T2BF SPP Lite API Specification

T2BF - tBlue Telematics Bluetooth Framework

Abstract

T2BF is a Bluetooth framework with advanced Hands-free and other features for telematic usages. This document describes the free of charge t2BF SPP Lite variant, which contains support for SPP only.

Application

This document describes an API for controlling and using the package.

Contents

1 Introduction

Cybercom provides a Bluetooth framework with advanced Hands-free and other features for telematic usages – tBlue Telematics Bluetooth Framework – **T2BF**. The framework can be controlled and monitored over one API, which is specified in this document. The T2BF SPP Lite variant, which is described in this document, only has support for SPP.

1.1 Purpose

The specification explains in detail how to control T2BF SPP Lite using the T2BF SPP Lite API.

1.2 Deviations

Some parts of the specification may be changed in future updates.

1.3 Terminology

BT	Bluetooth
BD	Bluetooth device
FMAN	T2BF File MANager
HF	Hands-free
HS	Headset
OPP	Object Push Profile
PIM	Personal Information Manager
SPP	Serial Port Profile
T2BF	tBlue Telematics Bluetooth Framework

1.4 Revision History

Rev	Date	Issued by	Changes
A	2009-01-16	Daniel Persson (DPER)	Based on t2BF API specification. All descriptions of t2BF functionality, except functionality included in SPP Lite variant, were removed. A list of available function calls in full t2BF was created. First release.

2 Position in System

T2BF API is situated at top level of the framework. It is a high level control with a perspective close to a man-machine interface. The figure 1 shows in general terms the modules controlled by API. The figure describes the full T2BF layout. The T2BF SPP Lite variant includes the T2BF, BTM, BTAL, HWAL and OSAL components.

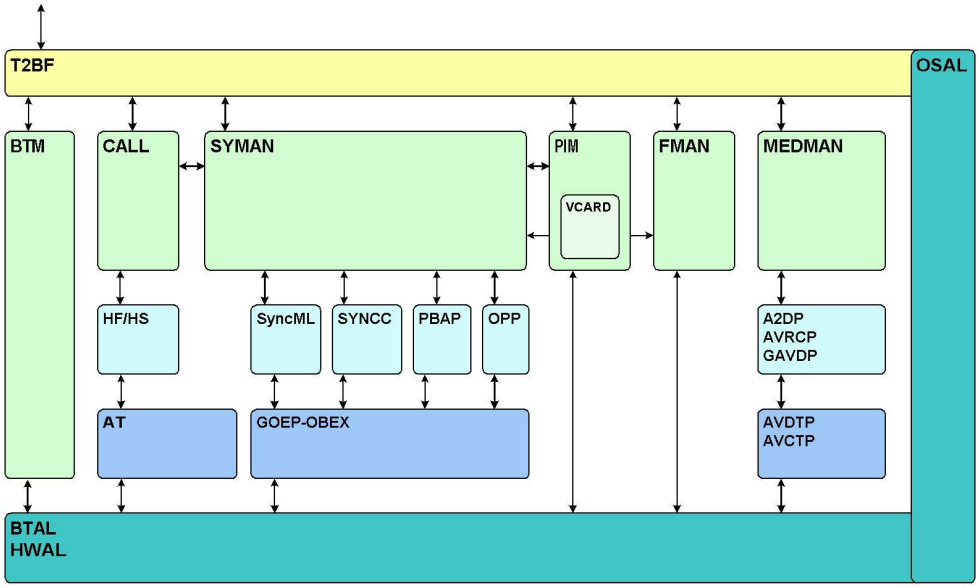


Figure 1. T2BF API location and T2BF overview.

3 Application Programmer Interface

T2BF API is specified in the following paragraphs.

3.1 Introduction

T2BF API consists of:

- Functions with input parameters and return values indicating result.
- Events in callback.
- Read functions for information or to read out event data.

Functions are divided into BT functions, call functions, sync functions and general functions. Events are divided into BT events, call events and sync events. Some events indicate that a read function should be called, in those cases that the event carries data or parameters of some sort.

The function names are written as they are defined in the actual T2BF SW module.

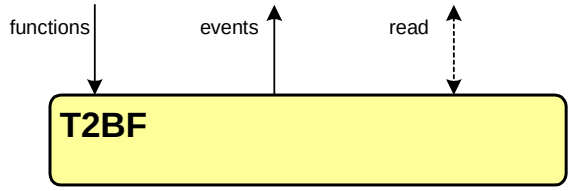


Figure 2. T2BF API flow

3.1.1 Events and flow control

Events that indicate data (number strings or object) **MAY** or **MUST** be followed by a read function to read out the data. MAY means that read function can be called if wanted. MUST mean that read function must be called as soon as possible. See event explanations. This is because data or linked with certain events are vital (such as an object received); T2BF will not send another same event before the read function is called.

Also, the functionality that the event originates from will halt until the read function is called.

No major data is transferred together with an event. This is to avoid choking the application or host with data it is not ready for.

Function outputs are the main channel for moving data. However, most functions transfer one part at a time and not for example 10-20 contacts at a time, to allow all the parts of the entire T2BFsystem to be able to send events and other things. This is especially important in those cases the T2BF API is accessed over a serial connection such as TSFP.

4 General functions

Functions of general kind belong to this group. There are two init functions, one for the main framework and one related to phonebook access and file storage.

4.1.1 T2BFInit

This function is used to initialise the basic T2BF system. This includes setting up parameters and a callback for events. This **MUST** be the first function called.

Input parameters:

Type	Meaning	Note
TT2BFBtSettings*	Pointer to type with BT settings	Not NULL.
TT2BFBtCb	Pointer to T2BF BT callback function.	Not NULL.

