 |
| tmolFreadClose | 15 |
| tmalFreadClose | 15 |
| tmolFreadGetInstanceSetup | 16 |
| tmalFreadGetInstanceSetup | 17 |
| tmolFreadInstanceSetup | 18 |
| tmalFreadInstanceSetup | 19 |
| tmolFreadStart | 20 |
| tmalFreadStart | 21 |
| tmolFreadStop | 22 |
| tmalFreadStop | 23 |
| tmalFreadReadBuffer | 24 |
| tmolFreadInstanceConfig | 25 |
| tmalFreadInstanceConfig | 26 |

## tmolFreadGetCapabilities

```
tmLibappErr_t tmolFreadGetCapabilities(
   ptmolFreadCapabilities_t  *cap
);
```

### Parameters

| | |
|---|---|
| `cap` | Pointer to the OL capabilities struct of type **tmolFreadCapabilities_t**. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |

### Description

Returns a pointer to the OL capabilities of Fread.

## tmalFreadGetCapabilities

```
tmLibappErr_t tmalFreadGetCapabilities(
   ptmalFreadCapabilities_t   *cap
);
```

### Parameters

| | |
|---|---|
| `*cap` | Pointer to the AL capabilities struct of type **tmalFreadCapabilities_t**. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |

### Description

Returns a pointer to the AL capabilities of Fread.

## tmolFreadOpen

```
tmLibappErr_t tmolFreadOpen(
   Int   *instance
);
```

### Parameters

| | |
|---|---|
| `instance` | Pointer to the OL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_MEMALLOC_FAILED` | Memory allocation for the instance variables failed. |

### Description

Allocates an OL instance of Fread for usage.

## tmalFreadOpen

```
tmLibappErr_t tmalFreadOpen(
   Int   *instance
);
```

### Parameters

| | |
|---|---|
| `instance` | Pointer to the AL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_MEMALLOC_FAILED` | Memory allocation for the instance variables failed. |

### Description

Allocates an AL instance of Fread for usage.

## tmolFreadClose

```
tmLibappErr_t tmolFreadClose(
   Int   instance
);
```

### Parameters

| | |
|---|---|
| `instance` | The OL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `FR_ERR_CLOSE_FAILED` | Stream could not be closed successfully. |

### Description

Deallocates the OL instance. **tmolFreadOpen** must have been called previously. The instance must be in the stopped state before this function completes.

## tmalFreadClose

```
tmLibappErr_t tmalFreadClose(
   Int   instance
);
```

### Parameters

| | |
|---|---|
| `instance` | The AL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `FR_ERR_CLOSE_FAILED` | Stream could not be closed successfully. |

### Description

Deallocates the AL instance. **tmalFreadOpen** must have been called previously. The instance must be in the stopped state before this function completes.

## tmolFreadGetInstanceSetup

```
tmLibappErr_t tmolFreadGetInstanceSetup(
   Int                         instance,
   ptmolFreadInstanceSetup_t  *setup
);
```

### Parameters

| | |
|---|---|
| `instance` | The OL instance. |
| `setup` | Pointer to a variable in which to return a pointer to the OL instance setup data. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |

### Description

Returns a pointer to a template OL instance setup structure in **setup**. **tmolFreadOpen** must have been called previously.

## tmalFreadGetInstanceSetup

```
tmLibappErr_t tmalFreadGetInstanceSetup(
   Int                          instance,
   ptmalFreadInstanceSetup_t  *setup
);
```

### Parameters

| | |
|---|---|
| `instance` | The AL instance. |
| `setup` | Pointer to a variable in which to return a pointer to the AL instance setup data. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |

### Description

Returns a pointer to a template AL instance setup structure in **setup**. **tmalFreadOpen** must have been called previously.

## tmolFreadInstanceSetup

```
tmLibappErr_t tmolFreadInstanceSetup(
   Int                         instance,
   tmolFreadInstanceSetup_t  *setup
);
```

### Parameters

| | |
|---|---|
| `instance` | The OL instance. |
| `setup` | Pointer to the OL instance setup struct of type **tmolFreadInstanceSetup_t**. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_INVALID_SETUP` | Setup parameters are not valid. |
| `TMLIBAPP_ERR_MEMALLOC_FAILED` | Memory allocation for the output descriptors failed. |
| `TMLIBAPP_ERR_NOT_STOPPED` | Instance is not in the stopped state. |
| `TMLIBAPP_ERR_TCREATE_FAILED` | Creation of the associated task failed. |
| `TMLIBAPP_ERR_TSUSPEND_FAILED` | Suspension of the associated task failed. |
| `FR_ERR_OPEN_FAILED` | File indicated by filename could not be opened. |
| `FR_ERR_CLOSE_FAILED` | A re-setup and the old stream could not be closed. |

### Description

Sets up the OL instance of Fread.

**tmolFreadOpen** must have been called previously. The instance must be in the stopped state.

## tmalFreadInstanceSetup

```
tmLibappErr_t tmalFreadInstanceSetup(
   Int                        instance,
   tmalFreadInstanceSetup_t  *setup
);
```

### Parameters

| | |
|---|---|
| `instance` | The AL instance. |
| `setup` | Pointer to the AL instance setup struct of type **tmalFreadInstanceSetup_t**. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_INVALID_SETUP` | Setup parameters are not valid. |
| `TMLIBAPP_ERR_NOT_STOPPED` | Instance is not in the stopped state. |
| `FR_ERR_OPEN_FAILED` | The open on the file indicated by filename failed. |
| `FR_ERR_CLOSE_FAILED` | A re-setup and the old stream could not be closed. |

### Description

Sets up the instance of Fread. **setup** includes the name of the file to be read. Fread uses the callback functions, **dataout**, **progress**, **completion**, and **control**, which can be set by the application if desired. However, it is not recommended to set dataout and control, as the default version works with TSSA.

**tmalFreadOpen** must have been called previously. The instance must be in the stopped state.

## tmolFreadStart

```
tmLibappErr_t tmolFreadStart(
   Int   instance
);
```

### Parameters

| | |
|---|---|
| `instance` | The OL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_NOT_SETUP` | Instance has not been previously set up. |
| `TMLIBAPP_ERR_ALREADY_STARTED` | Instance has previously been started. |
| `TMLIBAPP_ERR_TSTART_FAILED` | Start of the associated task failed. |
| `TMLIBAPP_ERR_TRESUME_FAILED` | Resumption of the associated task failed. |

### Description

Starts data streaming by calling tsaDefaultStart. Continuously reads from file into data packets to send out.

**tmolFreadOpen** must have been called previously and **tmolFreadInstanceSetup** must have been called previously.

## tmalFreadStart

```
tmLibappErr_t tmalFreadStart(
   Int   instance
);
```

### Parameters

| | |
|---|---|
| `instance` | The AL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_NOT_SETUP` | Instance has not been set up previously. |
| `TMLIBAPP_ERR_ALREADY_STARTED` | Instance has already been started. |
| `FR_ERR_READ_FAILED` | Read on the file failed. |

### Description

Starts data streaming. Automatically loops back to beginning when reached EOF. Calls progress functions at each EOF read with loop count as progress code. If progress function returns error, break out of processing loop. Otherwise, continues to loop back and behave like standard input. After breaking out of processing loop, expel packet if holding one. Calls completion function after expelling packets.

