Aculab Prosody ™ 2 (TiNG)

FAX API guide

Revision 4.19.10

PROPRIETARY INFORMATION

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
4.11.0	03.06.09	RS	Updated for T.38 and new style
4.13.0	29.06.11	NK	Obtaining page properties for each received page
4.13.1	22.02.12	EBJ	Updated to corporate fonts and reformatted
4.17.1	22.05.12	NK	Updated for API extensions
4.18.0	27.07.12	NK	Updated for revised API
4.18.3	26.09.12	PGD	Additional Clarifications
4.18.4	05.11.12	PGD	Correct polling/polled description for revised API.
4.18.5	14.01.13	PGD	Extended detection_timeout, v.27 responder 2400 DIS
4.18.6	19.03.13	PGD	QA corrections
4.19.3	18.05.15	AHK	Added total_page_timeout in new page reception function for protection against endless retries
4.19.4	27.05.15	EBJ	Reformatted
4.19.7	16.10.15	PGD	Add more error codes
4.19.8	13.11.15	PGD	Align with faxlib 4.19.8, new error code
4.19.10	7.7.16	PGD	Additional trace flag

CONTENTS

1 Introduction

This document describes version 4.11 of the Aculab Prosody Group 3 fax software application programmer interface (API). The document does not cover 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.
- TiNG API Guide, Appendix A: [1], which describes the Prosody TiNG Generic API and a host of other TiNG related topics.

For the Aculab Image Processing API guide please see Appendix A: [2]. This library contains various image manipulation tools.

2 Overviews

2.1 Firmware

A number of Prosody 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 TiNG firmware files necessary for Prosody Fax are listed in Appendix C.

For more comprehensive information on firmware, see Appendix A: [1]

2.1.1 FAX channel counts for Prosody cards

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

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

3 Notes on T.30 over IP

It is becoming quite common for TDM calls (voice and data) to be routed over an IP link at some point. This has some implications for a data transfer such as fax. Below are some points to be aware of if you know that your T.30 fax is going to be sent via an IP link.

The IP link must be “fax aware” and default to use the G.711 codec only, no other codecs can be used.

Packet loss concealment must be off. IP uses packet loss concealment to mask missing packets in speech. But this is detrimental to a data transmission such as fax.

Comfort noise must be off. Comfort noise is used to mask digital silence which can be uncomfortable during a conversation, and can even lead to a participant thinking that he has been disconnected. But introducing noise in a fax transmission is a bad idea.

Echo cancellation must be off. Echo cancellers can modify the signal which must not happen during a data transmission.

Automatic gain control must be off. This can modify the signal, which is not allowed.

Adaptive jitter buffers must be off. Adaptive jitter buffers can alter the timing of packets and may introduce delays, fax relies on packets arriving promptly.

Ideally, there should be no packet loss, but minimal packet loss can be tolerated. If packet loss is going to occur it is recommended to minimise the size of image data chunks that are sent. Use ECM as that will send partial pages rather than whole pages and use higher compression techniques (MMR instead of MR). Also, if the packet loss is spaced quite far apart, it would be better to send the fax quickly, so use a faster modem (V.17 rather than V.27).

4 The API

API Call	Description
smfax_create_session	Instantiation of fax session.
smfax_configure	Configure the fax session.
smfax_negotiate_caps	Commences ITU-T T.30 fax negotiation.
smfax_tx_page_process	Transmission of a fax page/image.
smfax_rx_page_process	Reception of a fax page/image.
smfax_rx_page_process_with_retry_timeout	Reception of a fax page with timeout against endless retries
smfax_load_page	Loading of an individual page from a group 3 compliant TIFF file into host computer volatile memory.
smfax_store_page	Storage of a received fax image to a group 3 compliant TIFF file from host computer volatile memory.
smfax_close_page	Release host OS resources taken up by received or loaded page.
smfax_close_session	Release host OS resources taken up by fax session.
smfax_polite_interrupt	Graceful interruption of a fax session.
smfax_rude_interrupt	Almost instantaneous interruption of a fax session.
smfax_trace_on	Turn on fax trace.
smfax_trace_off	Turn off fax trace.
smfax_get_msgs	Obtain V.21 HDLC messages.
smfax_version	Returns the fax library version.
smfax_error_to_text	Returns a string literal of the specified error code returned from the fax API.

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_configure` to configure the session.
- Invoke `smfax_negotiate_caps`.
- 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_process` specifying the necessary parameters to transmit the loaded page. Remember to specify whether or not the page to be transmitted is the last page.
- Invoke `smfax_close_page` when the page transmission API returns.
- 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.
- If it has no further use, free the allocated full duplex Prosody channel.

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 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_configure` to configure the session.
- Invoke `smfax_negotiate_caps`.
- If negotiation is successful, invoke `smfax_rx_page_process` or `smfax_rx_page_process_with_retry_timeout`
- After `smfax_rx_page_process` or `smfax_rx_page_process_with_retry_timeout` returns, invoke `smfax_store_page` to store the received page/image to a group 3 compliant TIFF file.
- 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_process` or `smfax_rx_page_process_with_retry_timeout` indicated, on return, that there are more pages, then `smfax_rx_page_process` or `smfax_rx_page_process_with_retry_timeout` must be called again and the page storage procedure must be repeated.
- Invoke `smfax_close_session` to release resources used by fax session.
- If it has no further use, free the allocated full duplex Prosody channel.