Output parameters:

-

Return values:

Value	Meaning
T2BF_STATUS_OK	T2BF init OK all input parameters accepted.
T2BF_STATUS_ERROR_PARAM	NULL parameters are not allowed for some.
T2BF_STATUS_ERROR	Unspecified error during initialization.
T2BF_STATUS_ERROR_OS	Problem with OS resources.
T2BF_STATUS_SOLID_MEM_ERROR	Problem with the non-volatile storage.
T2BF_STATUS_HW_ERROR	Problem with the communication HW (e.g. UART to HCI chip).
T2BF_STATUS_BT_COMMUNICATION_TIMEOUT	A timeout occurred during communication to BT HW (e.g. no response from HCI chip).
T2BF_STATUS_BT_COMMUNICATION_ERROR	Unexpected data received from BT HW (e.g. unexpected answer from HCI chip).

4.1.2 T2BFRegisterStrConv

With this function it is possible to register an external string converter function. This function can then be called by T2BF to convert strings with different character sets and encodings to UTF-16 format. This function will only be used by T2BF for strings with character sets not supported by T2BF's internal string converter.

Input parameters:

Type	Meaning	Note
TT2BFConvUtf16Cb	Pointer to converter function.	

Output parameters:

-

Return values:

-

4.1.2.1 TT2BFConvUtf16Cb

This type definition describes how the string converter callback should be implemented. The converter function should convert the input string from the encoding and character set given in the parameters to UTF-16 format, and place the result in the output buffer. If the result string becomes to large to fit into the output buffer, it should be truncated (with an ending NULL character).

Input parameters:

Type	Meaning	Note
TString	An ASCII string describing the character set of the input string.	NULL terminated.
TT2BFEncoding	Encoding of the input string. Possible values: T2BF_ENCODING_UNKNOWN Assume no encoding. T2BF_ENCODING_QUOTED_PRINTABLE String is encoded in quoted printable format. T2BF_ENCODING_BASE64 String is encoded in base64 format. T2BF_ENCODING_8BIT String is encoded in 8bit format.	
uint8*	Input string. This string should be converted from the character set and encoding described by the above parameters, to UTF-16 format.	NULL terminated.
uint16	Length of the output buffer. Indicates the number of 16-bit words.	

Output parameters:

Type	Meaning	Note
TUTF16String	Output buffer. Here the result of the conversion should be placed. This should be a UTF-16 string.	Should be NULL terminated.

Return values:

Value	Meaning
T2BF_STATUS_OK	Conversion was successful.
T2BF_STATUS_NOT_SUPPORTED	Conversion failed (character set not supported). When this error code is returned, T2BF will perform a "best effort" conversion, which will remove encodings but assume ASCII on the character set.
T2BF_STATUS_INFO_OVERFLOW	Conversion was successful but output truncated. The entire input string did not fit into the output buffer when converted to UTF-16.
T2BF_STATUS_ERROR	Something went wrong during conversion. T2BF will ignore the result and use an empty string.

4.1.3 T2BFStart

This function starts all BT functionality of the T2BF system.

Input parameters:

-

Output parameters:

-

Return values:

Value	Meaning
T2BF_STATUS_OK	T2BF started OK.
T2BF_STATUS_ERROR_OS	Problems with OS resources.
T2BF_STATUS_ERROR	Unspecified error.

4.1.4 T2BFStop

The function implementation will turn BT connectability and discoverability off. Use T2BFStart to reactivate BT connectability.

Input parameters:

-

Output parameters:

-

Return values:

Value	Meaning
T2BF_STATUS_OK	BT stopped OK.
T2BF_STATUS_ERROR	Unspecified error.

4.1.5 T2BFEnd

The function will shut down the entire T2BF framework. All processes and other resources used by the framework will be released. This function will disconnect any connected device, and also resets the HCI chip. If this function has been called, T2BF has to be reinitialized before it can be used again (using T2BFInit etc).

Input parameters:

-

Output parameters:

-

Return values:

Value	Meaning
T2BF_STATUS_OK	Framework was shut down correctly.
T2BF_STATUS_ERROR	Unspecified error.

4.1.6 T2BFReadLocalBdAddr

Function returns local BD address.

Input parameters:

-

Output parameters:

Value	Meaning
TBdAddr	Supply a pointer to a BD address type.

Return values:

-

4.1.7 T2BFReadSwVersion

Function returns the current SW version of the T2BF framework.

Input parameters:

Type	Meaning	Note
uint8	Max length to fill into buffer.	Use T2BF_SW_VERSION_LENGTH to make sure to get complete string.

Output parameters:

Value	Meaning
TString	Supply a string where SW version will be written. String is NULL terminated.

Return values:

-

4.1.8 T2BFReadNvsVersion

This function returns the current version of the NVS (Non Volatile Storage) handler in the T2BF framework.

Input parameters:

Type	Meaning	Note
uint8	Max length to fill into buffer.	Normally the version is only one character, followed by the NULL character.

Output parameters:

Value	Meaning
TString	Supply a string where version will be written. The string will be NULL terminated.

Return values:

-

4.1.9 T2BFWriteName

Writes local device name to T2BF.

Input parameters:

Type	Meaning	Note
TUTF16String	Local device name.	Must be NULL terminated.

Output parameters:

-

Return values:

Value	Meaning
T2BF_STATUS_OK	Name was written OK.
T2BF_STATUS_ERROR	Name write failed.

4.1.10 T2BFReadName

Reads local device name from T2BF.

Input parameters:

Type	Meaning	Note
uint8	Size of buffer.	

Output parameters:

Type	Meaning	Note
TUTF16String	Local device name.	String is always NULL terminated.

Return values:

Value	Meaning
T2BF_STATUS_OK	Name was read OK.
T2BF_STATUS_ERROR	Name read failed.

5 BT events and functions

BT events and functions are a group related to pairing and connection information to a remote device.

5.1 BT events

Connection events indicate changes in connection with remote BT device and pairing. Events will be sent to the callback of type TT2BFBtCb set in T2BFInit function. Do not perform work in callback function.