**tmalFreadOpen** must have been called previously and **tmalFreadInstanceSetup** must have been called previously.

## tmolFreadStop

```
tmLibappErr_t tmolFreadStop(
   Int   instance
);
```

### Parameters

| | |
|---|---|
| `instance` | The OL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_NOT_SETUP` | Instance has not been previously setup. |
| `TMLIBAPP_ERR_NOT_STARTED` | Instance has not been previously started. |
| `TMLIBAPP_ERR_TSUSPEND_FAILED` | Suspension of the associated task failed. |

### Description

Stops data streaming by calling **tsaDefaultStop**. Causes instance to fall out of its processing loop in **tmalFreadStart**.

**tmolFreadOpen** must have been called previously and **tmolFreadInstanceSetup** must have been called previously.

## tmalFreadStop

```
tmLibappErr_t tmalFreadStop(
   Int   instance
);
```

### Parameters

| | |
|---|---|
| `instance` | The AL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_NOT_SETUP` | Instance has not been previously setup. |

### Description

Stops data streaming. Causes the instance to fall out of its processing loop in **tmalFread-Start**.

**tmalFreadOpen** must have been called previously and **tmalFreadInstanceSetup** must have been called previously.

## tmalFreadReadBuffer

```
tmLibappErr_t tmalFreadReadBuffer(
   Int             instance,
   tmAvPacket_t   *packet
);
```

### Parameters

| | |
|---|---|
| `instance` | The AL instance. |
| `packet` | Pointer to the data packet. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_NOT_SETUP` | Instance has not been setup previously. |
| `TMLIBAPP_ERR_NULL_PACKET` | The packet pointer is null. |
| `FR_ERR_READ_FAILED` | Read on the file failed. |
| `FR_ERR_READ_EOF` | The end of file has been read. |

### Description

Sequentially reads data into the packet data buffers. This function represents the push model supported by **Fread** and is only available in the AL layer.

**tmalFreadOpen** must have been called previously and **tmalFreadInstanceSetup** must have been called previously.

## tmolFreadInstanceConfig

```
tmLibappErr_t tmolFreadInstanceConfig(
   Int                 instance,
   UInt32              flags,
   ptsaControlArgs_t   args
);
```

### Parameters

| | |
|---|---|
| instance | The OL instance. |
| flags | For future development. |
| args | Command arguments. |

### Return Codes

| | |
|---|---|
| TMLIBAPP_OK | Success. |
| TMLIBAPP_ERR_INVALID_INSTANCE | Not a valid instance. |
| TMLIBAPP_ERR_NOT_SETUP | Instance has not been setup previously. |
| TMLIBAPP_QUEUE_EMPTY | Timed out while waiting for response from component. |

### Description

Configures the instance. Calls tsaDefaultInstanceConfig to put a command packet on the control queue. The commands, **tmalFreadCommands_t**, are enumerated in tmalFread.h.

**tmolFreadOpen** must have been called previously.

## tmalFreadInstanceConfig

```
tmLibappErr_t tmalFreadInstanceConfig(
   Int                instance,
   ptsaControlArgs_t  args
);
```

### Parameters

| | |
|---|---|
| `instance` | The AL instance. |
| `args` | Command arguments. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_NOT_SETUP` | Instance has not been setup previously. |
| `FR_ERR_SEEK_FAILED` | Seek of beginning of file failed. |
| `FR_ERR_SEEK_OUT_OF_RANGE` | Attempted seek beyond end of file or before beginning of file. |
| `FR_ERR_OPEN_FAILED` | Open file failed. |
| `FR_ERR_CLOSE_FAILED` | Close file failed. |
| `TMLIBAPP_ERR_INVALID_COMMAND` | Unknown or invalid command. |

### Description

Configures the instance.It should be called when Dataout(GetEmpty) is called with the flag, **tsaDataoutCheckControl**. When it is called with the **tmalFreadRewind** command, it will seek to beginning of file. When called with the **tmalFreadNewFile** command, it will close current file and open a new file to read from. The commands, **tmalFreadCommands_t**, are enumerated in tmalFread.h.

**tmalFreadOpen** must have been called previously.

## Chapter 16

# File Writer (Fwrite) API

## TriMedia Fwrite API Overview

The File Writer component accepts streaming data as its input and writes the binary data to file. Fwrite continuously writes to a file, mimicking the behavior of standard output as redirected to a file. The exolFileIO sample program shows its usage with Fread. Fwrite accepts multi-buffered packets and writes each buffer sequentially to the file.

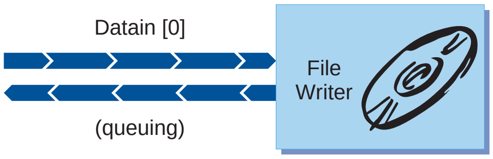

**Figure 2** Structure of the File Writer

### Inputs and Outputs

Fwrite takes a stream of TSSA packets as input filled with binary data. It writes the data into a file specified by the filename field in its instance setup structure. The data received from its input is not interpreted by the Fwrite component, it is written to the file without any change. The header and the number of buffers in each packet is set up by the application during **tsaDefaultInOutDescriptorCreate**. The format of the data is stored in the header of each packet.

### Errors

Errors from Fwrite come in the form of non-zero return values from its API. In addition, Fwrite used in debugging mode will also use the assert mechanism to flag invalid uses of the library. Errors can result from the setup, run-time, or clean up phases. All possible errors are described in *Fwrite API Functions* starting on page 32.

## Fwrite API Data Structures

This section presents the data structures used in the TriMedia Fwrite library. Since the Fwrite API is used in both the application layer and the operating system layer, each of the data structures below has a tmal and tmol counterpart.

| Name | Page |
|---|---|
| tmalFwriteCapabilities_t | 30 |
| tmolFwriteCapabilities_t | 30 |
| tmalFwriteInstanceSetup_t | 31 |
| tmolFwriteInstanceSetup_t | 31 |

## tmalFwriteCapabilities_t

```
typedef struct {
   ptsaDefaultCapabilities_t   defaultCapabilities;
} tmalFwriteCapabilities_t, *ptmalFwriteCapabilities_t;
```

## tmolFwriteCapabilities_t

```
typedef struct {
   ptsaDefaultCapabilities_t   defaultCapabilities;
} tmolFwriteCapabilities_t, *ptmolFwriteCapabilities_t;
```

### Fields

| | |
|---|---|
| `defaultCapabilities` | Pointer to default capabilities struct (see tsa.h). |

### Description

Describes the capabilities of Fwrite. Fwrite does not have capabilities fields other than those in **tsaDefaultCapabilities_t**.

## tmalFwriteInstanceSetup_t

```
typedef struct {
   ptsaDefaultInstanceSetup_t   defaultSetup;
   String                       fileName;
} tmalFwriteInstanceSetup_t, *ptmalFwriteInstanceSetup_t;
```

## tmolFwriteInstanceSetup_t

```
typedef struct {
   ptsaDefaultInstanceSetup_t   defaultSetup;
   String                       fileName;
} tmolFwriteInstanceSetup_t, *ptmolFwriteInstanceSetup_t;
```

### Fields

| | |
|---|---|
| `defaultSetup` | Pointer to default instance setup struct (see tsa.h) |
| `fileName` | Name (path) of the file to which to write. |

### Description

Describes the variables used by Fwrite to set up instance. Used by the application to pass initial information to Fwrite.