5 API headers and libraries

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

5.1 Library components

5.1.1 Microsoft Windows Linking

The library files (faxlib3_dll.lib and actiff.lib) for building a Prosody Fax enabled application can be found under 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
`faxlib3_dll`	Aculab implementation of ITU-T T.30 recommendation.
`actiff`	API for accessing group 3 compliant TIFF files.

5.1.2 Linux linking

The library files for building a Prosody Fax enabled application are 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
`libfaxlib`	Aculab implementation of ITU-T T.30 recommendation.
`libactiff`	API for accessing group 3 compliant TIFF files.

6 API call descriptions

6.1 Fax session creation

6.1.1 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.

6.1.2 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.

6.1.3 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.

6.1.4 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 6.1.1 on how to configure this.

vmp
See 6.1.2 on how to configure this.

fmp
See 6.1.3 on how to configure this.

6.1.5 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
kSMFaxTransportType_Classic	Specify this value when endpoints are not used.
kSMFaxTransportType_TDM	Specify this value when TDM endpoints are used.
kSMFaxTransportType_VMP	Specify this value when VMP endpoints are used.
kSMFaxTransportType_FMP	Specify this value when FMP endpoints are 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.

6.1.6 SMFAX_SESSION

Definition

```
typedef struct tSmfaxSession
{
tSMModuleId             module;                 /* in */
tSMModuleId             module_id;              /* deprecated */
tSMChannelId            channel;                /* in */
SMFAX_DATA_TRANSPORT_PARMS data_transport;      /* in */
void                    *faxsession_inst;       /* internal */
SMFAX_USER_OPTIONS      user_options;           /* deprecated */
tSM_INT                 exit_error_code;        /* out */
SMFAX_CAPS              fax_caps;               /* deprecated */
SMFAX_GLOBAL_DATA       *global_data;           /* deprecated */
tSM_UT32                fax_id;                 /* deprecated */
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].

channel
A valid full duplex Prosody channel allocated using the Prosody API. See Appendix A: [1].

data_transport
Allows the use of Prosody endpoints instead of the basic stream and timeslot TDM system.

faxsession_inst
For internal library use only. Do not change.

exit_error_code
Upon termination of the fax session (see later) this field will give a numeric indication of the termination reason.

trace_parms
A pointer to a trace parameters structure. This allows tracing of the fax session creation process. If this pointer is not null trace is turned on, else it stays off. The *fax_session* field of *trace_parms* is currently ignored and should be set to null.

6.1.7 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.

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 through a subsequent call to smfax_configure (setting SMFAX_USER_CONFIG and SMFAX_CAPABILITIES members).

The function can return one of four 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 successfully.

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`*

6.2 Fax session configuration

6.2.1 SMFAX_CAPABILITIES

Definition

```
typedef struct tSmfaxCapabilities
{
     tSM_UT32       v17                 :1;  /* in/out */
     tSM_UT32       v29                 :1;  /* in/out */
     tSM_UT32       v27ter              :1;  /* in/out */
     tSM_UT32       fCappedDataRate     :1;  /* in */

     tSM_UT32       v8                  :1;  /* in */
     tSM_UT32       v34                 :1;  /* in */

     tSM_UT32       data_rate;               /* in/out */
     tSM_UT32       min_data_rate            /* in */

     tSM_UT32       ECM                 :1;  /* in/out */
     tSM_UT32       ECMsize64           :1;  /* in */

     tSM_UT32       MR2D                :1;  /* in/out */
     tSM_UT32       MMRT6               :1;  /* in/out */
     tSM_UT32       Res200x200          :1;  /* in/out */
     tSM_UT32       TIFFWidth           :2;  /* in/out */

     tSM_UT32       polling_mode        :1;  /* in/out */

     char       subscriber_id[kSMMaxSubscriberLen+1]; /* in */
     char       remote_id[kSMMaxSubscriberLen+1];     /* out */
} SMFAX_CAPABILITIES;
```

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 V.17 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 V.29 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 V.27ter modem. Leave as zero to disable. Please ensure that the necessary firmware modules have been loaded.

fCappedDataRate
If V.17 has been enabled, setting this “C” bit-field to 1 will make V.17 7200 pbs fall back to V.27 4800 bps bypassing V.29. If neither V.17 nor V.29 has been enabled, setting this “C” bit-field to 1 will cause a fax responder to indicate V.27ter fall-back mode and thus request for V.27 that only the 2400 bps rate be used

v8
Set this “C” bit-field to 1 to enable support for the ITU-T V.8 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. Required for V.34.

v34
Set this “C” bit-field to 1 to enable support for the ITU-T V.34 modem. Leave as zero to disable. Please ensure that the necessary firmware modules have been loaded. V.34 requires V.8 and ECM.

data_rate
This field can take one of the following valid values to allow the user to select the maximum 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			
	V.27	V.29	V.17	V.34
kSMFaxDataRate_2400	✓			
kSMFaxDataRate_4800	✓			✓
kSMFaxDataRate_7200		✓	✓	✓
kSMFaxDataRate_9600		✓	✓	✓
kSMFaxDataRate_12000			✓	✓
kSMFaxDataRate_14400			✓	✓
kSMFaxDataRate_16800				✓
kSMFaxDataRate_19200				✓
kSMFaxDataRate_21600				✓
kSMFaxDataRate_24000				✓
kSMFaxDataRate_26400				✓
kSMFaxDataRate_28800				✓
kSMFaxDataRate_31200				✓
kSMFaxDataRate_33600				✓
kSMFaxDataRate_Default	Specifying this value will instruct the library to use the best data rate available for the preferred modem.			

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 will use this rate. Typically this would be the highest data rate that is used in the fax call.

The “preferred modem” is the modem modulation agreed by both terminals at negotiation time. 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.

min_data_rate
If this field is non-zero, return an error if the image transfer data rate ever drops below it.

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.

ECMsize64
Supported for fax receive only.

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

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.

TIFFWidth
The value in this field is the preferred 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)
kSMFaxTIFFWidth_A4	1728
kSMFaxTIFFWidth_A3	2432
kSMFaxTIFFWidth_B4	2048

polling_mode
If set to zero polling will not be negotiated and if set to one polling will be negotiated. The `polling_mode` of `sel_caps` member of `SMFAX_NEGOTIATE_CAPS_PARMS`, after the negotiation will indicate if polling is negotiated or not.

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.

6.2.2 SMFAX_USER_CONFIG

Definition