The events are written as they are defined in the actual T2BF SW module. Each BT event comes with the remote BD address and a service ID (when applicable).

5.1.1 TT2BFBtCb

This callback informs of BT events.

Input parameters:

Type	Meaning	Note
TT2BFBtEvent	BT event that has occurred.	See event meaning below.
TBdAddr	BD address that the event refers to.	-
TServiceId	Service ID that the event refers to when applicable	SERVICE_ID_UNKNOWN is used when the event does not refer to any service.

Output parameters:

-

Return values:

-

5.1.2 Events

TT2BFBtEvent type will be of the following values.

T2BF_BT_EVENT_CONNECTED

Event is received when connected with remote device at GAP level. No service ID.

T2BF_BT_EVENT_DISCONNECTED

Event is received whenever a connection with remote device at GAP level is lost or disconnected by remote device. The service id parameter indicates the reason why the

disconnection happened. Reason codes are specified in the Bluetooth specification (see chapter about “error codes”).

T2BF_BT_EVENT_CONNECTED_PORT

Event is received when connected with remote device at SPP level. Service ID will be included if the service has registered its port, which all standard T2BF services do.

T2BF_BT_EVENT_DISCONNECTED_PORT

Event is received when connected with a remote device at SPP level. Service ID will be included if the service has registered its port, which all the standard T2BF services do.

T2BF_BT_EVENT_CONNNECTION_LOST_PORT

Event is received when connection is lost with a remote device at SPP level. Service ID will be included if the service has registered its port, which all the standard T2BF services do.

T2BF_BT_EVENT_AUDIO_CONNECTED

An audio link is connected with the remote device. Service ID if available.

T2BF_BT_EVENT_AUDIO_DISCONNECTED

An audio link is disconnected with the remote device. Service ID if available.

T2BF_BT_EVENT_AUTHORISE

Not used at the moment.

T2BF_BT_EVENT_ AUTHORISATION_CANCELLED

Not used at the moment.

T2BF_BT_EVENT_AUTHENTICATE

A remote device needs to be authenticated. T2BFAuthenticate **MUST** be called accept or reject authentication. Use T2BFReadRemoteName to find remote BD name before authenticating. No valid service ID.

T2BF_BT_EVENT_AUTHENTICATION_CANCELLED

The remote device has cancelled its authentication. No valid service ID.

T2BF_BT_EVENT_AUTHENTICATED

Event is received when a remote device has successfully authenticated – “paired” with T2BF and can be found in the paired list by using the function T2BFReadPaired. No valid service ID.

T2BF_BT_EVENT_INQUIRY_RESULT

This event is generated after a call to T2BFInquiryStart, for each remote device that has been found during the inquiry. TBdAddr parameter will hold the Bluetooth device for the found device. TServiceId parameter will hold the index number (starting on 0) that the found device will have in the data structures used to store all results (parameters pBdAddresses, pDeviceInfo and pNames given to T2BF in T2BFInquiryStart).

T2BF_BT_EVENT_INQUIRY_COMPLETE

This event is generated when an inquiry sequence is completed. TServiceId parameter will indicate the number of found devices that has been stored in the result data

structures (parameters pBdAddresses, pDeviceInfo and pNames given to T2BF in T2BFInquiryStart).

T2BF_BT_EVENT_REMOTE_SERVICES

This event indicates the result of a call to T2BFRemoteServices. This event also indicates that a service search has been made during the first connection to a remote device. So if this event arrives during a connection the remotes services have been updated for that device in pairing list. TBdAddr indicates the Bluetooth address of the remote device that has been queried for its services. TServiceId indicates which services that were supported (out of the ones of interest). Several services could be supported.

T2BF_BT_EVENT_REMOTE_SERVICES_FAILED

This event indicates that the query for services of a remote device failed for some reason. This event also indicates that a service search has failed during the first connection to a remote device. So if this event arrives during a connection the remotes services have not been updated for that device in pairing list. Normally this is since the remote device did not respond (since units are out of range). TBdAddr indicates which remote unit the query was aimed for. Note that this events does not indicate anything about supported services. Since the remote device could not be reached, its services are still unknown.

T2BF_BT_EVENT_STARTED

T2BF started.

T2BF_BT_EVENT_STOPPED

T2BF stopped.

5.2 BT functions

This includes functions that relate to authentication, paired devices and connection management and information.

5.2.1 T2BFReadPaired

Use function to read how many remote devices are currently paired. It will return a number. Maximum units that can be paired is also written in the output.

Input parameters:

-

Output parameters:

Type	Meaning	Note
*uint8	Max number of units that can be paired.	

Return values:

Value	Meaning
uint8	Number of currently paired devices.

5.2.2 T2BFReadPairedInfo

Function is used to read out info about paired device, address and name if available.

Input parameters:

Type	Meaning	Note
uint8	Prio.	Highest prio is 1, this is the latest device. T2BFReadPaired return value is the lowest prio.
Boolean	TRUE – read latest connected. FALSE – read latest paired.	
uint8	Max length of name string.	

Output parameters:

Type	Meaning	Note
TBdAddr	BD address.	
TUTF16String	NULL terminated UTF-16 device name.	Name may not have been successfully read from remote device. This is indicated by a 0 length string.
TServiceId*	Services the remote device supports. Are presented like defined in TServiceId.	

Return values:

Value	Meaning
T2BF_STATUS_OK	Info read out OK.
T2BF_STATUS_ERROR_PARAM	Prio index out of range.

5.2.3 T2BFRemovePaired

Function is used to remove paired device.

Input parameters:

Type	Meaning	Note
TBdAddr	BD address of device to remove from paired list.	-

Output parameters:

-

Return values:

Type	Meaning
void	No return value.

5.2.4 T2BFRemoveAllPaired

Function is used to remove all paired devices.

Input parameters:

-

Output parameters:

-

Return values:

Type	Meaning
void	No return value.