## Fwrite API Functions

This section presents the functions used in the TriMedia Fwrite library. Since the Fwrite API is used in both the application layer and the operating system layer, each of the functions below, except for **tmalFwriteGetInstanceSetup** and **tmalFwriteWriteBuffer**, has a tmal and tmol counterpart.

| Name | Page |
|---|---|
| tmolFwriteGetCapabilities | 33 |
| tmalFwriteGetCapabilities | 33 |
| tmolFwriteOpen | 34 |
| tmalFwriteOpen | 34 |
| tmolFwriteClose | 35 |
| tmalFwriteClose | 35 |
| tmolFwriteGetInstanceSetup | 36 |
| tmalFwriteGetInstanceSetup | 37 |
| tmolFwriteInstanceSetup | 38 |
| tmalFwriteInstanceSetup | 39 |
| tmolFwriteStart | 40 |
| tmalFwriteStart | 41 |
| tmolFwriteStop | 42 |
| tmalFwriteStop | 43 |
| tmalFwriteWriteBuffer | 44 |
| tmolFwriteInstanceConfig | 45 |
| tmalFwriteInstanceConfig | 46 |

## tmolFwriteGetCapabilities

```
tmLibappErr_t tmolFwriteGetCapabilities(
   ptmolFwriteCapabilities_t   *cap
);
```

### Parameters

| | |
|---|---|
| `cap` | Pointer to the OL capabilities struct of type **tmolFwriteCapabilities_t.** |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |

### Description

Returns a pointer to the OL capabilities of Fwrite.

## tmalFwriteGetCapabilities

```
tmLibappErr_t tmalFwriteGetCapabilities(
   ptmalFwriteCapabilities_t   *cap
);
```

### Parameters

| | |
|---|---|
| `cap` | Pointer to the AL capabilities struct of type **tmalFwriteCapabilities_t** |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |

### Description

Returns a pointer to the AL capabilities of Fwrite.

## tmolFwriteOpen

```
tmLibappErr_t tmolFwriteOpen(
   Int   *instance
);
```

### Parameters

| | |
|---|---|
| `instance` | Pointer to the OL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_MEMALLOC_FAILED` | Memory allocation for the instance variables failed. |

### Description

Allocates an OL instance of Fwrite for usage.

## tmalFwriteOpen

```
tmLibappErr_t tmalFwriteOpen(
   Int   *instance
);
```

### Parameters

| | |
|---|---|
| `instance` | Pointer to the AL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_MEMALLOC_FAILED` | Memory allocation for the instance variables failed. |

### Description

Allocates an AL instance of Fwrite for usage.

## tmolFwriteClose

```
tmLibappErr_t tmolFwriteClose(
   Int   instance
);
```

### Parameters

| | |
|---|---|
| `instance` | The OL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Instance is not a valid instance. |
| `FW_ERR_CLOSE_FAILED` | Stream could not be closed successfully. |

### Description

Deallocates the OL instance. **tmalFwriteOpen** must have been called previously. The instance must be in the stopped state before this function completes.

## tmalFwriteClose

```
tmLibappErr_t tmalFwriteClose(
   Int   instance
);
```

### Parameters

| | |
|---|---|
| `instance` | The AL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `FW_ERR_CLOSE_FAILED` | Stream could not be closed successfully. |

### Description

Deallocates the AL instance. **tmalFwriteOpen** must have been called previously. The instance must be in the stopped state before this function completes.

## tmolFwriteGetInstanceSetup

```
tmLibappErr_t tmolFwriteGetInstanceSetup(
   Int                          instance,
   ptmolFwriteInstanceSetup_t  *setup
);
```

### Parameters

| | |
|---|---|
| `instance` | The OL instance. |
| `setup` | Pointer to a variable in which to return a pointer to the OL instance setup data. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |

### Description

Returns a pointer to a template OL instance setup structure in **setup**. **tmolFwriteOpen** must have been called previously.

## tmalFwriteGetInstanceSetup

```
tmLibappErr_t tmalFwriteGetInstanceSetup(
   Int                          instance,
   ptmalFwriteInstanceSetup_t  *setup
);
```

### Parameters

| | |
|---|---|
| `instance` | The AL instance. |
| `setup` | Pointer to a variable in which to return a pointer to the AL instance setup data. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |

### Description

Returns a pointer to a template AL instance setup structure in **setup**. **tmalFwriteOpen** must have been called previously.

## tmolFwriteInstanceSetup

```
tmLibappErr_t tmolFwriteInstanceSetup(
   Int                         instance,
   tmolFwriteInstanceSetup_t  *setup
);
```

### Parameters

| | |
|---|---|
| `instance` | The OL instance. |
| `setup` | Pointer to the OL instance setup struct of type **tmolFwriteInstanceSetup_t**. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_INVALID_SETUP` | Setup parameters are not valid. |
| `TMLIBAPP_ERR_MEMALLOC_FAILED` | Memory allocation for the input descriptors failed. |
| `TMLIBAPP_ERR_NOT_STOPPED` | Instance was not in the stopped state. |
| `TMLIBAPP_ERR_TCREATE_FAILED` | Creation of the associated task failed. |
| `TMLIBAPP_ERR_TSUSPEND_FAILED` | Suspension of the associated task failed. |
| `FW_ERR_OPEN_FAILED` | The open on the file indicated by filename failed. |
| `FW_ERR_CLOSE_FAILED` | A re-setup and the old stream could not be closed. |

### Description

Sets up the instance of Fwrite. **tmolFwriteOpen** must have been called previously. The instance must be in the stopped state.

## tmalFwriteInstanceSetup

```
tmLibappErr_t tmalFwriteInstanceSetup(
   Int                        instance,
   tmalFwriteInstanceSetup_t  *setup
);
```

### Parameters

| | |
|---|---|
| `instance` | The AL instance. |
| `setup` | Pointer to the AL instance setup struct of type **tmalFwriteInstanceSetup_t**. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_INVALID_SETUP` | Setup parameters are not valid. |
| `TMLIBAPP_ERR_NOT_STOPPED` | Instance was not in the stopped state. |
| `FW_ERR_OPEN_FAILED` | The open on the file indicated by filename failed. |
| `FW_ERR_CLOSE_FAILED` | A re-setup and the old stream could not be closed. |

### Description

Sets up the instance of Fwrite. **setup** includes the name of the file to write to. Fwrite uses the callback functions, **datain**, **completion**, and **control**, which can be set by the application if desired. It is not recommended however, to set datain and control, as the default version works with TSSA.

**tmalFwriteOpen** must have been called previously. The instance must be in the stopped state.

## tmolFwriteStart

```
tmLibappErr_t tmolFwriteStart(
   Int  instance
);
```

### Parameters

| | |
|---|---|
| `instance` | The OL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_NOT_SETUP` | Instance has not been set up previously. |
| `TMLIBAPP_ERR_ALREADY_STARTED` | Instance has already been started. |
| `TMLIBAPP_ERR_TSTART_FAILED` | Start of the associated task failed. |
| `TMLIBAPP_ERR_TRESUME_FAILED` | Resumption of the associated task failed. |

### Description

Starts data streaming by calling tsaDefaultStart. Continuously write to file with data from packets received.

