Aculab Prosody™ 2 (TiNG)

T.38 Gateway API guide

Revision 1.0

aculab

Document Revision

Rev	Date	By	Detail
0.0.1	13/09/06	PJB	Initial documentation for early beta release
0.0.2	19/09/06	PJB	Update for additional status information
0.0.3	12/06/07	DC	Added ECM disable.
0.0.4	13/06/07	DC	Corrected 'T38GW_TDM_MODEM'.
	27/06/07	DC	Changed status 'tSMT38JobStatusStopped' to 'tSMT38JobStatusTerminated' for 'sm_t38gw_destroy_job()'. Added LINUX section.
0.0.5	17/08/07	DC	Addition to 5.1.5
0.0.6	14/05/08	DC	Added static link library option to 3.1
0.0.7	18/06/08	DC	Added T38 Gateway trace section.
0.0.8	26/09/08	DC	Trace input and output data logging.
0.0.9	20/01/09	DC	Removed 'subject to change' from 2.2.
0.1.0	07/04/09	DC	Highlighted sections in destroy session/job.
0.1.1	24/02/11	DC	Added fsk tx power.
0.1.2	24/05/11	DC	Added reference to T.38 Gateway introduction.
0.1.3	22/06/11	DC	Added 'sm_t38gw_destroy_job_info' and 'information field'.
0.1.4	04/07/11	DC	Added 'spoofing disable' for g57.
0.1.5	19/08/11	DC	Added 32/64 for g58.
0.1.6	29/04/13	PGD	Added max packet length call and ".so" ref
1.0	20/05/13	EBJ	Updated to corporate fonts

CONTENTS

1 Introduction

This guide is designed to assist developers using the T.38 gateway API. The T.38 gateway API is a high-level approach to performing the required conversion between a T.30 device on the public switch telephone network and a T.38 device on an IP network. This guide covers the basic concepts used in the T.38 gateway API and provides a description of the available API functions.

This document should be read in conjunction with the Aculab T.38 Gateway introduction [4] and the Aculab T.38 Gateway logging description [5].

2 Overviews

2.1 Supported T.38 modes

The Aculab T.38 implementation supports T.38 over UDP using the facsimile UDP transport layer (UDP/TL) only.

The V.34 modem type is not supported.

2.2 T.38 and the Session Initiation Protocol (SIP)

The ITU T.38 recommendation [2] defines several T.38 specific parameters that may be negotiated using the Session Description Protocol (SDP) during the establishment of a SIP call. The table below lists a subset of these parameters and indicates suitable values for use with the Aculab T.38 implementation. Please refer to [4] [2] for descriptions of the parameters.

SDP Parameter	Supported Values
T38FaxVersion	0, 1 and 2
T38MaxBitRate	14400,12000,9600,7200,4800 and 2400 *
T38FaxRateManagement	TransferredTCF
T38FaxMaxBuffer	2048 (max)
T38FaxMaxDatagram	[4]
T38FaxUdpEC	t38UDPRedundancy

NOTE

*The T38MaxBitRate parameter should be set to the fastest speed that can be sustained over the T.30 (see [3]) side of a call. This value will vary based on which firmware modems have been loaded.

2.3 API Concepts

The T.38 gateway API allows a T.30 fax originating on the public switched telephone network to be transferred over an IP network to a T.38 fax device. The transfer can also occur in the opposite direction with a T.38 fax device as the originator.

The T.38 gateway is organised in terms of controller tasks, referred to as sessions and individual gateway tasks, referred to as jobs. There may be multiple sessions at a given time and each session is capable of managing multiple jobs. The user is required to provide four Prosody end-points when creating a job. Each end-point must be correctly connected to the outside world using the Prosody 2 TiNG API[1]. End-points must not be reconfigured once control of a job has been handed to the T.38 gateway API.

2.3.1 End-points

The Prosody 2 TiNG [1] defines three distinct end-point types:

- TDM
    - TDM[tx] Sends data to a TDM stream and timeslot
    - TDM[rx] Receives data from a TDM stream and timeslot
- VMP**
    - VMP[tx] Sends data using RTP to a remote device
    - VMP[rx] Receives data using RTP from a remote device
- FMP
    - FMP[tx] Sends data using UDP/TL to a remote device
    - FMP[rx] Receives data using UDP/TL from a remote device

NOTE
**VMP end-points do not support T.38 fax but may be used to send a T.30 fax over RTP.

Allocation and configuration of these end-points is beyond the scope of this document. It is the responsibility of the user to correctly configure the appropriate end-points prior to creating a T.38 job.