```
typedef struct tUserConfig
{
   tSMIEEE32Bit754854Float max_percent_badlines;   /* in */
   tSM_UT32       max_consec_badlines;             /* in */
   tSM_UT32       max_modem_fb;                    /* in */
   tSM_UT32       page_retries;                    /* in */
   tSM_INT        fax_mode;                        /* in */
   tSM_UT32       idle_timeout;                    /* in */
   tSM_UT32       ecm_continue_to_correct :1;      /* in */
   tSM_UT32       drop_speed_on_ctc       :1;      /* in */
} SMFAX_USER_CONFIG;
```

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. This parameter is not relevant when using ECM mode. 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. This parameter is not relevant when using ECM mode. 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 retrain to a slower modem speed or modulation, should this be possible.

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
kSMFaxModeTransmitter	A fax terminal that makes a call and then sends one or more pages.
kSMFaxModeReceiver	A fax terminal that answers a call and receives one or more pages.

ilde_timeout
When there is a continuous inactivity for this duration the fax will be terminated. If set to zero not used. If set to a value less than 35000ms rounded up to 35000ms. Otherwise used as specified up to the default timer 3600000ms.

ecm_continue_to_correct
Set this "C" bit-field to 1 to allow 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 allow the "Calling" or "Polled" terminal (during an ECM enabled fax call) to reduce the speed or modulation of the modem after the fourth PPR.

6.2.3 SMFAX_T38_SDP_OPTIONS

Definition

```
typedef struct tT38SDPOptions
{
 tSM_UT32       T38FaxVersion;                /* in */
 tSM_UT32       T38FaxRateManagement;         /* in */
 tSM_UT32       T38MaxBitRate;                /* in */
 tSM_UT32       T38FaxUdpEC;                  /* in */
 tSM_UT32       T38FaxMaxDatagram;            /* in */
 tSM_UT32       T38FaxMaxBuffer;              /* in */
 tSM_UT32       T38FaxFillBitRemoval    :1;   /* in */
 tSM_UT32       T38FaxTranscodingMMR    :1;   /* in */
 tSM_UT32       T38FaxTranscodingJBIG   :1;   /* in */
} SMFAX_T38_SDP_OPTIONS;
```

Description

T.38 options negotiated during call set-up. See Appendix A: [5].

T38FaxVersion
This field can take one of four values.

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.

T38FaxRateManagement
Two possible values, 1 for local TCF, and 2 for transferred TCF.

T38MaxBitRate
If negotiated, given priority over *SMFAX_CAPABILITIES.data_rate*.

T38FaxUdpEC
Two possible values, 1 for UDP FEC and 2 for UDP Redundancy (UDP FEC is currently not supported).

T38FaxMaxDatagram
Maximum T.38 payload in bytes.

T38FaxMaxBuffer
Not significant for Aculab T.38 IAF end-point implementation.

T38FaxFillBitRemoval
Not supported.

T38FaxTranscodingMMR
If enabled, *SMFAX_CAPS.MMRT6* and *SMFAX_CAPS.ECM* will be enabled.

T38FaxTranscodingJBIG
Not supported.

6.2.4 SMFAX_CONFIG_PARMS

Definition

```
typedef struct tSMfaxConfig
{
   SMFAX_SESSION           *fax_session;      /* in */
   SMFAX_CAPABILITIES      *capabilities;     /* in */
   SMFAX_USER_CONFIG       *user_config;      /* in */
   SMFAX_T38_SDP_OPTIONS   *t38_sdp;          /* in */
} SMFAX_CONFIG_PARMS;
```

Description

Once a fax session has been successfully created, this structure, along with its API function, smfax_configure should be used to configure the session.

fax_session
A pointer to a valid SMFAX_SESSION structure, as returned by smfax_create_session.

capabilities
A pointer to a valid SMFAX_CAPABILITIES structure, allocated and filled by the application.

user_config
A pointer to a valid SMFAX_USER_CONFIG structure, allocated and filled by the application.

t38_sdp
A pointer to a valid SMFAX_T38_SDP_OPTIONS structure, allocated and filled by the application. Must be present only for T.38 (kSMFaxTransportType_FMP or kSMFaxTransportType_EXT_T38), otherwise set to NULL pointer.

6.2.5 smfax_configure

Prototype Definition