**tmolFwriteOpen** must have been called previously. **tmolFwriteInstanceSetup** must have been called previously.

## tmalFwriteStart

```
tmLibappErr_t tmalFwriteStart(
   Int  instance
);
```

### Parameters

| | |
|---|---|
| `instance` | The AL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_NOT_SETUP` | Instance has not been set up previously. |
| `TMLIBAPP_ERR_ALREADY_STARTED` | Instance has already been started. |
| `FW_ERR_WRITE_FAILED` | Write to the file failed. |

### Description

Starts data streaming. Continuously write to file until an error or stop command has been encountered. After breaking out of processing loop, it expels packet if holding one. After expelling packets, completion function is called.

## tmolFwriteStop

```
tmLibappErr_t tmolFwriteStop(
   Int  instance
);
```

### Parameters

| | |
|---|---|
| `instance` | The OL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_NOT_SETUP` | Instance has not been set up previously. |
| `TMLIBAPP_ERR_NOT_STARTED` | Instance has not previously been started. |
| `TMLIBAPP_ERR_TSUSPEND_FAILED` | Suspension of the associated task failed. |

### Description

Stops data streaming by calling tsaDefaultStop. It Causes instance to fall out of its processing loop in tmalFwriteStart.

**tmolFwriteOpen** must have been called previously. **tmolFwriteInstanceSetup** must have been called previously.

## tmalFwriteStop

```
tmLibappErr_t tmalFwriteStop(
   Int  instance
);
```

### Parameters

| | |
|---|---|
| `instance` | The AL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_NOT_SETUP` | Instance has not previously been set up. |
| `TMLIBAPP_ERR_ALREADY_STOPPED` | Instance has already been stopped. |

### Description

Stops data streaming. It causes the instance to fall out of its processing loop in **tmal-FwriteStart**.

**tmolFwriteOpen** must have been called previously. **tmolFwriteInstanceSetup** must have been called previously.

## tmalFwriteWriteBuffer

```
tmLibappErr_t tmalFwriteWriteBuffer(
   Int            instance,
   tmAvPacket_t   *packet
);
```

### Parameters

| | |
|---|---|
| `instance` | The AL Instance. |
| `packet` | Pointer to the data packet. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_NOT_SETUP` | Instance has not been setup previously. |
| `FW_ERR_WRITE_FAILED` | Write to the file failed. |

### Description

Sequentially writes data from the packet data buffers into file. This function represents the push model supported by Fwrite and is only available in the AL layer.

**tmolFwriteOpen** must have been called previously. **tmolFwriteInstanceSetup** must have been called previously.

## tmolFwriteInstanceConfig

```
tmLibappErr_t tmalFwriteInstanceConfig(
   Int                 instance,
   UInt32              flags,
   ptsaControlArgs_t   args
);
```

### Parameters

| | |
|---|---|
| `instance` | The OL instance. |
| `flags` | For future development. |
| `args` | Command arguments. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_NOT_SETUP` | Instance has not been setup previously. |
| `TMLIBAPP_QUEUE_EMPTY` | Timed out while waiting for response from component. |

### Description

Configures the instance. Calls tsaDefaultInstanceConfig to put a command packet on the control queue. The commands, **tmalFwriteCommands_t**, are enumerated in tmalFwrite.h.

**tmolFwriteOpen** must have been called previously.

## tmalFwriteInstanceConfig

```
tmLibappErr_t tmolFwriteInstanceConfig(
   Int                 instance,
   ptsaControlArgs_t   args
);
```

### Parameters

| | |
|---|---|
| `instance` | The AL instance. |
| `args` | Command arguments. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_NOT_SETUP` | Instance has not been set up previously. |
| `FW_ERR_SEEK_FAILED` | Seek to beginning of file failed. |
| `FW_ERR_OPEN_FAILED` | Open file failed. |
| `FW_ERR_CLOSE_FAILED` | Close file failed. |
| `FW_ERR_UNKNOWN_COMMAND` | Unknown or invalid command. |

### Description

Configures the instance. Calls Datain(GetFull) when called with the, **tsaDatainCheckControl**. When called with the **tmalFwriteRewind** command, it will seek to beginning of file and overwrite data already in the file. When called with the **tmalFwriteNewFile** command, will close current file and open a new file to write to. The commands, **tmalFwriteCommands_t**, are enumerated in tmalFwrite.h.

**tmolFwriteOpen** must have been called previously.

# Chapter 17

# CopyIO API

## Overview

The CopyIO component provides the utility of copying packets from its input to its output with a possible delay measured in ticks. It can be used as a template of a streaming, task-based component with both input and output. On each packet received, CopyIO copies each field in the packet's header (except the packet ID and format), all other fields in the packet, and the contents of the packet's buffers. It then waits the specified amount of delay before sending it to its output.

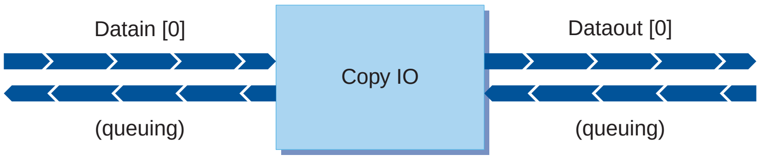

**Figure 3** Structure of CopyIO

### Using CopyIO as a Template

To develop a new component based on the copier as a template, start by copying the copier files into a local directory. You need tmalCopyIO.c, tmalCopyIO.h, tmolCopyIO.c, and tmolCopyIO.h. Then rename the files to reflect the purpose of your new component and do a search and replace of all instances of CopyIO. Be sure to get the **#define**s and **#ifdef**s that prevent double inclusion of header files. Now you can start to construct an application and makefile to test your new component. The sample programs exolCopyIO or exolCopyAudio are each reasonable starting points.

### Inputs and Outputs

CopyIO takes a stream of TSSA packets as input and produces a stream of TSSA packets as output with the same data content as the input. The packet data is not interpreted in any way by the CopyIO, but is delivered to the output with any change. The header of each packet and the number of buffers in each packet is set up by the application during **tsaDefaultInOutDescriptorCreate**. The component receiving data from CopyIO can interpret the data by looking at the format description in the header of each packet.

### Errors

Errors from CopyIO comes in the form of non-zero return values from its API. In addition, CopyIO used in debugging mode will also use the assert mechanism to flag invalid uses of the library. Errors can result from the set up, run-time, or clean up phases. All possible errors are described in *Fread API Functions* starting on page 12.

## CopyIO API Data Structures

This section describes the data structures used in the TriMedia CopyIO library. Since the CopyIO API is used in both the application layer and the operating system layer, each of the data structures below has a tmal and tmol counterpart.

| Name | Page |
|---|---|
| tmolCopyIOCapabilities_t | 50 |
| tmalCopyIOCapabilities_t | 50 |
| tmolCopyIOInstanceSetup_t | 51 |
| tmalCopyIOInstanceSetup_t | 51 |

## tmolCopyIOCapabilities_t

```
typedef struct {
   ptsaDefaultCapabilities_t   defaultCapabilities;
} tmolCopyIOCapabilities_t, *ptmolCopyIOCapabilities_t;
```

## tmalCopyIOCapabilities_t

```
typedef struct {
   ptsaDefaultCapabilities_t   defaultCapabilities;
} tmalCopyIOCapabilities_t,   *ptmalCopyIOCapabilities_t;
```

### Fields

| | |
|---|---|
| `defaultCapabilities` | Pointer to default capabilities struct (see **tsa.h**). |

### Description

Describes the capabilities of CopyIO. CopyIO does not have capabilities fields other than those in **tsaDefaultCapabilities_t**.

