

Aculab Prosody™ 2 (TiNG)

FAX API guide

PROPRIETARY INFORMATION

The information contained in this document is the property of Aculab Plc and may be the subject of patents pending or granted, and must not be copied or disclosed without prior written permission. It should not be used for commercial purposes without prior agreement in writing.

All trademarks recognised and acknowledged.

Aculab Plc endeavours to ensure that the information in this document is correct and fairly stated but does not accept liability for any error or omission.

The development of Aculab products and services is continuous and published information may not be up to date. It is important to check the current position with Aculab Plc.

Copyright © Aculab plc. 2005: All Rights Reserved.

Document Revision

Rev	Date	By	Detail
4.0.0	11.09.05	PA	First full release draft
4.0.1	15.01.05	PA	Incorporate Peer review changes
4.0.2	16.05.07	PA	Deprecated library files
4.0.3	07.09.07	PA	Deprecated library files and modifications for new Fax 4
4.0.4	08.11.07	PA	Minor change

CONTENTS	
1	Introduction..... 5
2	Overviews 6
2.1	Firmware..... 6
2.1.1	FAX channel counts for Prosody cards 6
2.1.2	FAX channel count for Prosody X cards..... 6
2.2	API..... 7
3	API headers and libraries 9
3.1	Library components 9
3.1.1	Microsoft Windows Linking 9
3.1.2	Sun Solaris and Linux linking 9
4	API call descriptions..... 10
4.1	Fax session initialisation..... 10
4.1.1	SMFAX_GLOBAL_DATA..... 10
4.1.2	SMFAX_CAPS..... 11
4.1.3	SMFAX_TRANSPORT_MEDIUM_TDM 14
4.1.4	SMFAX_TRANSPORT_MEDIUM_VMP..... 14
4.1.5	SMFAX_TRANSPORT_MEDIUM_FMP 14
4.1.6	SMFAX_TRANSPORT_MEDIUM..... 15
4.1.7	SMFAX_DATA_TRANSPORT_PARMS..... 15
4.1.8	SMFAX_USER_OPTIONS 16
4.1.9	SMFAX_SESSION 18
4.1.10	smfax_lib_init..... 19
4.1.11	smfax_create_session 19
4.2	Fax session negotiation 20
4.2.1	SMFAX_NEGOTIATE_PARMS 20
4.2.2	smfax_negotiate 21
4.3	Fax page processing..... 22
4.3.1	SMFAX_PAGE_PROCESS_PARMS..... 22
4.3.2	smfax_tx_page..... 23
4.3.3	smfax_rx_page 24
4.4	Page Manipulation 25
4.4.1	SMFAX_PAGE_ACCESS_PARMS 25
4.4.2	smfax_need_conversion 26
4.4.3	smfax_load_page..... 27
4.4.4	smfax_load_convert_page 28
4.4.5	smfax_store_page 29
4.5	Post fax session cleanup 30
4.5.1	smfax_close_page 30
4.5.2	smfax_close_session..... 30
4.6	Fax session interruption 31
4.6.1	smfax_rude_interrupt..... 31
4.6.2	smfax_polite_intterupt..... 32
4.7	Fax activity trace – debugging..... 33
4.7.1	SMFAX_TRACE_PARMS..... 33
4.7.2	smfax_trace_on 34
4.7.3	smfax_trace_off 34
4.8	Miscellaneous..... 35
4.8.1	smfax_get_recommended_fax_caps 35
4.8.2	smfax_get_recommended_user_options..... 35
4.8.3	smfax_error_to_text..... 36
4.8.4	smfax_get_msgs..... 36
4.8.5	smfax_version 38
5	Fax On Demand – Polled Mode Fax 39
5.1	Polling fax terminal..... 39
5.2	Polled fax terminal 39
Appendix A: References 40	

Appendix B: Error/Return Codes41
Appendix C: Firmware.....44
 C.1.1 Firmware modules for Prosody cards..... 44
 C.1.2 Additional firmware modules specific to Prosody X cards 44
 C.1.3 Additional firmware modules specific to enabling the T.38 feature 44

1 Introduction

This document describes the version 4.0 Aculab Prosody Group 3 fax software application programmer interface (API). This documentation covers the following:

- The Aculab Generic Call Control API, [Appendix A:\[4\]](#), which describes management of telephony resources.
- The Aculab Switch Driver API, [Appendix A: \[3\]](#), which enables switching of streams between resources and CT buses.
- Prosody Version 2 API Guide, [Appendix A: \[1\]](#), which describes the Prosody Generic API and a host of other prosody related topics.
- The Aculab Image Processing API guide, [Appendix A: \[2\]](#), which describes the ACTIFF library. This library contains various image manipulation tools.

2 Overviews

2.1 Firmware

A number of Prosody 2 (TiNG) firmware files need to be downloaded to each media processing resource that is to be used for fax transmission and/or reception.

The Prosody 2 (TiNG) firmware files necessary for Prosody Fax are listed in [Appendix C](#).

For more comprehensive information on firmware, see [Appendix A: \[1\]](#).

The total channel count supported by each media processing resource in the system is dependant on the hardware architecture of the host system, the bus bandwidth available, and the performance of the local mass storage system.

2.1.1 FAX channel counts for Prosody cards

Please contact Aculab technical support for the latest developments on Prosody channel counts.

2.1.2 FAX channel count for Prosody X cards

Please contact Aculab technical support for the latest developments on Prosody X channel counts.

2.2 API

API Call	Description
<code>smfax_create_session</code>	Instantiation of fax session.
<code>smfax_negotiate</code>	Commences ITU-T T.30 fax negotiation
<code>smfax_tx_page</code>	Transmission of a fax page/image.
<code>smfax_rx_page</code>	Reception of a fax page/image.
<code>smfax_load_page</code>	Loading of an individual page from a group 3 compliant TIFF file into host computer volatile memory.
<code>smfax_store_page</code>	Storage of a received fax image to a group 3 compliant TIFF file from host computer volatile memory.
<code>smfax_polite_interrupt</code>	Graceful interruption of a fax session.
<code>smfax_rude_interrupt</code>	Almost instantaneous interruption of a fax session.
<code>smfax_close_page</code>	Release host OS resources taken up by received or loaded page.
<code>smfax_close_session</code>	Release host OS resources taken up by fax session.
<code>smfax_trace_on</code>	Turn on fax trace.
<code>smfax_trace_off</code>	Turn off fax trace.
<code>smfax_get_recommended_fax_caps</code>	Set certain structure members to default values.
<code>smfax_get_recommended_user_options</code>	Set certain structure members to default values.
<code>smfax_error_to_text</code>	Returns a string literal of the specified error code returned from the fax API.
<code>smfax_lib_init</code>	Deprecated: Allocate and initialise internal library memory structures.
<code>smfax_need_conversion</code>	Deprecated: Query ACTIFF file to see if specified page requires a conversion.
<code>smfax_load_convert_page</code>	Deprecated: Load a page and convert it to the required format.

To transmit a fax, a Prosody Fax enabled application would do the following:

- Establish a call using the Aculab call control API, see [Appendix A: \[4\]](#).
- Allocate a full duplex Prosody channel, see [Appendix A:\[1\]](#).
- The established call is switched to the Prosody channel, using the Aculab switch API, see [Appendix A: \[3\]](#).
- Invoke `smfax_create_session` specifying the necessary parameters.
- Invoke `smfax_negotiate`.
- If negotiation is successful call `smfax_load_page` to load a page from a Group 3 compliant TIFF file, previously opened with the Aculab ACTIFF API, see [Appendix A: \[2\]](#).
- Upon successful negotiation, invoke `smfax_tx_page` specifying the necessary parameters to transmit the loaded page. Remember to specify whether or not the page to be transmitted is the last or not.
- Invoke `smfax_close_page` when the page transmission API returns.
- {optional} If there are more pages to send load them like the previous page and then invoke the API to transmit the page.
- Invoke `smfax_close_session` to release resources used by the fax session.
- Free the allocated full duplex Prosody channel, if it has no further use.