5.2.5 T2BFPairedDeletePolicy

Sets delete policy for the paired list, when the list is full. Deletion is performed when a new pairing is performed.

Input parameters:

Type	Meaning	Note
TT2BFDeletePolicy	T2BF_DELETE_POLICY_DENY Do not delete any devices, new pairings should be denied.	-

	T2BF_DELETE_POLICY_CONNECTED Delete the device that was connected the longest ago. T2BF_DELETE_POLICY_PAIRED Delete the device that was paired the longest ago.	

Output parameters:

-

Return values:

Type	Meaning
void	No return value.

5.2.6 T2BFWritePin

Change the PIN to be used during BT authorisation. The PIN is stored in non volatile memory. At first startup, a default PIN of "0000" is set until this function is called.

Input parameters:

Type	Meaning	Note
TString	PIN string	NULL terminated ASCII string.

Output parameters:

-

Return values:

Value	Meaning
T2BF_STATUS_OK	PIN was set OK.
T2BF_STATUS_ERROR	PIN was not set.

5.2.7 T2BFReadPin

Reads the current PIN.

Input parameters:

Type	Meaning	Note
uint8	Max length of the PIN,	

	including NULL character.	

Output parameters:

Type	Meaning	Note
TString	PIN buffer.	

Return values:

Value	Meaning
T2BF_STATUS_OK	PIN was read OK.
T2BF_STATUS_ERROR	PIN was not read.

5.2.8 T2BFAuthenticate

Function to either accept or deny an ongoing authorisation, after a T2BF_BT_EVENT_AUTHENTICATE event is received.

Input parameters:

Type	Meaning	Note
TBdAddr	BD address of remote to authenticate.	Address is received together with event.
boolean	TRUE if T2BF shall try to authenticate remote, FALSE if remote shall be denied.	

Output parameters:

-

Return values:

Type	Meaning
void	No return value.

5.2.9 T2BFAutoAuthentication

Auto authentication means that T2BF will use the default (0000) or set PIN (by calling T2BFPin) to automatically authenticate remote devices. If auto authentication is turned off, all authentication attempts by a remote device will be indicated by T2BF_BT_EVENT_AUTHENTICATE event. This event **MUST** be responded with a call to T2BFAuthenticate.

Auto authorisation is default set to T2BF_AUTO_AUTH_OFF before this function is called.

Input parameters:

Type	Meaning	Note
T2BFAutoAuth	T2BF_AUTO_AUTH_OFF All pairing events will be sent to user. Default if the function is not called. T2BF_AUTO_AUTH_ALLOW Default or set PIN is used to allow all pairings. T2BF_AUTO_AUTH_DENY All pairing events are denied.	-

Output parameters:

-

Return values:

Type	Meaning
void	No return value.

5.2.10 T2BFRemoteName

Use this function to get the remote user friendly name on a non-paired device. T2BF will automatically find remote name during pairing and store this name together with authentication information. Use T2BFReadPairedInfo to get this name.

Input parameters:

Type	Meaning	Note
TBdAddr	Address of remote device	If TRUE the new PIN will be output to the string.
uint16	Max length of output string.	If output name is longer than string, it will be truncated.

Output parameters:

Type	Meaning	Note
TUTF16String	Output string, NULL terminated UTF-16 string.	

Return values:

Value	Meaning
T2BF_STATUS_OK	Name read out OK.
T2BF_STATUS_ERROR	Not able to connect with remote device

5.2.11 T2BFDiscoverable

Function to control discoverability on and off. Turning discoverability off will exclude the possibility for new devices to find the device (for authentication purposes), and to receive object from non-paired devices.

Input parameters:

Type	Meaning	Note
boolean	TRUE – Discoverable is always on. FALSE – Discoverable is always off.	

Output parameters:

-

Return values:

Type	Meaning
void	No return value.

5.2.12 T2BFInquiryStart

This function initiates an inquiry process (search for devices in range). Results will be stored in the result arrays given by TBdAddr* pBdAddresses, TT2BFDeviceInfo* pDeviceInfo and TUTF16String pNames. The event T2BF_BT_EVENT_INQUIRY_RESULT will be generated for each device that is found. T2BF_BT_EVENT_INQUIRY_COMPLETE indicates that the inquiry process is finished.

The filtering that can be done filters in all units fulfilling any of the service or device classes given in the filters.

Input parameters:

Type	Meaning	Note
Uint8	Timeout in seconds.	The search for devices will go on for this period of time. After that, the name for each found device will be queried. Each name query

		will take some time, giving an unknown total amount of time for the entire process. Inquiry result events will not be generated during the inquiry timeout (specified by this parameter), but will start to arrive during the name query part of the process. Max timeout value is 60 seconds.
TServiceClass	Service class filter.	This filter can be used to filter out only devices with a specific service class in the result lists. One or several service classes can be filtered in.
TDeviceClass	Major device class filter.	This filter can be used to filter out only devices with a specific major device class in the result lists. One or several major device classes can be filtered in.
Uint8	Maximum number of results.	Indicates the length of the arrays used for storing the results.
Uint8	Maximum name length.	Indicates the length of the name string buffers in the array of name string results.

Output parameters:

Type	Meaning	Note
TUTF16String	Should point to an array with name string buffers where the inquiry results can be stored.	
TBdAddr*	Should point to an array of BD address buffers where the inquiry results can be stored.	
TT2BFDeviceInfo*	Should point to an array of device info fields where the inquiry results can be stored.	

Return values:

Value	Meaning
T2BF_STATUS_OK	Inquiry process started OK.

T2BF_STATUS_BUSY	Not allowed since an inquiry process is already ongoing.
T2BF_STATUS_ERROR	Unspecified error.

5.2.13 T2BFInquiryStop

This function stops an ongoing inquiry process. Any already found devices and corresponding information can be found in result arrays.