```
int smfax_configure(SMFAX_CONFIG_PARMS *fax_config_parms_);
```

The argument *fax_config_parms* is a pointer to a structure of type SMFAX_CONFIG_PARMS.

Description

This function will configure the terminal capabilities, other user configurable options, and T.38 SDP values (only for a T.38 session).

On success

kSMFaxStateMachineRunning	Fax session configured successfully.

On failure

kSMFaxBadParam	Bad parameter was detected.
kSMFaxNullPointer	Transport medium was T.38 but ***t38_sdp*** was set to null pointer.

6.3 Fax session negotiation

6.3.1 SMFAX_NEGOTIATE_CAPS_PARMS

Definition

```
typedef struct tSmfaxNegotiateCapsParms
{
     SMFAX_SESSION               *fax_session;          /* in */
     ACTIFF_PAGE_PROPERTIES      *page_props;           /* in/out */
     SMFAX_CAPABILITIES          *sel_caps;             /* in/out */
     tSM_UT32                    detection_timeout;     /* in */
     tSM_UT32                    negotiation_timeout;   /* in */
} SMFAX_NEGOTIATE_CAPS_PARMS;
```

Description

Once a fax session has been successfully created and configured, this structure, along with its API function, smfax_negotiate_caps 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

This member must be allocated by the application, and the address passed in. If this member is not required, it must be set to NULL. If used after negotiation this member can be queried for the agreed group 3 compliant TIFF image metrics (information such as image encoding and image resolution). The application must set this to a valid pointer or NULL.

sel_caps
This member must be allocated by the application, and the address passed in. If this member is not required, it must be set to NULL. If used, after the negotiation it will be set to reflect the capabilities negotiated by the two fax terminals. ***polling_mode*** member will be non-zero if a polling or polled session was negotiated.

detection_timeout
Only supported by the originator. If set a fax tone or a V.21 modem signal from the other terminal prior to expiry will cancel this. If expired the fax will be terminated. The valid range of values starts from 5000ms. If set to zero unused. Values less than 5000 are rounded up to 5000. When used to stimulate call cleardown in time shorter than normal T.30 35s timeout, T1 will remain at its standard setting. When used with value greater than normal T1 to accommodate peer system that plays voice anouncement prior to switching to fax, initial T1 will be set to this detection timeout value.

negotiation_timeout
Introduced due to the limitation that T1 would be cancelled after any valid V.21 HDLC packet, leading to lengthy unsuccessful negotiations. If expired the fax will be terminated. The valid range of values is 60000ms to the default timer 3600000ms. If set to zero unused. Values less than 60000 are rounded up to 60000.

6.3.2 smfax_negotiate_caps

Prototype Definition

```
int smfax_negotiate_caps(SMFAX_NEGOTIATE_CAPS_PARMS *fax_neg_parms_);
```

The argument *fax_neg_parms_* is a pointer to a structure of type SMFAX_NEGOTIATE_CAPS_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 *polling_mode* member of *sel_caps* member of SMFAX_NEGOTIATE_CAPS_PARMS should be interrogated before the next API function is called, the value of *polling_mode* will dictate which function is to be called next).

If the value of *polling_mode* is zero then a non-polling/non-polled fax session has been established. This would be the expected value if the *polling_mode* member of SMFAX_CAPABILITIES was set to zero in smfax_configure.

A polling/polled fax session has been established if the value of *polling_mode* is non-zero. This is expected if the value of *polling_mode* member of SMFAX_CAPABILITIES was set to non-zero in smfax_configure.

NOTE

For more on polling and polled fax sessions see Fax On Demand.

polling_mode	*fax_mode*	Next API function
0	kSMFaxModeTransmitter	smfax_tx_page_process
1	kSMFaxModeTransmitter	smfax_rx_page_process
0	kSMFaxModeReceiver	smfax_rx_page_process
1	kSMFaxModeReceiver	smfax_tx_page_process

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.

6.4 Fax page processing

6.4.1 SMFAX_TX_PAGE_PROCESS_PARMS

Definition

```
typedef struct tSmfaxTxPageProcessParms
{
     SMFAX_SESSION           *fax_session;  /* in */
     tSM_INT                 is_last_page;  /* in */
     SMFAX_CAPABILITIES      *sel_caps;     /* in/out */
     tSM_UT32                page_timeout;  /* in */
} SMFAX_TX_PAGE_PROCESS_PARMS;
```

Description

Once the negotiation has been successful, this structure can be used with `smfax_tx_page_process` to transmit a previously loaded fax page using `smfax_load_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`.

`is_last_page`
This member can be assigned one of two values. 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.

`is_last_page`	
`kSMFaxNotLastPage`	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.
`kSMFaxLastPage`	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.

`sel_caps`
This member must be allocated by the application, and the address passed in. If this member is not required, it must be set to NULL. If used, after the page transmission it will be set to reflect the capabilities used by the two fax terminals.

page_timeout
In milliseconds, the maximum time allowed for a continuous image data transmission or reception. If expired the fax will be terminated. The valid range of values is 60000ms to the default timer 3600000ms. If set to zero unused. Values less than 60000 are rounded up to 60000. Any fax control signal restarts this timer at the start of the image data transmission or reception.

6.4.2 smfax_tx_page_process

Prototype Definition