To receive a fax, a Prosody Fax enabled application would do the following:

- Establish a call using the Aculab call control API, see [Appendix A: \[4\]](#).
- Allocate a full duplex Prosody channel, see [Appendix A:\[1\]](#).
- The established call is switched to the Prosody channel, using the Aculab switch API, see [Appendix A: \[3\]](#).
- Invoke `smfax_create_session` specifying the necessary parameters.
- Invoke `smfax_negotiate`.
- If negotiation is successful, invoke `smfax_rx_page`.
- Invoke `smfax_store_page` to store the received page/image to a group 3 compliant TIFF file, after `smfax_rx_page` returns.
- Invoke `smfax_close_page` when the page reception API returns and the page is stored using the relevant API, or when convenient to the program architecture.
- If `smfax_rx_page` indicated, on return, that there are more pages then `smfax_rx_page` must be called again and the page storage procedure must be subsequently repeated.
- Invoke `smfax_close_session` to release resources used by fax session.
- Free the allocated full duplex Prosody channel, if it has no further use.

3 API headers and libraries

The prototype declarations of API calls described in this document can be found in the 'C' language header file `smfaxapi.h`. This file calls upon the header files:

- `actiff.h` - interface to TIFF image accessing API, see [Appendix A:\[2\]](#).
- `smdrvr.h` - interface to generic Prosody API calls, see [Appendix A:\[1\]](#).

3.1 Library components

3.1.1 Microsoft Windows Linking

The library files (*.lib) for building a Prosody Fax enabled application can be found in the `API/lib` directory of the fax distribution. The dynamic link library files (`faxlib3_dll.dll` and `actiff.dll`) can be found in the `bin` subdirectory of the Aculab V6 installation directory.

Component	Description
<code>faxlib3_dll</code>	Aculab implementation of ITU-T T.30 recommendation.
<code>actiff</code>	API for accessing group 3 compliant TIFF files.

3.1.2 Sun Solaris and Linux linking

The library files for building a Prosody Fax enabled application are now distributed as shared objects. The shared object libraries (`libfaxlib.so` and `libactiff.so`) can be found in the `lib` subdirectory of the Aculab V6 installation directory.

Component	Description
<code>libfaxlib</code>	Aculab implementation of ITU-T T.30 recommendation.
<code>libactiff</code>	API for accessing group 3 compliant TIFF files.

4 API call descriptions

4.1 Fax session initialisation

Initialisation structures

4.1.1 SMFAX_GLOBAL_DATA

```
typedef struct tSMFaxGlobalData
{
    void * global_data;           /* out */
} SMFAX_GLOBAL_DATA;
```

Description

This structure gives access to meta data for internal use within a fax session.

- `global_data`
A pointer to an Aculab proprietary type. For internal use only.

Note DEPRECATED – This structure has been included for legacy purposes. The use of this structure has been deprecated.

4.1.2 SMFAX_CAPS

Definition

```
typedef struct tSmfaxCaps
{
    tSM_UT32    v17                :1;          /* in/out */
    tSM_UT32    v29                :1;          /* in/out */
    tSM_UT32    v27ter             :1;          /* in/out */
    tSM_UT32    MR2D               :1;          /* in/out */
    tSM_UT32    MMRT6              :1;          /* in/out */
    tSM_UT32    ECM                 :1;          /* in/out */
    tSM_UT32    Res200x200         :1;          /* in/out */
    tSM_UT32    polling_mode       :1;          /* in */
    tSM_UT32    fCappedDataRate    :1;          /* in */
    tSM_UT32    TIFFWidth          :2;          /* in/out */
    tSM_UT32    v8                  :1;          /* in */
    tSM_UT32    v34                 :1;          /* in */
    tSM_UT32    ECMSize64          :1;          /* in */
    tSM_UT32    NoRcvrOp           :1;          /* in */
    tSM_UT32    data_rate;          /* in/out */
    char        subscriber_id[kSMMaxSubscriberLen+1]; /* in */
    char        remote_id[kSMMaxSubscriberLen+1];    /* out */
} SMFAX_CAPS;
```

Description

This structure enables the user to specify the capabilities of the local fax terminal. The capabilities should reflect the features supported by the firmware installed on the media processing resource to be used for fax.

- *v17*
Set this "C" bit-field to 1 to enable support for ITU-T V17 modem. Leave as zero to disable. Please ensure that the necessary firmware modules have been loaded.
- *v29*
Set this "C" bit-field to 1 to enable support for ITU-T V29 modem. Leave as zero to disable. Please ensure that the necessary firmware modules have been loaded.
- *v27ter*
Set this "C" bit-field to 1 to enable support for ITU-T V27ter modem. Leave as zero to disable. Please ensure that the necessary firmware modules have been loaded.
- *MR2D*
Set this "C" bit-field to 1 to enable support for ITU-T T.4 Modified Read/Two-dimensional encoding of group 3 compliant TIFF data. Leave as zero to disable. No additional firmware required for this feature.
- *MMRT6*
Set this "C" bit-field to 1 to enable support for ITU-T T.6 Modified Modified Read encoding of group 3 compliant TIFF data. Leave as zero to disable. No additional firmware required for this feature.

Note If *MMRT6* is enabled then enabling *ECM* becomes mandatory.

- *ECM*
Set this "C" bit-field to 1 to enable support for ITU-T T.30 Annex A "Error Correction Mode" and ITU-T T.4 Annex A. There are no explicit firmware requirements for this feature.
- *Res200x200*
Set this "C" bit-field to 1 to enable support for high-resolution images. Resolution increase is on the Y-axis. No additional firmware is required for this feature.
- *polling_mode*
This field is used internally by the library. To specify polling/pollled modes please see `SMFAX_USER_OPTIONS` structure definition (in particular the description of the field *fax_mode*), later in this document. No additional firmware is required for this feature.

- fCappedDataRate*
 Setting this “C” bit-field to 1 will enable the data rate capping feature. The data rate capping feature works by limiting the chosen data rate, in calling or polled modes, to the value selected by setting the field *data_rate*, which is explained later. No additional firmware is required for this feature. This feature would typically be used when the line conditions for a particular target network are deemed unreliable at higher data rates.
- TIFFWidth*
 The value in this field will relate to the width of the facsimile image to be transmitted. This value is a request to the remote to comply with this width. However, the remote end does not have to agree to this specified width and is allowed to dictate the actual width to be used. No additional firmware is required for this feature.

Constant Designation	Facsimile Image Width (pels)
kSMFfaxTIFFWidth_A4	1728
kSMFfaxTIFFWidth_A3	2432
kSMFfaxTIFFWidth_B4	2048

- v8*
 Set this “C” bit-field to 1 to enable support for the ITU-T V8 procedures for starting sessions of data transmission over the public switched telephone network. Leave as zero to disable. No additional firmware is required for this feature.
- v34*
 Set this “C” bit-field to 1 to enable support for the ITU-T V34 modem. Leave as zero to disable. Please ensure that the necessary firmware modules have been loaded.
- ECMsize64*
 Reserved for future use.
- NoRcvrOp*
 Reserved for future use.
- data_rate*
 This field can take one of the following valid values to allow the user to select the speed at which the “calling” or “polled” session should first try to train. No additional firmware is required for this feature.

Valid Value	Supporting Modem			
	V27	V29	V17	V34
kSMFfaxDataRate_2400	✓			
kSMFfaxDataRate_4800	✓			✓
kSMFfaxDataRate_7200		✓	✓	✓
kSMFfaxDataRate_9600		✓	✓	✓
kSMFfaxDataRate_12000			✓	✓
kSMFfaxDataRate_14400			✓	✓
kSMFfaxDataRate_16800				✓
kSMFfaxDataRate_19200				✓
kSMFfaxDataRate_21600				✓
kSMFfaxDataRate_24000				✓
kSMFfaxDataRate_26400				✓
kSMFfaxDataRate_28800				✓
kSMFfaxDataRate_31200				✓
kSMFfaxDataRate_33600				✓
kSMFfaxDataRate_Default	Specifying this value will instruct the library to use the best data rate available, for the preferred modem. This is also the default behavior seen in previous major releases of Prosody Fax.			

On return, this field will hold one of the above values to indicate what baud rate was last used. This does not mean, however, that the entire fax call used this rate. Typically this would be the lowest data rate that was used in the fax call.

Note The “preferred modem” is the modem modulation agreed, by both terminals, at negotiation time, as the modulation to afford the best speed.

Note If both *MR2D* and *MMRT6* fields are disabled then, the ITU-T T.30 recommendation default, ITU-T T.4 Modified Huffman encoding is used.