## tmolCopyIOInstanceSetup_t

```
typedef struct {
   ptsaDefaultInstanceSetup_t  defaultSetup;
   UInt32                      delay;
   tsaTimSleepFunc_t           TimSleep;
} tmolCopyIOInstanceSetup_t, *ptmolCopyIOInstanceSetup_t;
```

## tmalCopyIOInstanceSetup_t

```
typedef struct {
   ptsaDefaultInstanceSetup_t  defaultSetup;
   UInt32                      delay;
   tsaTimSleepFunc_t           TimSleep;
} tmalCopyIOInstanceSetup_t, *ptmalCopyIOInstanceSetup_t;
```

### Fields

| | |
|---|---|
| `defaultSetup` | Pointer to default instance setup struct (see **tsa.h**). |
| `delay` | The delay in number of ticks. |
| `TimSleep` | Function called to delay. Takes the number of ticks as argument. |

### Description

Describes the variables used by CopyIO to set up. Used by the application to pass initial information to CopyIO. When the application uses the OL layer, TimSleep is set to tsa-DefaultTimSleep, if the application did not provide it. When the application uses the AL layer, it should pass a pointer to TimSleep function if is wants a delay. Otherwise, there will be no delay in CopyIO.

## CopyIO API Functions

This section describes the functions used in the TriMedia CopyIO library. Since the CopyIO API is used in both the application layer and the operating system layer, each of the functions below, except for **tmalCopyIOCopyPacket**, has a tmal and a tmol counterpart.)

| Name | Page |
|---|---|
| tmolCopyIOGetCapabilities | 53 |
| tmalCopyIOGetCapabilities | 53 |
| tmolCopyIOOpen | 54 |
| tmalCopyIOOpen | 54 |
| tmolCopyIOClose | 55 |
| tmalCopyIOClose | 55 |
| tmolCopyIOGetInstanceSetup | 56 |
| tmalCopyIOGetInstanceSetup | 57 |
| tmolCopyIOInstanceSetup | 58 |
| tmalCopyIOInstanceSetup | 59 |
| tmolCopyIOStart | 60 |
| tmalCopyIOStart | 61 |
| tmolCopyIOStop | 62 |
| tmalCopyIOStop | 63 |
| tmalCopyIOCopyPacket | 64 |
| tmolCopyIOInstanceConfig | 65 |
| tmalCopyIOInstanceConfig | 66 |

## tmolCopyIOGetCapabilities

```
tmLibappErr_t tmolCopyIOGetCapabilities(
   ptmolCopyIOCapabilities_t   *cap
);
```

### Parameters

| | |
|---|---|
| `cap` | Pointer to the OL capabilities struct of type **tmolCopyIOCapabilities_t**. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |

### Description

Returns a pointer to the OL capabilities of CopyIO.

## tmalCopyIOGetCapabilities

```
tmLibappErr_t tmalCopyIOGetCapabilities(
   ptmalCopyIOCapabilities_t   *cap
);
```

### Parameters

| | |
|---|---|
| `cap` | Pointer to the AL capabilities struct of type **tmalCopyIOCapabilities_t**. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |

### Description

Returns a pointer to the AL capabilities of CopyIO.

## tmolCopyIOOpen

```
tmLibappErr_t tmolCopyIOOpen(
   Int   *instance
);
```

### Parameters

| | |
|---|---|
| `instance` | Pointer to the OL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_MEMALLOC_FAILED` | Memory allocation for the instance variables failed. |

### Description

Allocates an OL instance of CopyIO for usage.

## tmalCopyIOOpen

```
tmLibappErr_t tmalCopyIOOpen(
   Int   *instance
);
```

### Parameters

| | |
|---|---|
| `instance` | Pointer to the AL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_MEMALLOC_FAILED` | Memory allocation for the instance variables failed. |

### Description

Allocates an AL instance of CopyIO for usage.

## tmolCopyIOClose

```
tmLibappErr_t tmolCopyIOClose(
   Int   instance
);
```

### Parameters

| | |
|---|---|
| `instance` | The OL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |

### Description

Deallocates the OL instance. **tmolCopyIOOpen** must have been called previously. The instance must be in the stopped state before this function completes.

## tmalCopyIOClose

```
tmLibappErr_t tmalCopyIOClose(
   Int   *instance
);
```

### Parameters

| | |
|---|---|
| `instance` | The AL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |

### Description

Deallocates the AL instance. **tmolCopyIOOpen** must have been called previously. The instance must be in the stopped state before this function completes.

## tmolCopyIOGetInstanceSetup

```
tmLibappErr_t tmolCopyIOGetInstanceSetup(
   Int                         instance,
   tmolCopyIOInstanceSetup_t  *setup
);
```

### Parameters

| | |
|---|---|
| `instance` | The OL instance. |
| `setup` | Pointer to the OL instance setup struct of type **tmolCopyIOInstanceSetup_t**. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |

### Description

Returns a template OL instance setup struct of type **tmolCopyIOInstanceSetup_t** in parameter **setup**. **tmolCopyIOOpen** must have been called previously.

## tmalCopyIOGetInstanceSetup

```
tmLibappErr_t tmalCopyIOGetInstanceSetup(
   Int                        instance,
   tmalCopyIOInstanceSetup_t  *setup
);
```

### Parameters

| | |
|---|---|
| `instance` | The AL instance. |
| `setup` | Pointer to the AL instance setup struct of type **tmalCopyIOInstanceSetup_t**. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |

### Description

Returns a template AL instance setup struct of type **tmalCopyIOInstanceSetup_t** in parameter **setup**. **tmolCopyIOOpen** must have been called previously.

## tmolCopyIOInstanceSetup

```
tmLibappErr_t tmolCopyIOInstanceSetup(
   Int                        instance,
   tmolCopyIOInstanceSetup_t  *setup
);
```

### Parameters

| | |
|---|---|
| `instance` | The OL instance. |
| `setup` | Pointer to the OL instance setup struct of type **tmolCopyIOInstanceSetup_t**. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_INVALID_SETUP` | Setup parameters are not valid. |
| `TMLIBAPP_ERR_MEMALLOC_FAILED` | Memory allocation for the input or output descriptors failed. |
| `TMLIBAPP_ERR_NOT_STOPPED` | Instance is not in the stopped state. |
| `TMLIBAPP_ERR_TCREATE_FAILED` | Creation of the associated task failed. |
| `TMLIBAPP_ERR_TSUSPEND_FAILED` | Suspension of the associated task failed. |

### Description

Sets up the OL instance of CopyIO. **tmolCopyIOOpen** must have been called previously. The instance must be in the stopped state.

## tmalCopyIOInstanceSetup

```
tmLibappErr_t tmalCopyIOInstanceSetup(
   Int                        instance,
   tmalCopyIOInstanceSetup_t  *setup
);
```

### Parameters

| | |
|---|---|
| `instance` | The AL instance. |
| `setup` | Pointer to the AL instance setup struct of type**tmalCopyIOInstanceSetup_t**. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_INVALID_SETUP` | Setup parameters are not valid. |
| `TMLIBAPP_ERR_NOT_STOPPED` | Instance is not in the stopped state. |

### Description

Sets up the AL instance of CopyIO. **setup** includes the delay and TimSleep function if provided. CopyIO uses the callback functions, **datain**, **dataout**, **completion**, and **control**, which can be set by the application if desired. However, it is not recommended to set datain, dataout, and control, since the default version works with TSSA.