Generally, a T.38 gateway requires TDM end-points for the T.30 side of a call and FMP end-points for T.38. In some cases it is possible to send a T.30 fax over RTP, in this case VMP end-points can be used in place of TDM end-points. All end-points must be allocated on the same `tSMModuleId`.

2.3.2 Sessions

A session is responsible for managing instances of a T.38 gateway, referred to as jobs. A session identifier is used to describe each session. This identifier is required each time a job is added to a session and when the session is started or stopped. The session can manage multiple jobs each of which are added by the user. A job will begin executing either as it is added to an already executing session or when the session it is being added to is started. In order to start a session, the session identifier is passed into the session worker function. Once the worker function has been invoked a session is said to have started. Each session should run in it's own thread, it is the responsibility of the user to start a thread for each session with attributes appropriate to the user's application and for that thread to call the session worker function. A session is stopped by calling `sm_t38gw_stop_session()`. This will instruct a session to stop all remaining jobs and terminate as soon as all jobs belonging to the session have completed. When the session worker function returns, the session is said to have stopped and the session identifier can be destroyed.

2.3.3 Jobs

A job encapsulates a single instance of a T.38 gateway. Jobs are managed by a session. Each job acts independently within a given session and may be configured differently to other jobs in the same session. Individual jobs perform all necessary conversions between the T.30 and T.38 end-points, the direction of the conversion is specified by the user during creation of the job. The user must provide, correctly configured, end-points in order to create a job.

A job identifier is created for each job. The identifier is used to add a job to a session or to abort a running job. Once a job has been created, no further configuration changes are permitted on any of the end-points until the job has been destroyed. Adding a job to a session hands control of that job to the library. Once control of a job has been handed to the library, the user may interrogate the current status of, or abort, a running job. After receiving notification that a job has terminated, the user is responsible for destroying the job. Destruction of a job returns control of the end-points to the user.

2.3.4 Workers

Each T.38 gateway session requires a worker thread. Users must start a thread with appropriate attributes which must then call the worker function, `sm_t38gw_worker_fn()`.

2.3.5 Call-backs

The T.38 gateway library uses a call-back function to notify the user of state changes to a T.38 gateway job. The user is required to provide a function will not block or perform any action that is permitted to block when it is executed.

NOTE
The T.38 Gateway API functions may block and should not be used in a call-back

2.4 Limitations

2.4.1 Connecting calls via the T.38 Gateway API

It is neither possible nor desirable to connect two T.30 devices using the T.38 Gateway API. It is the responsibility of the user to detect that both parties are T.30 devices and to connect them directly. The same restrictions apply to T.38 devices. Where both parties in a call are using T.38 there is no need to use theT.38 Gateway API, instead the two devices must be connected directly by the user. Directly connecting two parties in this way removes any delay imposed by the gateway and allows the two endpoints to use the best possible transfer methods that they mutually support, rather than a subset supported by the T.38 Gateway API.

3 API header files and libraries

3.1 Microsoft Windows Operating System

Required header files:

'smt38gwlib.h'

NOTE

'smt38gwlib.h' requires 't38gwtypes.h' to be locatable when compiling applications, however, users do not need to include this file.

Required libraries and DLLs (DLL library)

'smt38gwlib.lib'

'smt38gwlib.dll'

Required libraries (Static link library – available on request)

'smt38gwlibstat.lib'

Note:- When using the static library add ***T38API=""*** to application compile defines.

NOTE

The T.38 Gateway API requires Aculab's TiNG.dll

3.2 Linux Operating System.

Required header files:

'smt38gwlib.h'

NOTE

'smt38gwlib.h' requires 't38gwtypes.h' to be locatable when compiling applications, however, users do not need to include this file.

Required libraries (Shared object library)

'libsmt38gwlib.so'

Required libraries (Static link library)

'libsmt38gwlib.a'

NOTE

The T.38 Gateway API requires Aculab's libTiNGshared.so

4 Data Types

These are the data types defined by the Prosody T.38 Gateway API:

Data Type	Description
tSMT38GWSessionId	A session identifier for T.38 gateway sessions. This type is created by sm_t38gw_create_session() and destroyed by sm_t38gw_destroy_session().
tSMT38GWJobId	A job identifier for T.38 gateway jobs. This type is created by sm_t38gw_create_job() and destroyed by sm_t38gw_destroy_job().
struct tSMT38GWEndpoint	A description of the Prosody end-points to be used in a T.38 gateway job. See sm_t38gw_create_job() for its definition.
T30TDM_EP	Describes an end-point that will carry T.30 data over the TDM, used by struct tSMT38GWEndpoint.
T30VMP_EP	Describes an end-point that will carry T.30 data over RTP, used by struct tSMT38GWEndpoint.
T38FMP_EP	Describes an end-point that will carry T.38 data over UDP, used by struct tSMT38GWEndpoint.
SM_T38GW_JOB_CONTEXT_PARMS	Defines the information that is available in the user-defined call-back function, job_notify(). See sm_t38gw_create_job().