- *subscriber_id*

A twenty digit string, typically the digital subscriber identifier. Valid characters are [0-9, + and space character]. If there is no valid subscriber id then it is advised that all twenty digits are set to the space character or the character '0'. Any unused digits should be set to the space character. No additional firmware is required for this feature.

- *remote_id*

A twenty digit string, typically the digital subscriber identifier of the remote device. The Prosody Fax library will internally set this field at negotiation time. No additional firmware is required for this feature.

4.1.3 SMFAX_TRANSPORT_MEDIUM_TDM

Definition

```
typedef struct tSmfaxTransportMedium_TDM
{
    tSMTDMrxId rx;                /* in */
    tSMTDMtxId tx;                /* in */
} SMFAX_TRANSPORT_MEDIUM_TDM;
```

Description

This structure allows the user to specify that TiNG TDM endpoints are to be used instead of the basic stream and time-slot TDM system. Please read the appropriate TiNG documentation (see [Appendix A: \[1\]](#)) on how to allocate and configure these types.

4.1.4 SMFAX_TRANSPORT_MEDIUM_VMP

Definition

```
typedef struct tSmfaxTransportMedium_VMP
{
    tSMVMPrxId rx;
    tSMVMPtxId tx;
} SMFAX_TRANSPORT_MEDIUM_VMP;
```

Description

This structure allows the user to specify that TiNG VMP endpoints are to be used instead of the basic stream and time-slot TDM system. Please read the appropriate TiNG documentation (see [Appendix A: \[1\]](#)) on how to allocate and configure these types.

4.1.5 SMFAX_TRANSPORT_MEDIUM_FMP

Definition

```
typedef struct tSmfaxTransportMedium_FMP
{
    tSMFMPrxId rx;
    tSMFMPTxId tx;
} SMFAX_TRANSPORT_MEDIUM_FMP;
```

Description

This structure allows the user to specify that TiNG FMP endpoints are to be used instead of the basic stream and time-slot TDM system. Please read the appropriate TiNG documentation (see [Appendix A: \[1\]](#)) on how to allocate and configure these types.

4.1.6 SMFAX_TRANSPORT_MEDIUM

Definition

```
typedef union tuSmfaxTransportMedium
{
    SMFAX_TRANSPORT_MEDIUM_TDM tdm;           /* in */
    SMFAX_TRANSPORT_MEDIUM_VMP vmp;         /* in */
    SMFAX_TRANSPORT_MEDIUM_FMP fmp;         /* in */
} SMFAX_TRANSPORT_MEDIUM;
```

Description

This union provides the user with a method for specifying one of many different forms of Prosody API endpoints.

- `tdm`
See 4.1.3 on how to configure this.
- `vmp`
See 4.1.4 on how to configure this.
- `fmp`
See 4.1.5 on how to configure this.

4.1.7 SMFAX_DATA_TRANSPORT_PARMS

Definition

```
typedef struct tSmfaxDataTransportParms
{
    int type;
    SMFAX_TRANSPORT_MEDIUM *transport;
} SMFAX_DATA_TRANSPORT_PARMS;
```

Description

This structure allows for backward compatibility, for when Prosody endpoints are not being used. However, it is highly recommended that developers become familiar with the use of Prosody endpoints.

- `type`
This field can take one of the following values.

Valid Value	Description
<code>kSMFAXTransportType_Classic</code>	Specify this value when endpoints are not being used.
<code>kSMFAXTransportType_TDM</code>	Specify this value when TDM endpoints are to be used.
<code>kSMFAXTransportType_VMP</code>	Specify this value when VMP endpoints are to be used.
<code>kSMFAXTransportType_FMP</code>	Specify this value when FMP endpoints are to be used.

- `transport`
This field must be set to NULL if `type` is set to `kSMFAXTransportType_Classic`. Otherwise, this field must point to a valid `SMFAX_TRANSPORT_MEDIUM` structure. One of the member fields of the `SMFAX_TRANSPORT_MEDIUM` structure must reflect the choice made by setting the `type` field, which is explained above.

4.1.8 SMFAX_USER_OPTIONS

Definition

```
typedef struct tUserOptions
{
    tSMIEEE32Bit754854Float max_percent_badlines; /* in */
    tSM_UT32 max_consec_badlines; /* in */
    tSM_UT32 max_modem_fb; /* in */
    tSM_UT32 page_retries; /* in */
    tSM_UT32 T38_ASN1_version; /* in */
    tSM_INT fax_mode; /* in */
    tSM_UT32 ecm_continue_to_correct :1; /* in */
    tSM_UT32 drop_speed_on_ctc :1; /* in */
    tSM_UT32 SkipToneGen :1; /* in */
    tSM_UT32 SkipSubsId :1; /* in */
    tSM_UT32 StrictT30 :1; /* in */
} SMFAX_USER_OPTIONS;
```

Description

This structure enables the user to control the way the library logic carries out the fax session.

- *max_percent_badlines*
Set this value to indicate the acceptable percentage of erroneous scan lines in a received page. This field has meaning to a “Called” or “Polling” terminal. The floating-point value 1.0 denotes one hundred percent. A value of 0.05 would indicate five percent.

Note The value for this field must be chosen with care. If the value is set too low for a noisy connection then frequent re-transmissions will be required, potentially increasing the transmission time. However, in an application where document accuracy is paramount, a high value could compromise the legitimacy of the document.

- *max_consec_badlines*
This field indicates the maximum number of consecutive erroneous scan lines to tolerate in a received page.

Note A library in “Called” or “Polling” mode would use *max_percent_badlines* and *max_consec_badlines* to rate the quality of the received document and subsequently request a retrain and/or retransmit.

- *max_modem_fb*
This field allows the user to specify how many times a “Calling” or “Polled” terminal should re-train to a slower modem speed or modulation, should this be possible.
- *T38_ASN1_version*
This field can take one of four possibly valid values. The values are

Value	Description
0	1998 ASN.1 syntax.
1	1998 ASN.1 syntax. TPKT not supported.
2	2002 ASN.1 syntax.
3	2002 ASN.1 syntax. V.34 and V.33 not supported.

See [Appendix A: \[5\]](#).

- *page_retries*
The number of times a specific page should be retransmitted, should there be a need.
- *fax_mode*
The terminal type of the local fax device. This field can take the following values.

Terminal Mode	Description
kSMFaxModeCalling	A fax terminal that makes a call and then sends one or more pages.
kSMFaxModeCalled	A fax terminal that answers a call and receives one or more pages.
kSMFaxModePolling	A fax terminal that makes a call and receives one or more pages.
kSMFaxModePolled	A fax terminal that answers a call and then sends one or more pages.

- *ecm_continue_to_correct*
 Set this "C" bit-field to 1 to enable the "Calling" or "Polled" terminal, during an ECM enabled fax call, to continue to re-transmit a partial page after the fourth partial page request (PPR). Leave as zero to continue with the next page (if any).
- *drop_speed_on_ctc*
 Set this "C" bit-field to 1 to enable the "Calling" or "Polled" terminal, during an ECM enabled fax call, to reduce the speed or modulation of the modem that transmits the page.
- *SkipToneGen*
 Reserved for future use.
- *SkipSubsId*
 Reserved for future use
- *StrictT30*
 Set this "C" bit-field to 1 to turn on a strict form of ITU-T T.30 recommendation. It is advised that this member be left as zero.

4.1.9 SMFAX_SESSION

Definition

```
typedef struct tSmfaxSession
{
    tSMModuleId          module;           /* in */
    tSMModuleId          module_id;       /* in */
    tSMChannelId         channel;         /* in */
    SMFAX_DATA_TRANSPORT_PARMS data_transport; /* in */
    SMFAX_USER_OPTIONS   user_options;    /* in */
    tSM_INT              exit_error_code; /* out */
    SMFAX_CAPS           fax_caps;        /* in */
    *global_data         *global_data;    /* in */
    tSM_UT32             fax_id;          /* in */
    SMFAX_TRACE_PARMS   *trace_parms;    /* in */
} SMFAX_SESSION;
```

Description

This is the primary interface structure to the Prosody FAX library.

- *module*
The media processing resource identifier that *channel* was allocated on, as returned by `sm_open_module()`. See [Appendix A: \[1\]](#).
- *module_id*
The media processing resource identifier that *channel* was allocated on, as returned by `sm_open_module()`. See [Appendix A: \[1\]](#).

Note If both *module* and *module_id* are non-zero then the value in *module* is used.