Input parameters:

-

Output parameters:

-

Return values:

Value	Meaning
T2BF_STATUS_OK	Inquiry process stopped.
T2BF_STATUS_ERROR	Unspecified error.

5.2.14 T2BFRemoteServices

This function retrieves information about services supported by a remote device. Results are given by the events T2BF_BT_EVENT_REMOTE_SERVICES and T2BF_BT_EVENT_REMOTE_SERVICES_FAILED.

Only the services indicated by the filter parameter are searched for.

If filter include all services and the supported services are unknown for the paired remote device the answer will also be saved in paired list.

Input parameters:

Type	Meaning	Note
TBdAddr	BD address of the remote device.	
TServiceId	Specifies the services that are of interest.	One or several services can be filtered in. These services are not possible to search for. They will never be showed in answer even if they

		are set in TServiceId. SERVICE_ID_OBEX SERVICE_ID_OP_CLIENT SERVICE_ID_FT_CLIENT SERVICE_ID_SYNCML_SERVER SERVICE_ID_DUN_DT

Output parameters:

-

Return values:

Value	Meaning
T2BF_STATUS_OK	Search for services started OK.
T2BF_STATUS_BUSY	Not allowed since a service search process is already ongoing.
T2BF_STATUS_ERROR	Unspecified error.

5.2.15 T2BFSetLowPowerParameters

This function is used to set parameters that should be used for low power modes. If low power modes is never activated (by T2BFSetLowPowerMode), it is not necessary to set these parameters.

Note that this function is used for low power modes that are initiated from the local device. Low power mode requests from the remote device will be accepted/rejected due to low level settings in the framework, and can not be controlled via this API.

This function is not supported in current version of t2BF.

Input parameters:

Type	Meaning	Note
TT2BFLowPowerParameters	Struct with parameter settings.	Refer to the Bluetooth specification for a detailed description of the parameters.

Output parameters:

-

Return values:

Type	Meaning
T2BF_STATUS_OK	Parameters were set correctly.
T2BF_STATUS_ERROR	Parameter setting failed.

5.2.16 T2BFSetLowPowerMode

This function is used to set current low power mode.

Note that this function is used for low power modes that are initiated from the local device. Low power mode requests from the remote device will be accepted/rejected due to low level settings in the framework, and can not be controlled via this API.

This function is not supported in current version of t2BF.

Input parameters:

Type	Meaning	Note
TT2BFLowPowerMode	T2BF_LOWPOWER_OFF Low power modes will not be initiated from the local device. T2BF_LOWPOWER_SNIFF_MODE Sniff mode will be initiated from the local device when possible.	

Output parameters:

-

Return values:

Type	Meaning
T2BF_STATUS_OK	Parameters were set correctly.
T2BF_STATUS_ERROR	Parameter setting failed.

5.2.17 T2BFReadHciChip

Reads FW version of the HCI chip, if available.

Input parameters:

Type	Meaning	Note
uint8	Max length of FW version	

	string.	

Output parameters:

Type	Meaning	Note
TString	FW version string.	String format is port dependant.

Return values:

Type	Meaning
T2BF_STATUS_OK	FW string was read.
T2BF_STATUS_ERROR	FW was not available.

5.2.18 T2BFGenericHci

Sends a generic HCI command to the HCI chip.

Warning: Sending generic HCI commands directly to the HCI chip may cause unpredicted behaviour, and should be used for debug purposes only.

Input parameters:

Type	Meaning	Note
uint8	OGF of the command.	See Bluetooth spec or datasheet for HCI chip.
uint16	OCF of the command.	See Bluetooth spec or datasheet for HCI chip.
uint8	Length of the parameter buffer passed to this function.	
uint8*	Parameter buffer.	Can be set to NULL.
uint8	Size of receive buffer.	

Output parameters:

Type	Meaning	Note
uint8*	Length of the received parameters.	String format is port dependant.
uint8*	Received parameters.	Command complete status is not included in the parameters.

Return values:

Type	Meaning
T2BF_STATUS_OK	Command was issued successfully.
T2BF_STATUS_ERROR	Command failed.

6 SPP events and functions

T2BF exports an SPP interface, which implements support for creating and using serial ports over Bluetooth.

6.1 SPP events

SPP events informs about the status of a specific serial port, such as connection creation and disconnection etc.

6.1.1 TT2BFSppCb

This callback informs of SPP events.

Input parameters:

Type	Meaning	Note
TT2BFSppEvent	SPP event	-
TPort	Port handle to which the event is related.	-

Output parameters:

-

Return values:

-

6.1.2 Events

TT2BFSppEvent type will be of the following values.

T2BF_SPP_EVENT_DISCONNECTED

The SPP connection was released.

T2BF_SPP_EVENT_CONNECTION_LOST

The SPP connection was lost.

T2BF_SPP_EVENT_CONNECTED

SPP connection was created.

T2BF_SPP_EVENT_DATA

There is data available for reading.

6.2 SPP functions

6.2.1 T2BFSppCreatePort

This function creates a serial port. All necessary resources are allocated, but the port is not yet visible to other devices.

Input parameters:

Type	Meaning	Note
uint16	Frame size suggestion	Basically any value will work, greater values increases bandwidth due to larger buffers.
TT2BFSecurity	Security setting for this port.	
TT2BFSppCb	Event callback for this port.	
TString	Name of the port.	

Output parameters:

Type	Meaning	Note
TPort	Port handle to use in all future calls related to this port.	

Return values:

Value	Meaning
T2BF_STATUS_OK	SPP port was created.
T2BF_STATUS_ERROR	Failed to create SPP port.

6.2.2 T2BFSppDelete

Deletes a serial port.

Input parameters:

Type	Meaning	Note
TPort	Handle to the port to delete.	

Output parameters:

-

Return values:

Value	Meaning
T2BF_STATUS_OK	SPP port was deleted.
T2BF_STATUS_ERROR	Invalid port handle.