```
int smfax_tx_page_process(SMFAX_TX_PAGE_PROCESS_PARMS *fax_pp_parms_);
```

The argument *fax_pp_parms_* is a pointer to a structure of type SMFAX_TX_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 compliant TIFF file page should have been loaded using `smfax_load_page`. After successful negotiation the *TIFFWidth* field of the *sel_caps* member of *SMFAX_NEGOTIATE_CAPS_PARMS* structure should be queried to discover what image width has been agreed. A group 3 compliant 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

`kSMFaxStateMachineRunning`	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.
`kSMFaxStateMachineTerminated`	Transmission may have been a success. No more invocations of the transmission or negotiation API can be made.

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.

A perfect transmission (indicated by peer fax sending T.30 MCF) will result in `exit_error_code` of *kSMDCNNormal*

A successful but imperfect transmission (indicated by peer fax sending T.30 RTP) will result in `exit_error_code` of *kSMDCNPageQualityPoor*

If after several retries/renegotiations (indicated by peer fax sending T.30 RTN) required quality still not achieved, a failed transmission will result in `exit_error_code` of *kSMDCNPageRetryExceeded*

6.4.3 SMFAX_RX_PAGE_PROCESS_PARMS

Definition

```
typedef struct tSmfaxRxPageProcessParms
{
   SMFAX_SESSION           *fax_session;  /* in */
   ACTIFF_PAGE_HANDLE      *page_handle;  /* in/out */
   tSM_INT                 renegotiate;   /* out */
   ACTIFF_PAGE_PROPERTIES  *page_props;   /* in/out */
   SMFAX_CAPABILITIES      *sel_caps;     /* in/out */
   tSM_UT32                page_timeout;  /* in */
} SMFAX_RX_PAGE_PROCESS_PARMS;
```

Description

Once negotiated, this structure can be used with `smfax_rx_page_process` to 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
The address of an ACTIFF_PAGE_HANDLE type allocated by the user application. This will be required by `smfax_close_page` after the page is stored using `smfax_store_page`.

renegotiate
This member will indicate whether or not the negotiation API must be called again. A non-zero value indicates that the negotiation API i.e. `smfax_negotiate_caps` must be called, a zero value indicates that the negotiation API must not be called.

page_props
This member must be allocated by the application, and the address passed in. If this member is not required, it must be set to NULL. For each page received this member can be queried for information such as the number of bad lines and the negotiated page properties.

sel_caps
This member must be allocated by the application, and the address passed in. If this member is not required, it must be set to NULL. If used, after the page reception it will be set to reflect the capabilities used by the two fax terminals.

page_timeout
In milliseconds, the maximum time allowed for a continuous image data transmission or reception. If expired the fax will be terminated. The valid range of values is 60000ms to the default timer 3600000ms. If set to zero unused. Values less than 60000 are rounded up to 60000. Any fax control signal restarts this timer at the start of the image data transmission or reception.

6.4.4 smfax_rx_page_process

Prototype Definition

```
int smfax_rx_page_process(SMFAX_RX_PAGE_PROCESS_PARMS *fax_pp_parms_);
```

The argument *fax_pp_parms_* is a pointer to a structure of type SMFAX_RX_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_RX_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 (T.30 EOM). The API user must call smfax_store_page to store recently received page. After storing the page call smfax_negotiate_caps if *renegotiate* is non zero.

On the occasions where retraining was required due to insufficiently good page quality (T.30 RTN) the fax library will retrain without user intervention. In that case the smfax_rx_page_process will return once a page has been successfully received (note the original rejected "too poor quality" page will not be available to application and not included in page count).

Returns

kSMFaxStateMachineRunning	Reception was a success. The state machine awaits the next command.
kSMFaxStateMachineTerminated	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.

Notes – when the return code is kSMFaxStateMachineTerminated an application normally would only decide to store a page if page is complete and peer fax has received confirmation of successful (but possibly imperfect) transmission. This is the case for the following exit/error codes:

kSMFaxDCNNormal

kSMFaxDCNNoDCNFromRemote

kSMFaxDCNPageQualityPoor

However some applications may wish to allow access to partially transferred pages, this is possible in non-ECM mode when the ***page_props*** field ***image_length*** is non-zero. It can be determined whether ECM mode was used through interrogation of the ***sel_caps*** field ***ECM***.

6.4.5 SMFAX_RX_PAGE_PROCESS_WITH_RETRY_TIMEOUT_PARMS

Definition

```
typedef struct tSmfaxRxPageProcessWithRetryTimeoutParms
{
   SMFAX_SESSION           *fax_session;  /* in */
   ACTIFF_PAGE_HANDLE      *page_handle;  /* in/out */
   tSM_INT                 renegotiate;   /* out */
   ACTIFF_PAGE_PROPERTIES  *page_props;   /* in/out */
   SMFAX_CAPABILITIES      *sel_caps;     /* in/out */
   tSM_UT32                page_timeout;  /* in */
   tSM_UT32                total_page_timeout;     /* in */
} SMFAX_RX_PAGE_PROCESS_WITH_RETRY_TIMEOUT_PARMS;
```

Description

Once negotiated, this structure can be used with `smfax_rx_page_process_with_retry_timeout` to receive a fax page. This structure is the same as `SMFAX_RX_PAGE_PROCESS_PARMS` except it has one additional member.

`total_page_timeout`
In milliseconds, the maximum elapsed time allowed for a page reception including retry attempts. If set to zero unused. If expired, the fax will be terminated. See Appendix E.2 for more details.

6.4.6 smfax_rx_page_process_with_retry_timeout

Prototype Definition