- *channel*
A valid full duplex Prosody channel allocated using the Prosody API. See [Appendix A: \[1\]](#).
- *user_options*
Options that specify local terminal mode and other assignable features. See [section SMFAX_USER_OPTIONS](#).
- *exit_error_code*
Upon termination of the fax session, see later, this field will give a numeric indication of the termination reason.
- *fax_caps*
The capabilities supported by this local terminal. After negotiation this field will indicate the capabilities used in the established fax session. See [section SMFAX_CAPS](#).
- *data_transport*
Allows the use of Prosody endpoints instead of the basic stream and timeslot TDM system. See 4.1.7.
- *global_data*
Is a pointer to the memory resource location returned by `smfax_lib_init`.

Note DEPRECATED – This field has been included for legacy purposes. The use of this field has been deprecated.

Debugging related fields

- *fax_id*
Assign this field as required by the application. This field is displayed in the debug trace to help identify trace statements belonging to a specific fax session.
- *trace_parms*
A pointer to a trace parameters structure. This allows tracing of the fax session creation process, which was not possible in previous major releases. If this pointer is not null, trace is turned on, else it stays off. The *fax_session* field of *trace_parms* is ignored in this use of `SMFAX_TRACE_PARMS` and can be set to null.

Initialisation API function

4.1.10 smfax_lib_init

Prototype Definition

```
int smfax_lib_init(SMFAX_GLOBAL_DATA * fax_global_data)
```

The parameter *fax_global_data* is a pointer to a structure of the following type:

```
typedef struct tSMFaxGlobalData
{
    void * global_data;
} SMFAX_GLOBAL_DATA;
```

Description

This function is to be called only once per application. Upon successful execution the pointer *fax_global_data*, will point to newly allocated memory resources used later on in a fax session.

For each subsequent fax session that is created the `SMFAX_SESSION` member *global_data* (described earlier) will point to this structure.

Note **DEPRECATED** – This function has been included for legacy purposes. The use of this function has been deprecated.

Returns

This function returns zero.

4.1.11 smfax_create_session

Prototype Definition

```
int smfax_create_session( SMFAX_SESSION *fax_session)
```

The argument *fax_session* is a pointer to a structure of type `SMFAX_SESSION`. See [section SMFAX_SESSION](#).

Description

This function is called to prepare the resources necessary for a fax session. Aspects such as terminal type and fax capabilities can be specified by interaction with [SMFAX_USER_OPTIONS](#) and [SMFAX_CAPS](#) members. See relevant sections above.

The function can return one of three values. Upon failure, the *exit_error_code* member of *fax_session* will provide a reason for the failure.

Returns

On success

`kSMFaxStateMachineRunning` Fax session created.

On failure

`kSMFaxStateMachineTerminated` A session was created but the state machine failed to run. Check the value of *exit_error_code*.

`kSMFaxNullPointer`

Both `module` and `module_id` fields of `SMFAX_SESSION` are zero, or the `channel` field of `SMFAX_SESSION` is equal to `kSMNullChannelId`

4.2 Fax session negotiation

Negotiation structures

4.2.1 SMFAX_NEGOTIATE_PARMS

Definition

```
typedef struct tSmfaxNegotiateParms
{
    SMFAX_SESSION                *fax_session;           /* in */
    ACTIFF_PAGE_PROPERTIES       *page_props;           /* out */
    tSM_INT                      is_polling;           /* out */
    SMFAX_CAPS                   sel_caps;             /* out */
} SMFAX_NEGOTIATE_PARMS;
```

Description

Once a fax session has been successfully created, this structure, along with its API function counterpart can be used to progress the fax session through the negotiation phase (ITU-T T.30 phase B).

- *fax_session*
A pointer to a valid [SMFAX_SESSION](#) structure, as returned by [smfax_create_session](#).
- *page_props*
Upon successful negotiation, this member will hold the agreed group 3 compliant TIFF image metrics (information such as image encoding and image resolution).
- *is_polling*
Upon successful negotiation, this member will be `kSMFaxPolling` if a polling/pollled session was agreed, otherwise it will be `kSMFaxNonPolling`.
- *sel_caps*
This field will be set to reflect the capabilities negotiated by the two fax terminals.

Negotiation API function

4.2.2 smfax_negotiate

Prototype Definition

```
int smfax_negotiate(SMFAX_NEGOTIATE_PARMS *fax_neg_params_);
```

The argument *fax_neg_params* is a pointer to a structure of type SMFAX_NEGOTIATE_PARMS. See section [SMFAX_NEGOTIATE_PARMS](#).

Description

This function will progress the fax session through the negotiation phase (ITU-T T.30 phase B). If the function returns successfully then the *is_polling* member of SMFAX_NEGOTIATE_PARMS should be interrogated before the next API function is called (the value of *is_polling* will dictate which function is to be called next).

If the value of *is_polling* is `kSMFaxNonPolling` then a non-polling/pollled fax session has been established. This would be the expected value if the *fax_mode* member of SMFAX_USER_OPTIONS were set to `kSMFaxModeCalling` or `kSMFaxModeCalled` prior to fax session creation.

A polling/pollled fax session has been established if the value of *is_polling* is `kSMFaxPolling`. This is expected if the value of *fax_mode* was set to `kSMFaxModePolling` or `kSMFaxModePolled` prior to fax session creation.

Note For more on polling and polled fax sessions see [Fax On Demand](#).

To decide which state machine API function to call next follow the rules tabulated below.

<i>is_polling</i>	<i>fax_mode</i>	Next API function
<code>kSMFaxNonPolling</code>	<code>kSMFaxModeCalling</code>	<code>smfax_tx_page</code>
<code>kSMFaxNonPolling</code>	<code>kSMFaxModeCalled</code>	<code>smfax_rx_page</code>
<code>kSMFaxPolling</code>	<code>kSMFaxModePolling</code>	<code>smfax_rx_page</code>
<code>kSMFaxPolling</code>	<code>kSMFaxModePolled</code>	<code>smfax_tx_page</code>

Aliases

Two aliases for this function have been defined in the fax API header file. They are, `smfax_tx_negotiate` and `smfax_rx_negotiate`. These aliases should ease the transition from the previous major release of the Aculab Prosody FAX API.

Returns

On success

`kSMFaxStateMachineRunning`

Negotiation was generally a success. The state machine awaits the next command.

On failure

`kSMFaxStateMachineTerminated`

Unsuccessful negotiation occurred. The state machine has been stopped.

Check the *exit_error_code* member of `SMFAX_SESSION` for a reason for this failure.

4.3 Fax page processing

Page processing structures

4.3.1 SMFAX_PAGE_PROCESS_PARMS

Definition

```
typedef struct tSmfaxPageProcessParms
{
    SMFAX_SESSION                *fax_session;           /* in */
    ACTIFF_PAGE_HANDLE           *page_handle;          /* in */
    tSM_INT                      is_last_page;         /* in */
    tSM_INT                      renegotiate;           /* out */
} SMFAX_PAGE_PROCESS_PARMS;
```

Description

Once a fax session has been created and negotiation has been successful; this structure can be used, in conjunction with the transmission API function, to transmit a previously loaded fax page; or, using the reception API function, receive a fax page. Both transmission and receive API functions will progress the fax session through the message procedure, ITU-T T.30 phase C.

- *fax_session*
A pointer to a valid [SMFAX_SESSION](#) structure as returned by [smfax_create_session](#).
- *page_handle*
A pointer to a [ACTIFF_PAGE_HANDLE](#) type.
- *is_last_page*
This member can be assigned one of two values and has meaning only when transmission is to take place. A fax session undertaking transmission will use the value of this member to control what handshake signals are sent once the fax page data has been transmitted.

<i>is_last_page</i>	
<i>kSMFaxNotLastPage</i>	The page to be transmitted is not the last page. The transmission API function will return allowing the API user to send more pages. This also means that another page must follow the current page.
<i>kSMFaxLastPage</i>	The page to be transmitted is the last page. The transmission API function will return once the terminate message has been sent. No further pages can be sent with this fax session.

- *renegotiate*
This member will indicate whether or not the negotiation API must be called again. A non-zero value indicates that the negotiation API must be called, a zero value indicates that the negotiation API need not be called. Once this value has been queried the user is advised to reset it to zero before continuing.