6.2.3 T2BFSppOpen

Opens a serial port for incoming connections. Port must be created prior to this call.

Input parameters:

Type	Meaning	Note
TPort	Handle to the port to open.	

Output parameters:

-

Return values:

Value	Meaning
T2BF_STATUS_OK	SPP port was opened.
T2BF_STATUS_ERROR	Invalid port handle.

6.2.4 T2BFSppClose

Closes a serial port that is open for connections.

Input parameters:

Type	Meaning	Note
TPort	Handle to the port to close.	

Output parameters:

-

Return values:

Value	Meaning
T2BF_STATUS_OK	SPP port was closed.
T2BF_STATUS_ERROR	Invalid port handle.

6.2.5 T2BFSppConnect

Connects a serial port to a remote serial port. Local port must be created prior to this call.

Input parameters:

Type	Meaning	Note
TPort	Handle to the port to use.	
TBdAddr	BD address of the remote device.	
TString	Name of the SPP port of the remote device.	Set to NULL if name should be ignored. In that case, the first SPP port on the remote device will be used.

Output parameters:

-

Return values:

Value	Meaning
T2BF_STATUS_OK	Connection was created.
T2BF_STATUS_NOT_CONNECTED	Failed to create connection.
T2BF_STATUS_NOT_SUPPORTED	Remote device does not support the SPP profile.
T2BF_STATUS_NOT_AVAILABLE	SPP port with the specified name is not available on the remote device.

6.2.6 T2BFSppDisconnect

Disconnects a serial port.

Input parameters:

Type	Meaning	Note
TPort	Handle to the port to disconnect.	

Output parameters:

-

Return values:

Value	Meaning
T2BF_STATUS_OK	SPP port was disconnected.
T2BF_STATUS_ERROR	Invalid port handle.

6.2.7 T2BFSppTx

Sends data on a connected serial port.

Input parameters:

Type	Meaning	Note
TPort	Handle to the port to use.	
uint16	Length of data.	
uint8*	Pointer to the data.	

Output parameters:

-

Return values:

Value	Meaning
T2BF_STATUS_OK	Data was sent.
T2BF_STATUS_ERROR	Transmission failed.

6.2.8 T2BFSppRx

Received data on a connected serial port. This function will block until the requested number of bytes have been received. Use T2BFSppRxCount to check for available data.

Input parameters:

Type	Meaning	Note
TPort	Handle to the port to use.	
uint16	Length of data to receive.	

Output parameters:

uint8*	Pointer to the data buffer.	

Return values:

Value	Meaning
T2BF_STATUS_OK	The requested data was received.
T2BF_STATUS_ERROR	Reception failed.

6.2.9 T2BFSppRxCount

Reports number of available bytes on a serial port.

Input parameters:

Type	Meaning	Note
TPort	Handle to the port to use.	

Output parameters:

-

Return values:

Type	Meaning
uint16	Number of bytes that are available for reading.

7 Full T2BF functionality

This chapter briefly describes the extra modules in the full t2BF version. For each module, all available functions are listed.

7.1 Call events and functions

Call events and functions form a group of functionality that controls the hands-free interface. This includes dialing, answering and similar tasks together with extended features such as list dial. The T2BF must be connected to a remote device to be able to do any call functionality. The number of supported features is depending on the capabilities of the remote device.

7.1.1 CALL functions

T2BFHandsfreeCreatePort
T2BFHandsfreeDeletePort
T2BFHandsfreeConnect
T2BFHandsfreeDisconnect
T2BFVoiceDial
T2BFRedial
T2BFAnswer
T2BFTerminate
T2BFTerminateAll
T2BFDialNumber
T2BFDtmf
T2BFSwitch
T2BFConference
T2BFPrivateConsultation
T2BFTransferAudio
T2BFVolume
T2BFReadNumberName
T2BFReadConfNumberName
T2BFReadAvailableList
T2BFReadList
T2BFReadOperator
T2BFReadSubscriberNumber

7.2 Sync events and functions

T2BF can access the phonebook from remote device by using different one-way synchronisation methods – BT sync profile, SyncML, OPP or AT. A fast resynchronisation is also possible when BT sync and SyncML are used.

Items that relate to phonebook are internally transferred to the PIM module. PIM can be set-up as either locally handled or external, host located. Refer to PIM events and functions.

Object received from OPP are internally transferred to the file manager (FMAN) module. FMAN will then transfer the file to the application/host. Refer to File events and functions.

7.2.1 Sync functions

The functions used to control the Sync manager (SYMAN) of T2BF are described here.

T2BFSync
T2BFSyncStop
T2BFReadObjectInfo
T2BFAcceptObject
T2BFRejectObject
T2BFReadSyncSupported

7.3 PIM events and functions

T2BF includes a PIM (Personal Information Manager) module that can either transfer PIM items, contacts information, to the application/host to process or store internally within the T2BF system.

In case of application/host located storage of PIM items, the app/host must respond to different events and retrieve PIM data. How the PIM data is further handled, stored and accessed is then the responsibility of the app/host. Benefit of this is total PIM data control from application point of view with all the storage resources of the host.

In case of internal or local storage of PIM items, no PIM events are used, but functions handling the PIM are provided by the API. Benefit of this is quick development of the application with an already complete internal PIM.

Host or local storage of PIM items is setup in the T2BFInitStore function and the T2BFPimSettings parameter. Also, the actual system HW must have storage space for the set-up PIM data area.

7.3.1 PIM functions

The functions used to access the internally stored PIM are described here.

The contacts in a phonebook are accessed in alphabetical order, collating according to the unicode collation standard.