```
int smfax_rx_page_process_with_retry_timeout(
SMFAX_RX_PAGE_PROCESS_WITH_RETRY_TIMEOUT_PARMS *fax_pp_parms_);
```

The argument *fax_pp_parms_* is a pointer to a structure of type SMFAX_RX_PAGE_PROCESS_WITH_RETRY_TIMEOUT_PARMS.

Description

This function is identical to `smfax_rx_page_process` except an additional ***`total_page_timeout`*** parameter is provided. For more details on use of this timeout, see Appendix E.2.

6.5 Page Manipulation

6.5.1 SMFAX_PAGE_ACCESS_PARMS

Definition

```
typedef struct tSmfaxPageAccessParms
{
     SMFAX_SESSION           *fax_session;      /* in */
     ACTIFF_FILE             *actiff;           /* in */
     ACTIFF_PAGE_PROPERTIES  *page_props;       /* deprecated */
     ACTIFF_PAGE_HANDLE      *page_handle;      /* in/out */
     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*. This field is set by `smfax_load_page` and `smfax_rx_page_process`, and used only by `smfax_close_page`

- *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*).

6.5.2 smfax_load_page

Prototype Definition

```
int smfax_load_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 SMFAX_PAGE_ACCESS_PARMS.

Description

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

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

The *actiff* member of *fax_pa_parms_* should point to an *ACTIFF_FILE* that has access to the desired Group 3 compliant TIFF, and has 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. The application 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.

6.5.3 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.

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_RX_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 compliant 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	The specified fax session pointer is null
kSMFaxPageWriteError	The specified ACTIFF_FILE pointer is invalid.
kSMFaxPageReadError	The specified page handle is a read only object.
kSMFaxPageNotStored	A null page handle was specified and the page was not stored.

6.6 Post fax session clearup

6.6.1 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.

6.6.2 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, if 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 that the fax session is still in use.

6.7 Fax session interruption

6.7.1 smfax_rude_interrupt

Prototype Definition

```
int smfax_rude_interrupt(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 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 is typically called when the call control API has indicated that there has been a remote disconnect event for this fax call.

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 dependent EVENT functions should be expected.

6.7.2 smfax_polite_interrupt

Prototype Definition

```
int smfax_polite_interrupt(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 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_process* or *smfax_tx_page_process*.

This function may be 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`	The fax session associated with argument *`sfs_`* is null or invalid. Perhaps it has been closed or was never created.
`Other`	Return values of operating system dependent EVENT functions should be expected.

6.8 Fax activity trace – debugging

6.8.1 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_IMAGE_DATA	Trace image data for some modes
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.

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 has a low latency.

6.8.2 smfax_trace_on

Prototype Definition

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

The argument *fax_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. This needs to be 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* is null.
Other	The BFILE API return codes should be expected.

6.8.3 smfax_trace_off

Prototype Definition

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

The argument *fax_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.

6.9 Miscelleneous

6.9.1 smfax_error_to_text

Prototype Definition

```
char * smfax_error_to_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.

6.9.2 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_process()` and `smfax_tx_page_process()`.

Prior to the very first call to `smfax_get_msgs()`, `SMFAX_GET_MSGS_PARMS.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 even though the message list is empty. This will happen if 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.