Fax page transmission

4.3.2 smfax_tx_page

Prototype Definition

```
int smfax_tx_page(SMFAX_PAGE_PROCESS_PARMS *fax_pp_parms_);
```

The argument *fax_pp_parms_* is a pointer to a structure of type SMFAX_PAGE_PROCESS_PARMS. See section SMFAX_PAGE_PROCESS_PARMS.

Description

This function will progress the fax session through the message procedure phase, ITU-T T.30 phase C.

Prior to invocation of this function, an appropriate Group 3 complaint TIFF file page should have been loaded using the page loading API. The SMFAX_PAGE_PROCESS_PARMS member *page_handle* should be a pointer to the loaded page.

Note After successful negotiation the TIFFWidth field of the sel_caps member of SMFAX_NEGOTIATE_PARMS structure should be queried to discover what image width has been agreed. A group 3 complaint TIFF file of such proportions should then be loaded.

The page loading must be done prior to calling this API function.

The member *is_last_page* should indicate the intention to send more pages or not, following the current page.

Returns

<code>kSMFaxStateMachineRunning</code>	Transmission was a success. The state machine is waiting for the next command. Typically, this API function would be called again with the next page to be sent.
<code>kSMFaxStateMachineTerminated</code>	Transmission may have been a success. No more invocations of the transmission or negotiation API can be made.

Note If the latest page to be sent was the last page then, on successful transmission, the transmission API function will return with *kSMFaxStateMachineTerminated*. However, the return of this code does not imply that the transmission was a success. The exit/error code member of *fax_session* should be interrogated for an explanation of why this API function returned.

Fax page reception

4.3.3 smfax_rx_page

Prototype Definition

```
int smfax_rx_page(SMFAX_PAGE_PROCESS_PARMS *fax_pp_parms);
```

The argument *fax_pp_parms* is a pointer to a structure of type SMFAX_PAGE_PROCESS_PARMS. See [section SMFAX_PAGE_PROCESS_PARMS](#).

Description

This function will progress the fax session through the message procedure phase, ITU-T T.30 phase C.

If the return code of this function indicates that the state machine is still running, then this API function must be called again. The remote terminal has more pages to send to this terminal.

The *renegotiate* member of SMFAX_PAGE_PROCESS_PARMS must be interrogated following the API function return. The *renegotiate* member will indicate whether or not the negotiation API must be called again. The software library described in this document is capable of calling the negotiation phase again, when necessary, without user intervention. However, this would mean that the user would miss the opportunity to store the recently received page.

If the *renegotiate* member indicates negotiation needs to take place, then the API user must call the negotiation API. Prior to calling the negotiation API, the user may invoke the page storage API to store the recently received page.

Returns

<code>kSMFaxStateMachineRunning</code>	Reception was a success. The state machine awaits the next command.
<code>kSMFaxStateMachineTerminated</code>	No more invocations of the reception or negotiation API can be made with this fax session. Check the exit/error code to ascertain if the last reception was a success or not.

4.4 Page Manipulation

Page manipulation structures

4.4.1 SMFAX_PAGE_ACCESS_PARMS

Definition

```
typedef struct tSmfaxPageAccessParms
{
    SMFAX_SESSION                *fax_session           /* in */
    ACTIFF_FILE                  *actiff;              /* in */
    ACTIFF_PAGE_HANDLE           *page_handle;         /* in/out */
    ACTIFF_PAGE_PROPERTIES       *page_props          /* unused */
    tSM_UT32                     page_index;         /* in */
} SMFAX_PAGE_ACCESS_PARMS;
```

Description

This structure allows the retrieval of an individual page from a Group 3 compliant TIFF into an Aculab proprietary type, the `ACTIFF_PAGE_HANDLE`. It also allows a page held in an instance of the Aculab proprietary type to be stored to a Group 3 compliant TIFF.

- `fax_session`
A pointer to a valid fax session, as returned by the fax session creation API.
- `actiff`
A valid pointer to an `ACTIFF_FILE` that has been opened for read or write, as returned by the appropriate Aculab Image Processing API. See [Appendix A: \[2\]](#).
- `page_handle`
A valid pointer to an `ACTIFF_PAGE_HANDLE`.
- `page_index`
The index in the Group 3 compliant TIFF of the page in which we are interested. Page indexing begins at 0. The first page in a TIFF has a page index of 0 (zero). In a TIFF with `N` pages the last page has a page index of $(N-1)$.

4.4.2 smfax_need_conversion

Prototype Definition

```
int smfax_need_conversion (SMFAX_PAGE_ACCESS_PARMS *fax_pa_parms_)
```

The argument *fax_pa_parms_* is a pointer to a structure of type SMFAX_PAGE_ACCESS_PARMS.

Description

Looks at page *page_index* in the TIFF file represented by *actiff*. If the required page is found, the properties of that page are compared with the properties outlined by the “page properties structure” *page_props*. Once the comparison is made, it will indicate if the page format is suitable for transmission or if it requires “On-the-fly” conversion to match the required format.

TIFF file page indexing begins at zero. Therefore, the first page has index 0 the second has an index of 1 and so forth.

This function has the task of preparing the *page_handle*, so it is able to hold TIFF page data.

If the return code indicates that a page matching the properties specified by *page_props* has not been found, then the function *smfax_load_convert_page* must be invoked next. When the return code indicates a match was found, the function *smfax_load_page* must be invoked next.

Note **DEPRECATED** – This function has been deprecated and has been included for backward compatibility only. It should be expected that this function will be withdrawn from the API. There is no replacement for this API function as it is no longer required.

Returns

<i>kSMFaxTranscodeNeeded</i>	The required page properties were not matched.
<i>kSMFaxTranscodeNotNeeded</i>	There is a page that is appropriately encoded.
<i>kSMFaxReadError</i>	If the page specified by <i>page_index</i> is not found.

4.4.3 smfax_load_page

Prototype Definition

```
int smfax_load_page(SMFAX_PAGE_ACCESS_PARMS *fax_pa_params_);
```

The argument *fax_pa_params_* is a pointer to a structure of type SMFAX_PAGE_ACCESS_PARMS. See section [SMFAX_PAGE_ACCESS_PARMS](#).

Description

This function allows the retrieval of an individual page from a Group 3 complaint TIFF into an Aculab proprietary type, the ACTIFF_PAGE_HANDLE.

The *page_handle* member of *fax_pa_params_* must point to an ACTIFF_PAGE_HANDLE that has been initialised to zero.

The *actiff* member of *fax_pa_params_* should point to an ACTIFF_FILE that has access to the desired Group 3 complaint TIFF, and have been opened for reading. See Aculab Image Processing API, [Appendix A:\[2\]](#).

The page to be loaded will be indicated by the *page_index* member.

This function is typically called when a TIFF page is to be transmitted to the remote fax device. It is imperative that this function is called prior to the page transmission API function.

Note A user must wait for the transmission API function to return before calling this API function on the next page to be sent, if any.

Returns

On success

kSMFaxPageOK

The specified page has been successfully loaded into the ACTIFF_PAGE_HANDLE.

On failure

kSMFaxNullPointer

The specified fax session pointer is null.

kSMFaxPageReadError

The specified ACTIFF_FILE pointer is invalid.

kSMFaxPageNotStored

The specified page was not found.

kSMFaxDCNRemoteIncompatible

The negotiation phase failed (or was not executed) or the system could not allocate memory for internal use.

kSMFaxPageWrongWidth

The specified page does not match the agreed width.

kSMFaxPageResourceError

Some internal fax state machine process yielded invalid data.

kSMFaxResourceAllocFailed

The system could not allocate memory resource for an internal type.

4.4.4 smfax_load_convert_page

Prototype

```
int smfax_load_convert_page( SMFAX_PAGE_ACCESS_PARMS *fax_pa_parms)
```

The argument *fax_pa_parms* is a pointer to a structure of type *SMFAX_PAGE_ACCESS_PARMS*. See section [SMFAX_PAGE_ACCESS_PARMS](#).

Description

The function *smfax_need_conversion* must be called before this function can be invoked.

This function loads a page from a TIFF file specified by the pointer *actiff*, this will be a valid pointer to an *ACTIFF_FILE* previously returned by a call to *actiff_read_open* (see [Appendix:\[2\]](#)).

This function converts the requested page from its existing format to the required resolution and encoding method. This means that the page is not already in a suitable format as required by the receiver and needs to be converted to the format specified by the *ACTIFF_PAGE_PROPERTIES* argument in the invocation of *smfax_need_conversion()*.