T2BFPimPhonebookHandle
T2BFPimPhonebookSize
T2BFPimPhonebookDelete
T2BFPimPhonebookDeleteSync
T2BFPimReadInitalCharList
T2BFPimSearch
T2BFPimContactSyncType
T2BFPimContactName
T2BFPimContactNumberCount
T2BFPimContactNumber
T2BFPimContactEmailCount
T2BFPimContactEmail

T2BFPimContactAddressCount
T2BFPimContactAddressType
T2BFPimContactAddressField
T2BFPimContactLabelCount
T2BFPimContactLabel
T2BFPimDoneReadContact

7.3.2 Host PIM functions

The PIM storage can be set-up to be the responsibility of the host, host located PIM storage. The T2BF PIM module will in this case send events to application/host to instruct of different PIM data transfers that are needed. The host PIM functions are used to respond to these events.

Host PIM should keep note of each single vCard/contact together with its ID string to be able to delete it when performing a fast sync.

T2BFHostPimSetPhonebook
T2BFHostPimPhonebookCleared
T2BFHostPimPhonebookDeleted
T2BFHostPimPutRead
T2BFHostPimGetWrite
T2BFHostPimVcardDeleted
T2BFHostPimGetAtContact

7.4 File events and functions

T2BF includes a file manager (called FMAN) module that can either transfer files to the application/host to process or store internally within the T2BF system.

7.4.1 Host File functions

The File manager storage is set-up to be the responsibility of the host. The T2BF File manager module – FMAN - will in this case send events to application/host to instruct of different file data transfers that are needed. The host file functions are used to respond to these events.

T2BFHostFileGetFilename
T2BFHostFileOpen
T2BFHostFilePutRead
T2BFHostFileGetWrite
T2BFHostFileClose

7.5 Media events and functions

This interface gives access to the media streaming functionality in T2BF. This interface supports A2DP sink, and AVRCP controller.

7.5.1 Media functions

T2BFMediaCreatePort
T2BFMediaDeletePort
T2BFMediaConnect
T2BFMediaDisconnect
T2BFMediaControl

7.6 Media metadata events and functions

This interface gives access to media metadata functionality in T2BF for a media controller unit. Metadata is a part of the AVRCP version 1.3, and this interface supports the AVRCP controller side.

7.6.1 Media Metadata functions

T2BFMediaMetadataInit
T2BFMediaGroupCmd
T2BFMediaRegPlaybackPos
T2BFMediaRegBatteryStatus
T2BFMediaRegSystemStatus
T2BFMediaGetPlayerApplicationValue
T2BFMediaSetPlayerApplicationValue

7.7 DUN events and functions

This interface gives access to the dial up network profile functionality in T2BF.

7.7.1 DUN functions

T2BFDunConnect
T2BFDunDisconnect
T2BFDunRead
T2BFDunWrite

8 Parameter types

Parameter types used in function descriptions are not further explained in this document but reference is given to the API implementation header file [1].

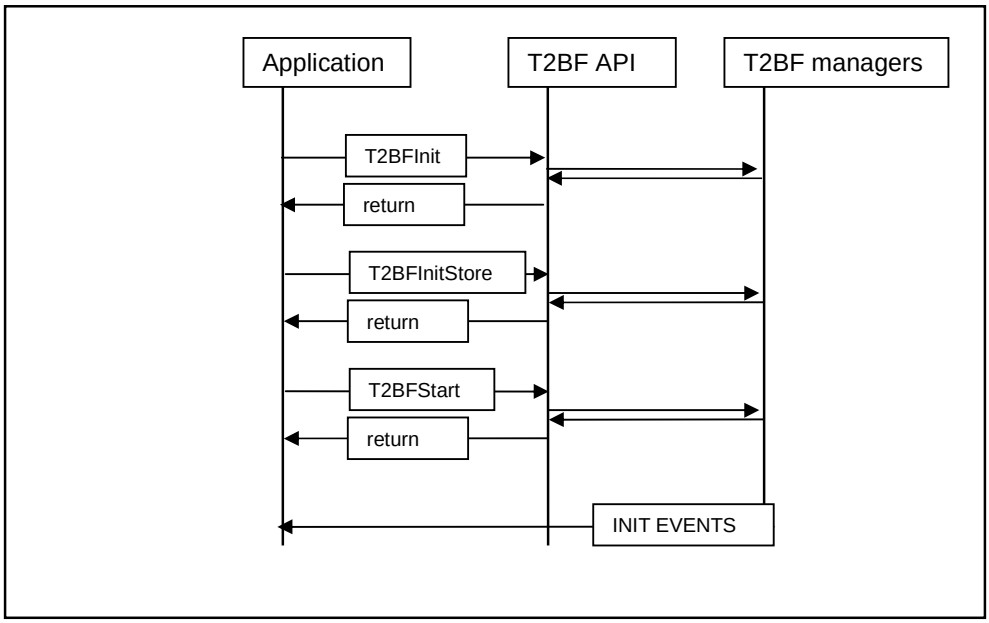
In general, all strings are null terminated. If the string is an output, the maximum length field must be set. If the output string is longer than maximum length, the string is truncated and null terminated. Lengths will however be indicated in the callback value field.

9 T2BF sequence diagrams

The following paragraphs show some of the usual usage scenarios, and in what sequence to call the T2BF API to achieve certain functionality. Combining different scenarios may also possible.