6.9.3 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;
```

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.

7 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.

7.1 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` then `smfax_configure` specifying the necessary parameters. In particular, the *`fax_mode`* field of *`SMFAX_USER_CONFIG`* must be set to *`kSMFaxModeTransmitter`* and the *`polling_mode`* field of *`SMFAX_CAPABILITIES`* should be set to a value of 1. Next invoke `smfax_negotiate_caps`.

If negotiation is successful and the value of the *`sel_caps->polling_mode`* field of *`SMFAX_NEGOTIATE_CAPS_PARMS`* is non-zero, then `smfax_rx_page_process` or `smfax_rx_page_process_with_retry_timeout` should be called appropriately to receive one or more pages of fax.

If the negotiation was a success but the value of *`polling_mode`* is zero, then either a page must be loaded and sent with `smfax_tx_page_process`, or `smfax_polite_interrupt` should be called, followed by a call to `smfax_tx/rx_page_process` or `smfax_rx_page_process_with_retry_timeout`, in order 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.

7.2 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` then `smfax_configure` specifying the necessary parameters. In particular, the *`fax_mode`* field of *`SMFAX_USER_OPTIONS`* must be set to *`kSMFaxModeReceiver`* and the *`polling_mode`* field of *`SMFAX_CAPABILITIES`* should be set to a value of 1. Next invoke `smfax_negotiate_caps`.

If negotiation is successful and the value of the *`sel_caps->polling_mode`* field of *`SMFAX_NEGOTIATE_CAPS_PARMS`* is non-zero, then `smfax_tx_page_process` should be called appropriately to send one or more pages of fax.

If the negotiation was a success but the value of *`polling_mode`* is zero, then either this terminal must call `smfax_rx_page_process` or

`smfax_rx_page_process_with_retry_timeout` to receive a fax, or `smfax_polite_interrupt` should be called, followed by a call to `smfax_tx/rx_page_process` or `smfax_rx_page_process_with_retry_timeout`, in order 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`.

Appendix A: References

[1] TiNG 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: Exit and 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.

Exit Code	Description
kSMFaxDCNNormal	Fax session came to an end with no failures.
kSMFaxDCNCNGFailed	Generation of the CNG tone failed.
kSMFaxDCNCTCMaxTries	ECM TX: The CTC command was sent 3 times following the fourth received PPR in the current sequence.
kSMFaxDCNCTCNotEnabled	ECM TX: The option to continue to correct is disabled. Therefore ending the call after the fourth PPR is received.
kSMFaxDCNNoDCNFromRemote	RX sent and repeated MCF for the final page, but did not receive DCN.
kSMFaxDCNPageQualityPoor	The quality of the final page may be poor.
kSMFaxDCNPageReadError	API user called the page transmission function without calling the page load function or the page load function failed.
kSMFaxDCNPageRetryExceeded	In non-ECM a specific fax page was resent N number of times without success, where N is set using the *SMFAX_USER_OPTIONS* API structure. In ECM CTC was sent N number of times.
kSMFaxDCNRemoteAbrupt	The remote terminal sent a premature DCN, e.g. during negotiation.
kSMFaxDCNRemoteIncompatible	The capabilities negotiation failed because a common mode of operation could not be established with the remote, e.g. supported modems and/or receiver vs. transmitter.
kSMFaxDCNRemoteSentDCN	RX received a DCN after page data, but may not be the final page. Any complete pages received are saved.
kSMFaxDCNTimerExpired	A nondescript timer expired. This is typically Aculab Prosody Fax default timer.
kSMFaxDCNTimerT1Expired	Fax calls were connected but no messages were received from the remote within the duration of timer T1 (35 seconds approx).
kSMFaxDCNTimerT2Expired	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.
kSMFaxDCNTimerT4Expired	No response was received in answer to sent

Exit Code	Description
	handshake command.
kSMFaxDCNTimerT5Expired	ECM only. Local terminal waited for approximately sixty seconds for the remote terminal to become ready.
kSMFaxDCNTrainCapsExhausted	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.
kSMFaxDCNTrainRetryExceeded	The 1500mS training sequence was sent repeatedly with no response from the remote.
kSMFaxDCNUserInterrupt	Calling an interrupt API function terminated the fax session during a period where any modems were inactive.
kSMFaxDCNUserOption	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*.
kSMFaxDCNV21PacketUnexpected	Received a control command that can not be interpreted in the current mode, e.g. CTC in V.34.
kSMFaxDCNV21ResendLimitHit	A handshake message was sent repeatedly with no response from the remote and this terminal is entitled to terminate the call.
kSMFaxERRORApiInCorrect	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.
kSMFaxResourceAllocFailed	Operating system was unable to allocate memory resource for fax library use.
kSMFaxDCNCTCNoAnswer	Deprecated and no longer used by the library. Included for backward compatibility. Recommended reviewing and updating the application code that includes these.
kSMFaxDCNCTRNotReceived	
kSMFaxDCNInvalidPageHandle kSMFaxDCNModemAbort kSMFaxDCNModemError	
kSMFaxDCNPageAccessDenied	
kSMFaxDCNQuitOnERR	
kSMFaxDCNRRNoAnswer	
kSMFaxDCNRemoteNotReady	
kSMFaxDCNTrainWaitExpired	
kSMFaxDCNTranscodeFailed	
kSMFaxDCNV21PacketCRCBad	

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.
kSMFaxDCNMinDataRateReached	Bit rate would drop below minimum rate permitted.
kSMFaxDCNT38PacketUnexpected	Expected T.38 V.21 packet but got packet with different protocol

API return codes	Description
kSMFaxDCNTiNGOtherError	Unexpected error from TiNG API
kSMFaxDCNTiNGNoFirmware	Required firmware not loaded (or feature unsupported by PS)
kSMFaxDCNTiNGNoResources	Out of resource used by TiNG – if Linux possibly hit file descriptor limit for system
kSMFaxDCNTiNGWrongState	TiNG API reporting channel found in inappropriate state
kSMFaxDCNBadInternalParam	Internal parameter out of range (V.34 bit rate)
kSMFaxDCNBadChannelState	Channel in unexpected state
kSMFaxDCNBadModemState	Expected modem to be idle but wasn't