The Prosody T.38 Gateway API also uses the following data types defined in the Prosody 2 TiNG API guide.

- tSMModuleId
- tSMTDMtxId
- tSMTDMrxId
- tSMVMPtxId
- tSMVMPrxId
- tSMFMPtxId
- tSMFMPrxId

5 API call descriptions

5.1 sm_t38gw_create_session

Prototype Definition

```
int sm_t38gw_create_session(struct sm_t38gw_session_parms *session_parms)
```

Parameters

`*session_parms`

a structure of the following type:

```
typedef struct sm_t38gw_session_parms {
   tSMT38GWSessionId session;/* out */
} SM_T38GW_SESSION_PARMS;
```

Description

Creates a T.38 Gateway session. The created session is able to manage multiple T.38 jobs.

Fields

session

The newly created `tSMT38GWSessionId`.

Returns

0 if call completed successfully, otherwise a standard error such as:

`ERR_SM_DEVERR` - device error

5.2 sm_t38gw_stop_session

Prototype Definition

```
int sm_t38gw_stop_session(struct sm_t38gw_stop_session_parms
*stop_session_parms)
```

Parameters

`*stop_session_parms`

a structure of the following type:

```
typedef struct sm_t38gw_stop_session_parms {
   tSMT38GWSessionId session;/* in */
} SM_T38GW_STOP_SESSION_PARMS;
```

Description

Requests that a currently executing T.38 session stops. The session will not stop until all jobs have terminated. When a session stops the worker function, `sm_t38gw_worker_fn()`, will return.

Fields

session

A `tSMT38GWSessionId` obtained from a previous call to `sm_t38gw_create_session()`.

Returns

0 if call completed successfully, otherwise a standard error such as:

`ERR_SM_DEVERR` - device error

5.3 sm_t38gw_destroy_session

Prototype Definition

```
int sm_t38gw_destroy_session(tSMT38GWSessionId session)
```

Parameters

session

A `tSMT38GWSessionId` that has been previously created by a call to `sm_t38gw_create_session()`

Description

Destroys the T.38 gateway session and invalidates the `tSMT38GWSessionId`.

NOTE

Once the worker function has been called for a session, the session must not be destroyed until the worker function has returned.

Returns

0 if call completed successfully, otherwise a standard error such as:

`ERR_SM_DEVERR` - device error

5.4 sm_t38gw_worker_fn

Prototype Definition

```
int sm_t38gw_worker_fn(struct sm_t38gw_worker_parms *worker_parms)
```

Parameters

`*worker_parms`

a structure of the following type:

```
typedef struct sm_t38gw_worker_parms {
   tSMT38GWSessionId session;/* in */
} SM_T38GW_WORKER_PARMS;
```

Description

This worker function is intended to run in its own thread. It is the responsibility of the user to start a thread with the appropriate attributes and to subsequently invoke this function.

The worker will terminate upon a fatal error or as a result of the user requesting that a session be stopped.

Fields

session

A `tSMT38GWSessionId` obtained from a previous call to `sm_t38gw_create_session()`

Returns

0 if call completed successfully, otherwise a standard error such as:

`ERR_SM_DEVERR` - device error

5.5 sm_t38gw_create_job

Prototype Definition

```
int sm_t38gw_create_job(struct sm_t38gw_create_job_parms *create_job_parms)
```

Parameters

`*create_job_parms`

a structure of the following type:

```
typedef struct sm_t38gw_create_job_parms {
   tSMT38GWJobId job;/* out */
   struct tSMT38GWEndpoint {
        enum kSMT38GWDeviceType {
             kSMT38GWDeviceTypeT30TDM,
             kSMT38GWDeviceTypeT30VMP,
             kSMT38GWDeviceTypeT38FMP,
        } type;/* in */
        union {
             struct {
                  tSMTDMtxId tdmtx;/* in */
                  tSMTDMrxId tdmrx;/* in */
             } T30TDM_EP;/* in */
             struct {
                  tSMVMPtxId vmptx;/* in */
                  tSMVMPrxId vmprx;/* in */
             } T30VMP_EP;/* in */
             struct {
                  tSMFMPtxId fmptx;/* in */
                  tSMFMPrxId fmprx;/* in */
                  int PreCorrigendum;/* in */
             } T38FMP_EP;/* in */
        } u;/* in */
   } local_endpoint;/* in */
   struct tSMT38GWEndpoint remote_endpoint; /* in */
   tSMModuleId module; /* in */
   unsigned T38GWASN1Version; /* in */
   unsigned ECM_disable;      /* in */
   unsigned char modems;      /* in */
   int      fsk_tx_power;     /* in */
   int      spoofing_disable; /* in */
   void *user_id; /* in */
   void(*job_notify)(SM_T38GW_JOB_CONTEXT_PARMS *job_context); /* in */
} SM_T38GW_CREATE_JOB_PARMS;
```

Description

Creates a T.38 Gateway job.