**tmolCopyIOOpen** must have been called previously. The instance must be in the stopped state.

## tmolCopyIOStart

```
tmLibappErr_t tmolCopyIOStart(
   Int   instance
);
```

### Parameters

| | |
|---|---|
| `instance` | The OL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_NOT_SETUP` | Instance has not been set up previously. |
| `TMLIBAPP_ERR_ALREADY_STARTED` | Instance has already been started. |
| `TMLIBAPP_ERR_TSTART_FAILED` | Start of the associated task failed. |
| `TMLIBAPP_ERR_TRESUME_FAILED` | Resumption of the associated task failed. |

### Description

Starts data streaming by calling tsaDefaultStart. **tmolCopyIOOpen** must have been called previously. **tmolCopyIOInstanceSetup** must have been called previously.

## tmalCopyIOStart

```
tmLibappErr_t tmalCopyIOStart(
   Int   instance
);
```

### Parameters

| | |
|---|---|
| `instance` | The AL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_NOT_SETUP` | Instance has not been set up previously. |
| `TMLIBAPP_ERR_ALREADY_STARTED` | Instance has already been started. |

### Description

Starts data streaming. It continuously gets input and output packets and copy contents of the input packet to the output packet, with a delay of a set number of ticks before sending output packet to output full queue. If TimSleep is Null, then there is no delay.

**tmolCopyIOOpen** must have been called previously. **tmolCopyIOInstanceSetup** must have been called previously.

## tmolCopyIOStop

```
tmLibappErr_t tmolCopyIOStop(
   Int   instance
);
```

### Parameters

| | |
|---|---|
| `instance` | The OL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_NOT_SETUP` | Instance has not been previously set up. |
| `TMLIBAPP_ERR_NOT_STARTED` | Instance has not been previously started. |
| `TMLIBAPP_ERR_TSUSPEND_FAILED` | Suspension of the associated task failed. |

### Description

Stops data streaming by calling tsaDefaultStop. Causes instance to fall out of its processing loop in tmalCopyIOStart. **tmolCopyIOOpen** must have been called previously. **tmolCopyIOInstanceSetup** must have been called previously.

## tmalCopyIOStop

```
tmLibappErr_t tmalCopyIOStop(
   Int   instance
);
```

### Parameters

| | |
|---|---|
| `instance` | The AL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_NOT_SETUP` | Instance has not been previously set up. |
| `TMLIBAPP_ERR_ALREADY_STOPPED` | Instance has already been stopped. |

### Description

Stops data streaming. It causes instance to fall out of its processing loop in tmalCopy-IOStart. **tmolCopyIOOpen** must have been called previously. **tmolCopyIOInstanceSetup** must have been called previously.

## tmalCopyIOCopyPacket

```
tmLibappErr_t tmalCopyIOCopyPacket(
   Int             instance,
   tmAvPacket_t   *inpacket,
   tmAvPacket_t   *outpacket
);
```

### Parameters

| | |
|---|---|
| `instance` | The AL instance. |
| `inpacket` | Pointer to packet to copy from. |
| `outpacket` | Pointer to packet to copy to. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `CP_ERR_ALLOCATED_BUFFERS` | **allocatedBuffers** not the same. |
| `CP_ERR_BUFSIZE` | A corresponding **bufSize** of a buffer descriptor of the packets not the same. |

### Description

Copies each field of inpacket's header, other fields of inpacket, and inpacket's buffer contents to outpacket. It fills all fields of outpacket's header, other fields of outpackets, and its buffer contents. This function represents the push model supported by CopyIO and is only available in the AL layer.

**tmolCopyIOOpen** must have been called previously. **tmolCopyIOInstanceSetup** must have been called previously. **inpacket** and **outpacket** must have the same number of allocated buffers. Each data buffers of inpacket and outpacket must be the same size.

## tmolCopyIOInstanceConfig

```
tmLibappErr_t tmalCopyIOInstanceConfig(
   Int                 instance,
   UInt32              flags,
   ptsaControlArgs_t   args
);
```

### Parameters

| | |
|---|---|
| `instance` | The OL instance. |
| `flags` | For future development. |
| `args` | Command arguments. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_NOT_SETUP` | Instance has not been setup previously. |
| `TMLIBAPP_QUEUE_EMPTY` | Timed out while waiting for response from component. |

### Description

Configures the instance. It calls **tsaDefaultInstanceConfig** to put a command packet on the control queue. The commands, **tmalCopyIOCommands_t**, are enumerated in tmalCopyIO.h. **tmolCopyIOOpen** must have been called previously.

## tmalCopyIOInstanceConfig

```
tmLibappErr_t tmalCopyIOInstanceConfig(
   Int                 instance,
   ptsaControlArgs_t   cmdArgs
);
```

### Parameters

| | |
|---|---|
| `instance` | The AL instance. |
| `args` | Command arguments. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_NOT_SETUP` | Instance has not been setup previously. |
| `CP_ERR_UNKNOWN_COMMAND` | Unknown or invalid command. |

### Description

Configures the instance. Called when Datain(GetFull) and Dataout(GetEmpty) are called with the flags, **tsaDatainCheckControl** and **tsaDataoutCheckControl**, respectively. When called with the **tmalCopyIOChangeDelay** command, will set the delay to the value in **args–>parameter**. The commands,tm**alCopyIOCommands_t**, are enumerated in tmal-CopyIO.h. **tmolCopyIOOpen** must have been called previously.

# Chapter 18

# CopyInPlace API

## Overview

The CopyInPlace component provides the utility of copying packets from its input to its output with a possible delay measured in ticks. It is derived from the TriMedia CopyIO component with the enhancement of using the TSSA ‘in place’ mechanism. It can be used as a template of an ‘in place’, streaming, task-based component. Unlike CopyIO, CopyInPlace does not actually copy data. It is designed to allow a system designer to process or examine data without moving it, that is, in place.

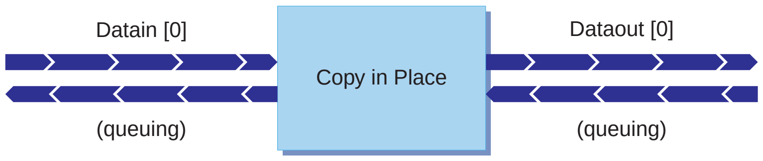

**Figure 4** Structure of CopyInPlace

### Using CopyInPlace as a Template

To develop a new component based on the copier as a template, start by copying the copier files into a local directory. You need tmalCopyInPlace.c, tmalCopyInPlace.h, tmolCopyInPlace.c, and tmolCopyInPlace.h. Then rename the files to reflect the purpose of your new component and do a search and replace of all instances of CopyInPlace. Be sure to get the **#defines** and **#ifdefs** that prevent double inclusion of the header files. Now you can start to construct an application and makefile to test your new component.

### Inputs and Outputs

CopyInPlace takes a stream of TSSA packets as input and produces a stream of TSSA packets as output with the same data content as the input. The packet data is not interpreted in any way by the CopyInPlace, but delivered to the output with any change. The header of each packet and the number of buffers in each packet is set up by the application during **tsaDefaultInOutDescriptorCreate**. The component receiving data from CopyInPlace can interpret the data by looking at the format description in the header of each packet.