kSMFaxDCNNoToneSet	Problem with setting up tone detection
kSMFaxDCNWaitError	Error on waiting for event
kSMFaxDCNWrongStatus	Error in sequencing of TiNG status values
kSMFaxDCNV8Failed	Error during V.8 negotiation
kSMFaxDCNEmptyList	Internal error no T.30 packet pending
kSMFaxDCNT38V34PacketUnexpected	V.34 related T.38 packet received when not expected
kSMFaxDCNT38V34BadRate	V.34 related T.38 bit rate invalid
kSMFaxDCNDetTimerExpired	CED/DIS detection timeout
kSMFaxDCNInternalError	Unspecified internal error occurred
kSMFaxDCNV21PacketBad	Invalid T.30 indicator in received V.21 packet
kSMFaxDCNCodeNotSet	Result code has not yet been set by library.
kSMFaxDCNRTNExceeded	Sequence of peer selected RTN rates illogical and would lead to endless loop
kSMFaxDCNShortECMBlockExceeded	Peer invalid partial use of non-final ECM block detected.
kSMFaxDCNLateResumeAmbiguity	In non-ECM mode, disconnect as, due to late scheduling on overloaded system, it was not possible to safely disambiguate retransmit request type.

Appendix C: Firmware

The table below details the firmware modules that are necessary to send and receive faxes under Prosody 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].

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

C.1 Firmware modules for Prosody cards

Common	Receive	Transmit
inchan	v17rx	v17tx
outchan	v29rx	v29tx
fskrx	v27rx	v27tx
fsktx	v34hd	v34hd
fskpll		
hdlcrx		
hdlctx		
syncrx		
synctx		
six2five		
tonegen		
td		
datafeed		

C.2 Firmware modules specific to V.34 and Fax over RTP

When V.34 functionality is desired (also implies use of V.8) the firmware modules `v34hd`, `ansam`, `prefsuf` and `fskasyrx` must be loaded on to all the applicable media processing resources. Unlike the V.27, V.29 and V.17 implementations there is only one firmware module for V.34 which implements both the receiver and transmitter. When the transport medium is VMP then the firmware modules `vmptx`, `vmprx` will be required.

C.3 Additional firmware modules specific to enabling T.38

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`, `ifptx` to enable them to perform T.38 endpoint facsimile. For applications that will be doing a re-invite, `vmprx` and `vmptx` will also be required.

Appendix D: Deprecated functions and data structures

Some API calls from earlier versions of faxlib are now deprecated, they are listed here for reference. For new applications the, more general, API calls in API call descriptions should now be used, for example invoking smfax_configure is more flexible and preferred to using the deprecated fields in smfax_create_session.

D.1 Deprecated functions

API Call
smfax_lib_init
smfax_create_session_t38
smfax_negotiate
smfax_rx_negotiate
smfax_tx_negotiate
smfax_negotiate_with_detection_timeout
smfax_negotiate_with_timers
smfax_tx_page
smfax_tx_page_with_timer
smfax_rx_page
smfax_rx_page_get_props
smfax_rx_page_with_timer
smfax_need_conversion
smfax_load_convert_page
smfax_wait_for_status
smfax_get_status
smfax_get_recommended_fax_caps
smfax_get_recommended_user_options

D.2 Deprecated data sturctures

API Call
SMFAX_GLOBAL_DATA
SMFAX_USER_OPTIONS
SMFAX_CAPS
SMFAX_NEGOTIATE_PARMS
SMFAX_NEGOTIATE_TIMEOUT_PARMS
SMFAX_NEGOTIATE_TIMERS_PARMS
SMFAX_PAGE_PROCESS_PARMS
SMFAX_PAGE_PROCESS_PROPS_PARMS
SMFAX_PAGE_PROCESS_TIMER_PARMS
SMFAX_STATUS

Appendix E: Use Of Timeouts

A number of timeout facilities are provided for fax sessions. This appendix gives more information on their use.

E.1 Fax Origination

For a fax originator, it might not be known in advance if the peer entity being connected is actually a fax. The ***detection_timeout*** facility that can be set in smfax_negotiate_caps allows an application to cleardown a call if there is no sign of a fax machine responding (with a fax tone or V.21 data) within the specified period.

Once it has been determined there is a peer fax responding entity, the maximum elapsed time for the negotiation phase can be limited using the ***negotiation_timeout*** (again set in `smfax_negotiate_caps`). This timer is in addition to the T.30 specified T1 negotiation timer, and can be used to curtail a stuck negotiation where nevertheless V.21 messages are still being exchanged.

Following negotiation, and during page transmission, the ***page_timeout*** parameter to smfax_tx_page_process may be used to limit maximum page transmission time. Note this only limits the time transmitting high speed image data between retries, if page retries or T.30 ECM PPRs occur, the timeout will be reset for each new page or partial page transmission attempt.

Finally an ***idle_timeout*** (set for session through smfax_configure) allows earlier abandonment of fax transmission if it appears peer entity has ceased participating in fax session (for example if line goes silent but call has not been cleared).

E.2 Fax Reception

As for fax origination, the maximum elapsed time for the negotiation phase can be limited using the ***negotiation_timeout*** (set in smfax_negotiate_caps).

Following negotiation and during page reception, the ***page_timeout*** parameter in smfax_rx_page_process or smfax_rx_page_process_with_retry_timeout may be used to limit maximum page reception time. Note this only limits the time receiving high speed image data between retries, if page retries or T.30 ECM PPRs occur, the timeout will be reset for each new page or partial page reception attempt.

An additional retry timeout called ***total_page_timeout*** is provided in smfax_rx_page_process_with_retry_timeout which allows the total reception time including retries to be limited. This allows protection against a misbehaving originating fax that never abandons its retry sequence. If this timeout is used, it should be set to a large enough value to allow for an originator to legitimately fall back through modem speeds (a value in the order of 60 minutes might be reasonable in most environments).

As for fax origination an ***idle_timeout*** (again set for session through smfax_configure) allows earlier abandonment of fax transmission if it appears that the peer entity has ceased participating in the fax session (for example if the line goes silent but the call has not been cleared).