Note **DEPRECATED** – This function has been deprecated and has been included for backward compatibility only. It should be expected that the function will be withdrawn from this API. The function *smfax_load_page()* should be used instead.

Returns

kSMFaxPageOK

kSMFaxWriteError

kSMFaxResourceError

The page was loaded correctly.

Unable to write any data to *page_handle*.

Could not allocate required resources to proceed with loading of specified page.

4.4.5 smfax_store_page

Prototype Definition

```
int smfax_store_page(SMFAX_PAGE_ACCESS_PARMS *fax_pa_parms_);
```

The argument *fax_pa_parms_* is a pointer to a structure of type `SMFAX_PAGE_ACCESS_PARMS`. See section [SMFAX_PAGE_ACCESS_PARMS](#).

Description

This function allows the storage of a received fax image from the Aculab proprietary type (`ACTIFF_PAGE_HANDLE`) to a Group 3 compliant TIFF.

The *page_handle* member of `SMFAX_PAGE_ACCESS_PARMS` should point to the same location as the *page_handle* member of [SMFAX_PAGE_PROCESS_PARMS](#), as returned by the reception API function.

The *actiff* member of *fax_pa_parms_* should point to an `ACTIFF_FILE` that has access to the desired Group 3 complaint TIFF, and has been opened for writing. See Aculab Image Processing API, [Appendix A:\[2\]](#).

This function is typically called when this fax device has received a fax image. This function need not be called straight after the reception API has returned. It can be called once the final fax page has been received, providing each received page uses a separate `ACTIFF_PAGE_HANDLE`.

Returns

On success

`kSMFaxPageOK`

The specified page has been successfully stored from the `ACTIFF_PAGE_HANDLE` to the specified Group 3 TIFF file.

On failure

`kSMFaxNullPointer`
`kSMFaxPageWriteError`
`kSMFaxPageReadError`
`kSMFaxPageNotStored`

The specified fax session pointer is null
 The specified `ACTIFF_FILE` pointer is invalid.
 The specified page handle is a read only object.
 A null page handle was specified and the page was not stored.

4.5 Post fax session cleanup

4.5.1 smfax_close_page

Prototype Definition

```
int smfax_close_page(ACTIFF_PAGE_HANDLE *page_handle_);
```

The argument *page_handle_* is a pointer to a valid `ACTIFF_PAGE_HANDLE`.

Description

This function will release any resources taken up by either a fax image that has been received or a fax image that was loaded from persistent storage for transmission.

Typically, this function should be called once there is no longer any use for the specified `ACTIFF_PAGE_HANDLE`.

Returns

On success

`kSMFaxPageOK` The resources have been successfully released.

On failure

`kSMFaxPageResourceError` Internal processing error. A null temporary `ACTIFF_FILE` associated with this page handle.

`kSMFaxPageReadError` The page handle is marked as read only but there is no reader object associated with it.

`kSMFaxNullPointer` A null page handle was specified.

--- Other --- Error codes from the Prosody Fax Image Processing API error code set should be expected.

4.5.2 smfax_close_session

Prototype Definition

```
int smfax_close_session(SMFAX_SESSION *fax_session_);
```

The argument *fax_session_* must be a valid pointer to a previously created fax session, as returned by the session creation API function.

Description

This function will release any resources taken up by the fax session instance that is specified. The fax API user must avoid calling this function while the specified fax session is still in use. For example, the page transmission or page reception API is still operating on this fax session.

Once a fax session is closed in this way. No more API calls can be made on this fax session.

Returns

On success

0 The resources taken up by the fax session have been released.

On failure

`kSMFaxNullPointer` The specified fax session is invalid.

`kSMFaxERRORApiInCorrect` Returned if the specified fax session's state machine is still operating. Meaning, the fax session is still in use.

4.6 Fax session interruption

4.6.1 smfax_rude_interrupt

Prototype Definition

```
int smfax_rude_interrupt(SMFAX_SESSION *sfs_);
```

The argument `sfs_` must be a valid pointer to a previously created fax session, as returned by the session creation API function.

Description

This API function can be used to bring the internal fax state machine to a halt. Upon invocation of this function, a synchronous fax API function would be expected to return promptly.

This API function would typically be called when the call control API has indicated that there has been a remote disconnect event for this fax call.

Note This function may be called from a separate thread while the specified fax session is in use by one of the page processing or the negotiation API functions.

Returns

On success

0

The interrupt was successfully issued.

On failure

kSMFaxNullPointer

If the fax session associated with argument `sfs_` is null/invalid. Perhaps it has been closed or was never created.

--- Other ---

Return values of operating system dependant EVENT functions should be expected.

4.6.2 smfax_polite_intterrupt

Prototype Definition

```
int smfax_polite_interrupt(SMFAX_SESSION *sfs_);
```

The argument `sfs_` must be a valid pointer to a previously created fax session, as returned by the session creation API function.

Description

This API function can be used to bring the internal fax state machine to a graceful halt. Upon invocation of this function, a synchronous fax API function may return promptly.

This function is typically called when the return of the negotiation API function indicates that a different type of fax session has been established to that which is desired.

For example, a 'Called' fax session may have been indicated at session creation but the remote terminal instead has negotiated a 'Polling' mode fax session.

In this situation, this API function would be invoked prior to calling a blocking API function such as `smfax_rx_page` or `smfax_tx_page`.

Note This function maybe called from a separate thread while the specified fax session is in use by one of the page processing or negotiation API functions.

Returns

On success

0

The interrupt was successfully issued.

On failure

`kSMFaxNullPointer`

If the fax session associated with argument `sfs_` is null/invalid. Perhaps it has been closed or was never created.

--- Other ---

Return values of operating system dependant EVENT functions should be expected.

4.7 Fax activity trace – debugging

Trace manipulation structures

4.7.1 SMFAX_TRACE_PARMS

Definition

```
typedef struct tSmfaxTraceParms
{
    SMFAX_SESSION      *fax_session;           /* in */
    SMFAX_STATUS       *fax_status;           /* unused */
    tSM_INT             timeout;              /* unused */
    BFILE              *log_file;            /* in */
    tSM_UT32            trace_level;          /* in */
} SMFAX_TRACE_PARMS;
```

Description

This structure allows the manipulation of fax activity trace.

- *fax_session*
A valid fax session object as returned by the fax session creation API function.
- *log_file*
A valid BFILE object. As returned by the appropriate BFILE API. See [Appendix A:\[1\]](#). The BFILE object will typically be associated with a persistent file on a mass storage subsystem (i.e. a hard disk drive).
- *trace_level*
Is a mask which filters the type of trace going to *log_file*. This field can be a disjunction (bit-wise OR) of predefined constants found in the API header file.

Constant Designation	Description
TRACE_DEBUG	Low-level information. For developer use.
TRACE_WARNING	Incidents that may cause mischief but need not terminate the fax session.
TRACE_ERROR	Incidents that will bring the fax session to an end.
TRACE_STATE_MACHINE	Internal state machine activity.
TRACE_STATES	Report of incoming and outgoing handshake messages.
TRACE_API_CALLS	Report of all invoked API calls.
TRACE_ALL	Trace everything.

For example, to trace handshake messages, warnings and errors only; this field may be set as follows:

```
SMFAX_TRACE_PARMS stp;
...
...
stp.trace_level = TRACE_STATES | TRACE_WARNING | TRACE_ERROR;
```

The BFILE API, see [Appendix A:\[1\]](#), does not tie an API user down to tracing to a hard disk, the API is extensible enough to allow a user to overload specific functions to associate the 'file' with any storage medium they may require. So, for example, read and write may be performed on an IP socket.

Note On an Aculab Prosody Fax enabled system that may perform moderate to high fax channel counts it is advised that the storage medium to which the trace will be directed should have a low latency.

4.7.2 smfax_trace_on

Prototype Definition

```
int smfax_trace_on(SMFAX_TRACE_PARMS *trace_parms_);
```

The argument *trace_parms_* must be a valid pointer to a [SMFAX_TRACE_PARMS](#) structure.

Description

In the event where a fax session is not responding in the way it is expected to, it may be necessary to inspect what is going on underneath the API calls. In such an event, invoking *smfax_trace_on* will stream textual information to the file associated with `SMFAX_TRACE_PARMS.log_file`.