9.1 T2BF init and start sequence

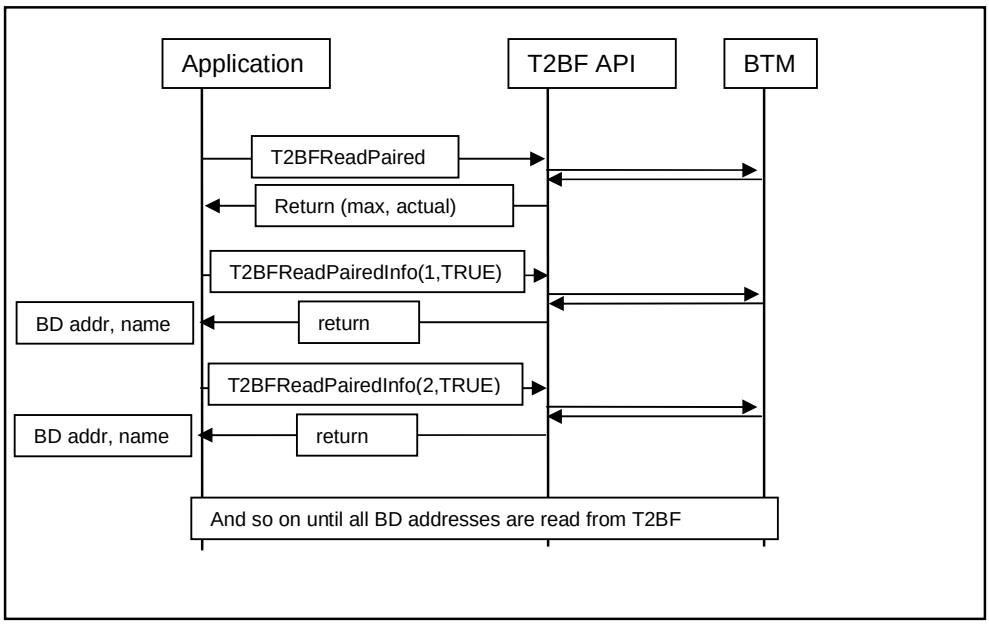
T2BF shall first be initialized and started before any other functionality is available. This is achieved by the sequence below:



After event T2BF_CALL_EVENT_NOT_CONNECTED, the framework can be used.

9.2 T2BF paired devices

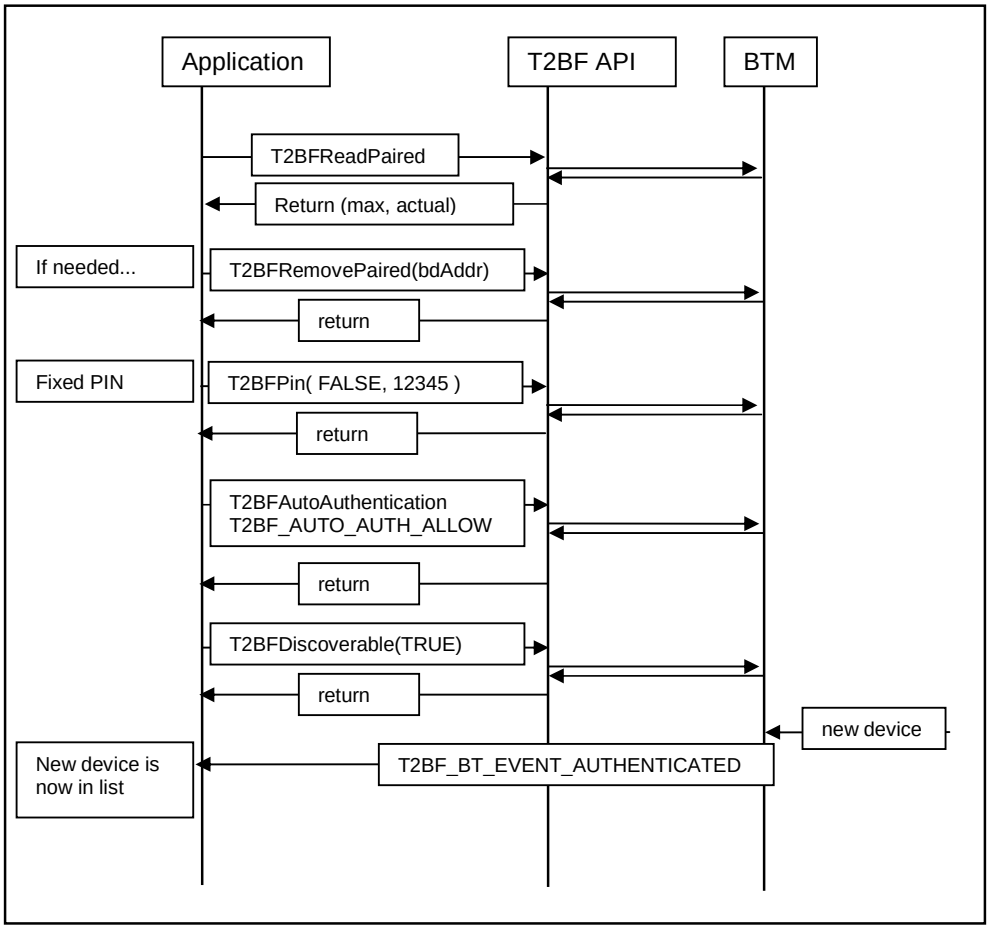
After being started, it's good to find all paired remote devices to decide which one to connect with. Paired info is kept in a list that can be sorted either by last paired or last connected device. This is selected by a Boolean in T2BFReadPairedInfo function.



9.3 T2BF new paired device

T2BF can be set to accept a new paired device. First there must be room in the paired list. Remove one if needed. Then turn pairing on to either a fixed PIN or authentication mode. When authentication mode is used (call T2BFAutoAuthentication with parameter T2BF_AUTO_AUTH_OFF), an event T2BF_BT_EVENT_AUTHENTICATE must be responded with a T2BFAuthenticate. Authentication mode is default after T2BF init and start.

Pairing can be completely turned off by calling T2BFAutoAuthentication with parameter T2BF_AUTO_AUTH_DENY.


Application
T2BF API
BTM
T2BFReadPaired
Return (max, actual)
If needed...
T2BFRemovePaired(bdAddr)
return
Fixed PIN
T2BFPin(FALSE, 12345)
return
T2BFAutoAuthentication
T2BF_AUTO_AUTH_ALLOW
return
T2BFDiscoverable(TRUE)
return
new device
New device is now in list
T2BF_BT_EVENT_AUTHENTICATED

10 References

[1] T2BF SW module header files: t2bf.h and btypes.h

[2] 104-1214 TSFP Specification