### Errors

Errors from CopyInPlace comes in the form of non-zero return values from its API. In addition, CopyInPlace used in debugging mode will also use the assert mechanism to flag invalid uses of the library. Errors can result from the setup, run-time, or clean up phases. All possible errors are described in *CopyInPlace API Functions* starting on page 72.

## CopyInPlace API Data Structures

This section describes the data structures used in the TriMedia CopyInPlace library. Since the CopyInPlace API is used in both the application layer and the operating system layer, each of the data structures below has a tmal and tmol counterpart.

| Name | Page |
|---|---|
| tmolCopyInPlaceCapabilities_t | 70 |
| tmalCopyInPlaceCapabilities_t | 70 |
| tmolCopyInPlaceInstanceSetup_t | 71 |
| tmalCopyInPlaceInstanceSetup_t | 71 |

## tmolCopyInPlaceCapabilities_t

```
typedef struct {
   ptsaDefaultCapabilities_t   defaultCapabilities;
} tmolCopyInPlaceCapabilities_t, *ptmolCopyInPlaceCapabilities_t;
```

## tmalCopyInPlaceCapabilities_t

```
typedef struct {
   ptsaDefaultCapabilities_t   defaultCapabilities;
} tmalCopyInPlaceCapabilities_t, *ptmalCopyInPlaceCapabilities_t;
```

### Fields

| | |
|---|---|
| `defaultCapabilities` | Pointer to default capabilities struct (see **tsa.h**). |

### Description

Describes the capabilities of CopyInPlace. CopyInPlace does not have capabilities fields other than those in **tsaDefaultCapabilities_t**. In its defaultCapabilities structure, CopyIn-Place includes tsaCapFlagsInPlace in its capabilityFlags. This notifies the default TSSA layer to reattach the empty queues of its corresponding input and output so it is bypassed when empty packets are returned. See the TSSA Advance Details chapter for more details about “in place” components.

## tmolCopyInPlaceInstanceSetup_t

```
typedef struct {
   ptsaDefaultInstanceSetup_t   defaultSetup;
   UInt32                       delay;
   tsaTimSleepFunc_t            TimSleep;
} tmolCopyInPlaceInstanceSetup_t, *ptmolCopyInPlaceInstanceSetup_t;
```

## tmalCopyInPlaceInstanceSetup_t

```
typedef struct {
   ptsaDefaultInstanceSetup_t   defaultSetup;
   UInt32                       delay;
   tsaTimSleepFunc_t            TimSleep;
} tmalCopyInPlaceInstanceSetup_t, *ptmalCopyInPlaceInstanceSetup_t;
```

### Fields

| | |
|---|---|
| `defaultSetup` | Pointer to default instance setup struct (see **tsa.h**). |
| `delay` | The delay in number of ticks. |
| `TimSleep` | Function called to delay. Takes the number of ticks as argument. |

### Description

Describes the variables used by CopyInPlace to set up. Used by the application to pass initial information to CopyInPlace. When the application uses the OL layer, **TimSleep** is set to **tsaDefaultTimSleep**, if the application did not provide it. When the application uses the AL layer, it should provide the **TimSleep** function if is wants a delay. Otherwise, there will be no delay in CopyInPlace.

## CopyInPlace API Functions

This section describes the functions used in the TriMedia CopyInPlace library. Since the CopyInPlace API is used in both the application layer and the operating system layer, each of the functions below, except for **tmalCopyInPlaceCopyPacket**, has a tmal and a tmol counterpart.)

| Name | Page |
|---|---|
| tmolCopyInPlaceGetCapabilities | 73 |
| tmalCopyInPlaceOpen | 76 |
| tmolCopyInPlaceOpen | 75 |
| tmalCopyInPlaceOpen | 76 |
| tmolCopyInPlaceClose | 77 |
| tmalCopyInPlaceClose | 78 |
| tmolCopyInPlaceGetInstanceSetup | 79 |
| tmalCopyInPlaceGetInstanceSetup | 80 |
| tmolCopyInPlaceInstanceSetup | 81 |
| tmalCopyInPlaceInstanceSetup | 82 |
| tmolCopyInPlaceStart | 83 |
| tmalCopyInPlaceStart | 84 |
| tmolCopyInPlaceStop | 85 |
| tmalCopyInPlaceStop | 86 |
| tmolCopyInPlaceInstanceConfig | 87 |
| tmalCopyInPlaceInstanceConfig | 88 |

## tmolCopyInPlaceGetCapabilities

```
tmLibappErr_t tmolCopyInPlaceGetCapabilities(
   ptmolCopyInPlaceCapabilities_t   *cap
);
```

### Parameters

| | |
|---|---|
| `cap` | Pointer to the OL capabilities struct of type **tmolCopyInPlaceCapabilities_t**. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Successful completion. |

### Description

Returns a pointer to the OL capabilities of CopyInPlace.

## tmalCopyInPlaceGetCapabilities

```
tmLibappErr_t tmalCopyInPlaceGetCapabilities(
   ptmalCopyInPlaceCapabilities_t   *cap
);
```

### Parameters

| | |
|---|---|
| `cap` | Pointer to the AL capabilities struct of type **tmalCopyInPlaceCapabilities_t**. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Successful completion. |

### Description

Returns a pointer to the AL capabilities of CopyInPlace.

## tmolCopyInPlaceOpen

```
tmLibappErr_t tmolCopyInPlaceOpen(
   Int   *instance
);
```

### Parameters

| | |
|---|---|
| `instance` | Pointer to the OL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_MEMALLOC_FAILED` | Memory allocation for the instance variables failed. |

### Description

Allocates an OL instance of CopyInPlace for usage.

## tmalCopyInPlaceOpen

```
tmLibappErr_t tmalCopyInPlaceOpen(
   Int   *instance
);
```

### Parameters

| | |
|---|---|
| `instance` | Pointer to the AL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_MEMALLOC_FAILED` | Memory allocation for the instance variables failed. |

### Description

Allocates an AL instance of CopyInPlace for usage.

## tmolCopyInPlaceClose

```
tmLibappErr_t tmolCopyInPlaceClose(
   Int   instance
);
```

### Parameters

| | |
|---|---|
| `instance` | The OL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |

### Description

Deallocates the OL instance. **tmolCopyInPlaceOpen** must have been called previously. The instance must be in the stopped state before this function completes.

## tmalCopyInPlaceClose

```
tmLibappErr_t tmalCopyInPlaceClose(
   Int   *instance
);
```

### Parameters

| | |
|---|---|
| `instance` | The AL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |

### Description

Deallocates the AL instance. **tmolCopyInPlaceOpen** must have been called previously. The instance must be in the stopped state before this function completes.

## tmolCopyInPlaceGetInstanceSetup

```
tmLibappErr_t tmolCopyInPlaceGetInstanceSetup(
   Int                             instance,
   tmolCopyInPlaceInstanceSetup_t   *setup
);
```

### Parameters

| | |
|---|---|
| `instance` | The OL instance. |
| `setup` | Pointer to the OL instance setup struct of type **tmolCopyInPlaceInstanceSetup_t**. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |

### Description

Returns a template OL instance setup struct of type **tmolCopyInPlaceInstanceSetup_t** in parameter **setup**. **tmolCopyInPlaceOpen** must have been called previously.