The *fax_session* field must point to a valid `SMFAX_SESSION` object as returned by the fax session creations API. Preferably the fax session for which the user intends to obtain trace.

Set the *trace_level* field, as previously described, to specify the trace filter level. The filter level need not remain the same throughout the lifetime of a fax session. Should a different level of detail be required, then this API can be called again with a different value for the *trace_level* field.

Returns

On success

0

The trace activation was a success.

On failure

`kSMFaxNullPointer`

The field *fax_session* and/or *log_file* are null.

--- Other ---

The BFILE API return codes should be expected.

4.7.3 smfax_trace_off

Prototype Definition

```
int smfax_trace_off(SMFAX_TRACE_PARMS * trace_parms_);
```

The argument *trace_parms_* must be a valid pointer to a [SMFAX_TRACE_PARMS](#) structure.

Description

If fax activity trace is no longer required before the end of the fax session, then this function can be called to stop trace from going to the file associated with *log_file*.

This API function is not mandatory, thus it need not be called prior to fax session clear up.

Returns

On success

0

The trace activation was a success.

On failure

`kSMFaxNullPointer`

The field *fax_session* is null. Or internal trace dispatcher was null.

4.8 Miscellaneous

4.8.1 `smfax_get_recommended_fax_caps`

Prototype Definition

```
int smfax_get_recommended_fax_caps(SMFAX_CAPS *fax_caps_);
```

The argument `fax_caps_` must be a valid pointer to a `SMFAX_CAPS` structure. More importantly, it must relate to the instance that is to be used in fax session creation.

Description

Explicitly sets certain fields to Aculab recommended values. These values may be overridden by the API user as need be.

This function does not explicitly set the `v34`, `v17`, `v29` and `v27(tertiary)` fields. These values should reflect the modem firmware installed on the media processing resource that is to be used for this fax session.

Returns

On success	0	Fields were set to default values.
On failure	<code>kSMFaxNullPointer</code>	The <code>fax_caps_</code> argument is null.

4.8.2 `smfax_get_recommended_user_options`

Prototype Definition

```
int smfax_get_recommended_user_options(SMFAX_USER_OPTIONS *user_opts_);
```

The argument `user_opts_` must be a valid pointer to a `SMFAX_USER_OPTIONS` structure. More importantly, it must relate to the instance that is to be used in fax session creation.

Description

Explicitly sets certain fields to Aculab recommended values. These values may be overridden by the API user as need be.

This function does not explicitly set the `fax_mode` field. The value for this field should reflect what mode of fax operation the API user desires.

Returns

On success	0	Fields were set to default values.
On failure	<code>kSMFaxNullPointer</code>	The <code>user_opts_</code> argument is null.

4.8.3 smfax_error_to_text

Prototype Definition

```
char * smfax_error_text(int err_);
```

The argument *err_* must be the error code that is to be mapped to a string literal.

Description

This function returns a string literal for the specified integer error code. The integer argument should typically be the *exit_error_code* field of *SMFAX_SESSION*.

The string literal set is limited to the error codes belonging to the fax library. String literals for Prosody error codes are not provided.

Returns

On success

Null terminated string

The error code is in the fax library error code set.

On failure

NULL

The error code scope is outside the fax library.

4.8.4 smfax_get_msgs

Prototype Definition

```
int smfax_get_msgs(SMFAX_GET_MSGS_PARMS *get_msg)
```

Where the parameter *get_msg* is a pointer to the following type:

```
typedef struct tSmfaxGetMsgsParms
{
    SMFAX_SESSION          *fax_session;
    SMFAX_MSGS             *msgs;
} SMFAX_GET_MSGS_PARMS;
```

Inputs

fax_session is a pointer to a structure as described in the “Fax Session Initialisation” section.

Outputs

msgs on return may point to a primitive linked-list of *SMFAX_MSGS*.

The member *msgs* is a pointer to the following type:

```
typedef struct tSmfaxMsgs
{
    int                msg_len;
    unsigned char      *msg;
    unsigned char      fSent;
    struct tSmfaxMsgs *next;
} SMFAX_MSGS;
```

Description

The fields in `SMFAX_MSGS` are all output fields and have the following meanings.

- `msg_len`
The length, in octets, of the array of unsigned chars pointed to by `msg`.
- `msg`
A pointer to an array of unsigned chars that holds one frame of the handshake octets exchanged during the specified fax call. The third array element is the facsimile control field. However, if this element holds a value of zero then this may be an ECM message and the fourth element, if any, should be treated as the facsimile control field. The 'C' header file `defs.h`, found in the Aculab fax distribution, contains constants that relate the hexadecimal number of the facsimile control field to ITU-T T.30 acronyms.
- `fSent`
If set to non-zero, indicates that the message in `msg` was transmitted by this terminal. Otherwise the message in `msg` was received from the remote fax terminal.
- `next`
If there were more messages exchanged during the lifetime of this fax call then `next` will point to the next message to have been exchanged.

This function will typically be called following a call to the negotiation function(s). It can also be called following calls to the page exchange functions, `smfax_rx_page()` and `smfax_tx_page()`.

Prior to the very first call to `smfax_get_msgs()`, `SMFAX_GET_MSGS_PARAMS.msgs` must point to `NULL`.

It must be noted that one or more messages may exhibit the same facsimile control field. This is not an error, but may be an indication that the message was sent more than once during the lifetime of this fax call.

This function may return with success but the message list may be empty. This would not strictly be an error. It may be the case that at the time this function was called there were no messages exchanged.

Returns

0	If the call was successful.
--- other ---	Error codes for system dependent mutex functions should be expected.

4.8.5 smfax_version

Prototype Definition

```
const char* smfax_version(SMFAX_VERSION_PARMS *verparms_)
```

Where the parameter *verparms_* is a pointer to the following type:

```
typedef struct tSmfaxVersionParms
{
    char                version_str[VERSION_STR_LEN];
}
SMFAX_VERSION_PARMS;
```

Inputs

None

Outputs

None

Description

This function allows a user to report the version number of the Aculab Prosody Fax API programmatically. The version number is returned as a null terminated string.

The argument *verparms_* has been included for future expansion.

Returns

The Fax API version number as a null terminated string.

5 Fax On Demand – Polled Mode Fax

In this mode of operation the fax terminal making the call intends to receive one or more pages of fax and the answering fax terminal will typically send one or more pages.

This facilitates fax services such as sports results and weather reports.

5.1 Polling fax terminal

The polling fax terminal is the fax terminal that makes the call and receives one or more pages of fax. The Aculab Prosody Fax library can operate in this mode.

To achieve a polling fax terminal an API user can do the following:

- Establish a call using the Aculab call control API, see [Appendix A:\[4\]](#).
- Allocate a full duplex Prosody channel, see [Appendix A:\[1\]](#).
- The established call is switched to the Prosody channel, using Aculab switch API, see [Appendix A: \[3\]](#).
- Invoke `smfax_create_session` specifying the necessary parameters. In particular, the `fax_mode` field of `SMFAX_USER_OPTIONS` must be set to `kSMFaxModePolling`.
- Invoke `smfax_negotiate`.
- If negotiation is successful and the value of the `is_polling` field of `SMFAX_NEGOTIATE_PARMS` is `kSMFaxPolling`, then `smfax_rx_page` should be called appropriately to receive one or more pages of fax.
- If the negotiation was a success but the value of `is_polling` is `kSMFaxNonPolling`, then either a page must be loaded and sent with `smfax_tx_page`, or `smfax_polite_interrupt` should be called, followed by a call to `smfax_tx/rx_page`, to send a disconnect fax message to the remote fax terminal and bring the fax session to an end.
- Clear up used resources by calling `smfax_close_page` and `smfax_close_session` appropriately.

5.2 Polled fax terminal

The polled fax terminal is the fax terminal that answers a call and sends one or more pages of fax. The Aculab Prosody Fax library can operate in this mode.