In order for a T.38 gateway job to successfully communicate with T.38 and T.30 devices, the end-points must be correctly connected, prior to invoking this API call. The `local_endpoint` refers to the device initiating the call, the originating device. The `remote_endpoint` refers to the device terminating the call, the destination device. The `local_endpoint` must be set up for communicating with the originating (CNG sending) fax device , with the `remote endpoint` set with the path to the responding (answering) fax device.

When specifying the available modem capabilities, only modems that have had their firmware modules loaded should be added to the bit-mask. It is recommended that the firmware for all of the available modems be loaded in order to provide the highest level of compatibility between the T.38 gateway and T.30 fax devices.

Fields

Job

An identifier for the newly created job.

local_endpoint

The end-points for communication with the local device

type

The type of device to communicate with One of these values:

`kSMT38GWDeviceTypeT30TDM`
A T.30 device over TDM

`kSMT38GWDeviceTypeT30VMP`
A T.30 device over RTP

`kSMT38GWDeviceTypeT38`
A T.38 device

T30TDM_EP

`tdmtx`
The TDM end-point for transmission to a T.30 device

`tdmrx`
The TDM end-point for reception from a T.30 device

T30VMP_EP

`vmptx`
The VMP end-point for transmission to a T.30 device

`vmprx`
The VMP end-point for reception from a T.30 device

T38FMP_EP

`fmptx`
The FMP end-point for transmission to a T.38 device

`fmprx`
The FMP end-point for reception from a T.38 device

`PreCorrigendum`
This field should be left blank; the library sets it as needed, based on the value of `T38GWASN1Version`.

remote_enpoint

The end-points for communication with the local device.

module

The `tSMModuleId` upon which the end-points were allocated.

T38GWASN1Version

The T.38 ASN.1 version to be used by this job, as defined in [2].

ECM_disable

Disables ECM in remote DIS when non-zero.

Default is zero (no modification of DIS message).

modems

A bit-mask specifying the modems supported by the T.30 component of the gateway. The available modems are:

```
T38GW_TDM_MODEM_V17
T38GW_TDM_MODEM_V29
T38GW_TDM_MODEM_V27
```

`fsk_tx_power`

Sets v21 transmission power, value used displayed in T.38 Gateway log at job start.

This parameter has a usable range of –13 - -31 dBm0. Power above –13 is clipped to –13, lowest usable power level is –31. A value of 0 forces default of –13.

spoofing_disable

Disable T.30 fax spoofing when non-zero. Default is spoofing enabled.

*user_id

A user-defined identifier that is returned to the user in the call-back function. This can be a pointer to a user-defined data structure or any other unique identifier to be associated with a `tSMT38GWJobId`.

(void)*job_notify(SM_T38GW_JOB_CONTEXT_PARMS *job_context)

A user-provided call-back function that is used, by the library, to notify users of a change in the status of a given job. The call-back function must not perform any actions that might cause it to block, this includes calling any T.38 gateway API or Prosody API functions.

Requires:

`*job_context`

a structure of the following type:

```
typedef struct sm_t38gw_job_context_parms {
     tSMT38GWJobId job;
     void *user_id;
} SM_T38GW_JOB_CONTEXT_PARMS;
```

Fields

Job

A `tSMT38GWJobId` obtained from a call to `sm_t38gw_create_job()`.

*user_id

The user-defined identifier that was supplied to `sm_t38gw_create_job()`.

Returns

0 if call completed successfully, otherwise a standard error such as:

`ERR_SM_DEVERR` - device error

5.6 sm_t38gw_set_job_mpl

Prototype Definition

```
int sm_t38gw_set_job_mpl(struct sm_t38gw_set_job_mpl_parms
*set_job_mpl_parms)
```

Parameters

`*set_job_mpl_parms`

a structure of the following type:

```
typedef struct sm_t38gw_set_job_mpl_parms {
   tSMT38GWJobId job;
   int max_packet_length;
} SM_T38GW_SET_JOB_MPL_PARMS;
```

Description

Optional call to configure gateway job to a set maximum T.38 packet length. If used, must be called prior to sm_t38gw_add_job.