## tmalCopyInPlaceGetInstanceSetup

```
tmLibappErr_t tmalCopyInPlaceGetInstanceSetup(
   Int                              instance,
   tmalCopyInPlaceInstanceSetup_t   *setup
);
```

### Parameters

| | |
|---|---|
| `instance` | The AL instance. |
| `setup` | Pointer to the AL instance setup struct of type **tmalCopyInPlaceInstanceSetup_t**. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |

### Description

Returns a template AL instance setup struct of type **tmalCopyInPlaceInstanceSetup_t** in parameter **setup**. **tmolCopyInPlaceOpen** must have been called previously.

## tmolCopyInPlaceInstanceSetup

```
tmLibappErr_t tmolCopyInPlaceInstanceSetup(
   Int                                instance,
   tmolCopyInPlaceInstanceSetup_t   *setup
);
```

### Parameters

| | |
|---|---|
| `instance` | The OL instance. |
| `setup` | Pointer to the OL instance setup struct of type **tmolCopyInPlaceInstanceSetup_t**. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_INVALID_SETUP` | Setup parameters are not valid. |
| `TMLIBAPP_ERR_MEMALLOC_FAILED` | Memory allocation for the input or output descriptors failed. |
| `TMLIBAPP_ERR_NOT_STOPPED` | Instance is not in the stopped state. |
| `TMLIBAPP_ERR_TCREATE_FAILED` | Creation of the associated task failed. |
| `TMLIBAPP_ERR_TSUSPEND_FAILED` | Suspension of the associated task failed. |

### Description

Sets up the OL instance of CopyInPlace. **tmolCopyInPlaceOpen** must have been called previously. The instance must be in the stopped state.

## tmalCopyInPlaceInstanceSetup

```
tmLibappErr_t tmalCopyInPlaceInstanceSetup(
   Int                               instance,
   tmalCopyInPlaceInstanceSetup_t   *setup
);
```

### Parameters

| | |
|---|---|
| `instance` | The AL instance. |
| `setup` | Pointer to the AL instance setup struct of type **tmalCopyInPlaceInstanceSetup_t**. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_INVALID_SETUP` | Setup parameters are not valid. |
| `TMLIBAPP_ERR_NOT_STOPPED` | Instance is not in the stopped state. |

### Description

Sets up the AL instance of CopyInPlace. **setup** includes the delay and TimSleep function if provided. CopyInPlace uses the callback functions, **datain**, **dataout**, **completion**, and **control**, which can be set by the application if desired. It is not recommended however, to set datain, dataout, and control, as the default version works with TSSA.

**tmolCopyInPlaceOpen** must have been called previously. The instance must be in the stopped state.

## tmolCopyInPlaceStart

```
tmLibappErr_t tmolCopyInPlaceStart(
   Int   instance
);
```

### Parameters

| | |
|---|---|
| `instance` | The OL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_NOT_SETUP` | Instance has not been set up previously. |
| `TMLIBAPP_ERR_ALREADY_STARTED` | Instance has already been started. |
| `TMLIBAPP_ERR_TSTART_FAILED` | Start of the associated task failed. |
| `TMLIBAPP_ERR_TRESUME_FAILED` | Resumption of the associated task failed. |

### Description

Starts data streaming by calling **tsaDefaultStart**. **tmolCopyInPlaceOpen** must have been called previously. **tmolCopyInPlaceInstanceSetup** must have been called previously.

## tmalCopyInPlaceStart

```
tmLibappErr_t tmalCopyInPlaceStart(
   Int   instance
);
```

### Parameters

| | |
|---|---|
| `instance` | The AL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_NOT_SETUP` | Instance has not been set up previously. |
| `TMLIBAPP_ERR_ALREADY_STARTED` | Instance has already been started. |

### Description

Starts data streaming. It continuously gets input packets and sends it to the output full queue with a delay of a set number of ticks. If TimSleep is null, then there is no delay. Because CopyInPlace is a "in place" component, it sends its input packets directly to the output, without dealing with any empty queues in the process. In other words, it does not return input full packets to the input empty queue, and it does not retrieve output empty packets from the output empty queue.

**tmolCopyInPlaceOpen** must have been called previously. **tmolCopyInPlaceInstanceSetup** must have been called previously.

## tmolCopyInPlaceStop

```
tmLibappErr_t tmolCopyInPlaceStop(
   Int   instance
);
```

### Parameters

| | |
|---|---|
| `instance` | The OL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_NOT_SETUP` | Instance has not been previously set up. |
| `TMLIBAPP_ERR_NOT_STARTED` | Instance has not been previously started. |
| `TMLIBAPP_ERR_TSUSPEND_FAILED` | Suspension of the associated task failed. |

### Description

Stops data streaming by calling **tsaDefaultStop**. Causes instance to fall out of its processing loop in **tmalCopyInPlaceStart**.

**tmolCopyInPlaceOpen** must have been called previously. **tmolCopyInPlaceInstanceSetup** must have been called previously.

## tmalCopyInPlaceStop

```
tmLibappErr_t tmalCopyInPlaceStop(
   Int   instance
);
```

### Parameters

| | |
|---|---|
| `instance` | The AL instance. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_NOT_SETUP` | Instance has not been previously set up. |
| `TMLIBAPP_ERR_ALREADY_STOPPED` | Instance has already been stopped. |

### Description

Stops data streaming. It causes instance to fall out of its processing loop in **tmalCopyIn-PlaceStart**.

**tmolCopyInPlaceOpen** must have been called previously. **tmolCopyInPlaceInstanceSetup** must have been called previously.

## tmolCopyInPlaceInstanceConfig

```
tmLibappErr_t tmolCopyInPlaceInstanceConfig(
   Int                instance,
   UInt32             flags,
   ptsaControlArgs_t  args
);
```

### Parameters

| | |
|---|---|
| `instance` | The OL instance. |
| `flags` | For future development. |
| `args` | Command arguments. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_NOT_SETUP` | Instance has not been setup previously. |
| `TMLIBAPP_QUEUE_EMPTY` | Timed out while waiting for response from component. |

### Description

Configures the instance. It calls tsaDefaultInstanceConfig to put a command packet on the control queue. The commands **tmalCopyInPlaceCommands_t** are enumerated in tmalCopyInPlace.h. **tmolCopyInPlaceOpen** must have been called previously.

## tmalCopyInPlaceInstanceConfig

```
tmLibappErr_t tmalCopyInPlaceInstanceConfig(
   Int                 instance,
   ptsaControlArgs_t   cmdArgs
);
```

### Parameters

| | |
|---|---|
| `instance` | The AL instance. |
| `args` | Command arguments. |

### Return Codes

| | |
|---|---|
| `TMLIBAPP_OK` | Success. |
| `TMLIBAPP_ERR_INVALID_INSTANCE` | Not a valid instance. |
| `TMLIBAPP_ERR_NOT_SETUP` | Instance has not been setup previously. |
| `CP_ERR_UNKNOWN_COMMAND` | Unknown or invalid command. |

### Description

Configures the instance. It is called when Datain (GetFull) and Dataout (GetEmpty) are called with the flags, **tsaDatainCheckControl** and **tsaDataoutCheckControl**, respectively. When called with the **tmalCopyInPlaceChangeDelay** command, it will set the delay to the value in **args–>parameter**. The commands are enumerated in in **tmalCopyInPlace-Commands_t** in tmalCopyInPlace.h. **tmalCopyInPlaceOpen** previously called.