To achieve a polled fax terminal an API user can do the following:

- Establish a call using the Aculab call control API, see [Appendix A:\[4\]](#).
- Allocate a full duplex Prosody channel, see [Appendix A:\[1\]](#).
- The established call is switched to the Prosody channel, using Aculab switch API, see [Appendix A: \[3\]](#).
- Invoke `smfax_create_session` specifying the necessary parameters. In particular, the `fax_mode` field of `SMFAX_USER_OPTIONS` must be set to `kSMFaxModePolled`.
- If negotiation is successful and the value of the `is_polling` field of `SMFAX_NEGOTIATE_PARMS` is `kSMFaxPolling`, then `smfax_tx_page` should be called appropriately to send one or more pages of fax.
- If the negotiation was a success but the value of `is_polling` is `kSMFaxNonPolling`, then either this terminal must call `smfax_rx_page` to receive a fax, or `smfax_polite_interrupt` should be called, followed by a call to `smfax_tx/rx_page`, to send a disconnect fax message to the remote fax terminal and bring the fax session to an end.
- Clear up used resources by calling `smfax_close_page` and `smfax_close_session` appropriately.

Appendix A: References

- [1] Prosody Version 2 API Guide.
- [2] Prosody Fax Image Processing (ACTIFF) API
- [3] Aculab Switch Driver API
- [4] Aculab Call Control Driver API

All the above documents can be found online at <http://www.aculab.com/>. Navigate to Support | Technical documents | API guides | (Document you're looking for)

- [5] ITU-T T.38 (09/2005)

This document can be found on the ITU publications website <http://www.itu.int/publications/>

Appendix B: Error/Return Codes

These are the values that the `exit_error_code` field of `SMFAX_SESSION` may take upon fax termination.

Of the error codes described below, the `kSMFAXDCN...` codes can be found in the header file `smfaxapi.h`.

Error/Exit Code	Description
<code>kSMFAXDCNNormal</code>	Fax session came to an end with no failures.
<code>kSMFAXDCNCNGFailed</code>	Either the generation of the CNG tone failed or ITU-T V.8 procedures ended in failure.
<code>kSMFAXDCNCTCMaxTries</code>	ECM TX: The CTC command was sent 3 times following the fourth received PPR in the current sequence.
<code>kSMFAXDCNCTCNotEnabled</code>	ECM TX: The option to continue to correct is disabled. Encountered after the fourth PPR message to be received.
<code>kSMFAXDCNModemAbort</code>	Calling an interrupt API function terminated the fax session during a period where a modem may have been active.
<code>KSMFAXDCNModemError</code>	During a period of modem activity a failure occurred. Modem could not be started or stopped.
<code>KSMFAXDCNPageAccessDenied</code>	Fax session was unable to write received fax image data to the designated temporary memory resource for holding fax image data. Operating System Error.
<code>KSMFAXDCNPageQualityPoor</code>	The quality of the last fax image may be poor. Or during an ECM receive partial page requests were exchanged.
<code>KSMFAXDCNPageReadError</code>	Can occur when API user calls the page transmission API function without calling the page load API function.
<code>KSMFAXDCNPageRetryExceeded</code>	A specific fax image was resent N number of times without success. Where N is set using the <code>SMFAX_USER_OPTIONS</code> API structure.
<code>kSMFAXDCNRemoteAbrupt</code>	The remote terminal sent a DCN when we were least expecting it too.
<code>kSMFAXDCNRemoteIncompatible</code>	The remote fax terminal is not compatible. The remote end doesn't support any of the modems supported by the local terminal or is incapable of receiving/transmitting.
<code>kSMFAXDCNRemoteSentDCN</code>	The remote terminal sent us a DCN command when this terminal was expecting another command.
<code>kSMFAXDCNTimerExpired</code>	A nondescript timer expired. This is typically a timer specific to Aculab Prosody Fax.
<code>kSMFAXDCNTimerT1Expired</code>	Fax calls were connected but no messages were received from the remote within the duration of timer T1 (35 seconds approx).
<code>kSMFAXDCNTimerT2Expired</code>	If returned during the negotiation phase then the 1.5secs training check bytes did not arrive before this timer expired. If returned during the page reception phase then no fax image was exchanged before this timer expired.
<code>kSMFAXDCNTimerT4Expired</code>	No response was received in answer to sent handshake command.
<code>kSMFAXDCNTimerT5Expired</code>	ECM only. Local terminal waited for approximately sixty seconds for the remote terminal to become ready.

kSMFxDcNTrainCapsExhausted	The 1500mS training sequence was sent many times and there are no more data rates to try. Or a page was retransmitted many times and there are no more data rates to try.
kSMFxDcNTrainRetryExceeded	The 1500mS training sequence was sent repeatedly with no response from the remote.
kSMFxDcNUserInterrupt	Calling an interrupt API function terminated the fax session during a period where any modems were inactive.
kSMFxDcNUserOption	Typically, the number of times the modem modulation and/or speed has had to fall back is equal to the max_modem_fb field of SMFAX_USER_OPTIONS.
kSMFxDcNV21PacketUnexpected	A handshake command other than the one expected was received and the only course of action is to terminate the call.
kSMFxDcNV21ResendLimitHit	A handshake message was sent repeatedly with no response from the remote and this terminal is entitled to terminate the call.
kSMFxDcNERRORapiInCorrect	The user has called an API function that cannot be called at this time. Please revisit the API guide section on how to use the API.
kSMFxDcNResourceAllocFailed	Operating system was unable to allocate memory resource for fax library use.

Error/Exit Code	
kSMFxDcNCTCNoAnswer	These codes will no longer be returned by the API and have only been included to be compatible with existing code that uses them in 'switch' or 'if' clauses and expressions.
kSMFxDcNCTRNotReceived	
kSMFxDcNInvalidPageHandle	
kSMFxDcNNoDCNFromRemote	
kSMFxDcNQuitOnERR	
kSMFxDcNRRNoAnswer	
kSMFxDcNRemoteNotReady	
kSMFxDcNTrainWaitExpired	
kSMFxDcNTranscodeFailed	
kSMFxDcNV21PacketCRCBad	

API return codes	Description
kSMFaxNullPointer	An invalid pointer was passed to the API function that returns this code.
kSMFaxNoSessionCreated	The operating system was unable to allocate enough memory resource for a fax session object.
kSMFaxNoTIFFPageCreated	The operating system was unable to allocate enough memory resource for fax image storage.
kSMFaxResourceReleaseFailed	The releasing of a system resource failed.
kSMFaxWrongAPICall	No longer returned by this API. Included for backward compatibility.
kSMFaxStateMachineTerminated	The fax session state machine has been stopped. No further API calls can be made to the state machine.
kSMFaxStateMachineRunning	The fax session state machine is still running. Further API calls can be made to the state machine.
kSMFaxPageOK	Indicated successful operations on a fax page. These can be load, store or close operations.
kSMFaxPageWriteError	An invalid ACTIFF_FILE pointer was specified when trying to store a page.
kSMFaxPageReadError	This code can be returned when a write operation is carried out on a read only fax page. Or when an invalid ACTIFF_FILE pointer has been specified.
kSMFaxPageNotStored	Returned when user specified pointers appear valid but internal references are null. Indicates system failed to allocate some memory resource but not enough for all purposes.
kSMFaxPageResourceError	Internal error. The fax session has encountered problems with the agreed width. The width maybe out of range.
kSMFaxPageWrongWidth	When the width of the page to be loaded is not equal to the one in that was agreed.

Appendix C: Firmware

The table below details the firmware modules that are necessary to send and receive faxes under Prosody 2 (TiNG). The common files must be downloaded in order for fax to function regardless of the mode of operation.

For descriptions of the firmware modules listed below please see Appendix A: [1].

Note It is recommended that all receive or transmit files are downloaded so as to ensure compatibility with the majority of fax machines.

C.1.1 Firmware modules for Prosody cards

Common	Receive	Transmit
Inchan	V29rx	V17tx
Outchan	V27rx	V29tx
Fskrx		V27tx
Fsktx		
Fskpll		
Hdlcrx		
Hdlctx		
Syncrx		
Syncrx		
Six2five		
Tonegen		
Td		

C.1.2 Additional firmware modules specific to Prosody X cards

In addition to the modules listed above, the enhanced media processing resources on Prosody X cards can be loaded with the firmware module V17rx.

When ITU-T T.V34 functionality is desired the firmware modules v34hd, ansam and fskasyrx must be loaded on to all enhanced media processing resources that will be required to perform. Unlike the V27, V29 and V17 implementation there is only one firmware module that implements both the receiver and transmitter functionality.

C.1.3 Additional firmware modules specific to enabling the T.38 feature

In addition to the modules listed above, the enhanced media processing resources on Prosody X cards must be loaded with the firmware modules fmprx, fmptx, ifprx and ifptx to enable them to perform T.38 endpoint facsimile.