Fields

Job

A `tSMT38GWJobId` obtained from a call to `sm_t38gw_create_job()`.

max_packet_length

The maximum T.38 packet length to be specified for the gateway job.

Returns

0 if call completed successfully, otherwise a standard error such as:

`ERR_SM_DEVERR` - device error

5.7 sm_t38gw_add_job

Prototype Definition

```
int sm_t38gw_add_job(struct sm_t38gw_add_job_parms *job_parms)
```

Parameters

`*job_parms`

a structure of the following type:

```
typedef struct sm_t38gw_add_job_parms {
   tSMT38GWJobId job;/* in */
   tSMT38GWSessionId session;/* in */
} SM_T38GW_ADD_JOB_PARMS;
```

Description

Adds `job` to the T.38 Gateway session, `session`. A T.38 gateway job will begin as soon as it is successfully added to an executing T.38 gateway session. If the worker function, `sm_t38gw_worker_fn()`, has not yet been invoked for a session, any jobs that have been added to the session will begin when the worker function is called.

Fields

Job

A `tSMT38GWJobId` obtained from a call to `sm_t38gw_create_job()`.

session

A `tSMT38GWSessionId` obtained from a call to `sm_t38gw_create_session()`. This is the session that will control `job`.

Returns

0 if call completed successfully, otherwise a standard error such as:

`ERR_SM_DEVERR` - device error

`ERR_T38GW_SESSION_FULL` – This session is full and cannot support any more jobs. A new session must be created for this job

5.8 sm_t38gw_job_status

Prototype Definition

```
int sm_t38gw_job_status(struct sm_t38gw_job_status_parms *status_parms)
```

Parameters

`*status_parms`

a structure of the following type:

```
typedef struct sm_t38gw_job_status_parms {
   tSMT38GWJobId job;/* in */
   struct sm_t38gw_job_status_report {
        enum tSMT38GWJobStatus {
             tSMT38GWJobStatusRunning,
             tSMT38GWJobStatusTerminated,
        } status; /* out */
         enum tSMT38GWJobTerminationReason {
             tSMT38GWJobTerminationDCN,
             tSMT38GWJobTerminationUser,
             tSMT38GWJobTerminationError,
        } termination_reason; /* out */
         enum tSMT38GWJobFaxOutcome {
             tSMT38GWJobFaxOutcomeTransferIncomplete,
             tSMT38GWJobFaxOutcomeTransferConfirmed,
        } fax_outcome; /* out */
         int page_count; /* out */
   } report; /* out */
  U32 info;  /* out */

} SM_T38GW_JOB_STATUS_REPORT;} SM_T38GW_JOB_STATUS_PARMS;
```

Description

Interrogates the T.38 gateway job represented by *`job`*.

Fields

Job

A `tSMT38GWJobId` obtained from a call to `sm_t38gw_create_job()`.

report

A status report containing information about the current state of:

A T.38 Gateway Job.

status

The current status of job. One of these values:

`tSMT38GWJobStatusRunning`

The T.38 Gateway job is currently running

`tSMT38GWJobStatusTerminated`

The T.38 Gateway job has finished and the user is now responsible for *`job`* and any resources associated with it.

termination_reason

This field is only valid if the status is tSMT38GWJobStatusTerminated. One of these values:

`tSMT38GWJobTerminationDCN`

The T.38 Gateway job was stopped by a disconnect signal from one of the fax devices (this in itself does not indicate success, see 'fax_outcome')

`tSMT38GWJobTerminationUser`

The T.38 Gateway job was aborted by the user

`tSMT38GWJobTerminationError`

The T.38 Gateway job terminated due to an error

fax_outcome

The outcome of the fax as perceived by the T.38 Gateway. The T.38 Gateway API attempts to give an indication of whether the remote fax device successfully received all of the pages of a fax. This field is only valid if the status is `tSMT38GWJobStatusTerminated`. One of these values:

`tSMT38GWJobFaxOutcomeTransferIncomplete`

The T.38 Gateway API could not confirm that all pages were successfully received as it did not receive the appropriate acknowledgements from the remote fax device.

`tSMT38GWJobFaxOutcomeTransferConfimed`

The T.38 Gateway API received confirmation that the last page of the fax was successfully received by the remote fax device.

page_count

The number of pages that have been confirmed by the remote fax device.

Info

Unsigned 32 bit information parameter field (5.12)

Returns

0 if call completed successfully, otherwise a standard error such as:

`ERR_SM_DEVERR` - device error

5.9 sm_t38gw_abort_job

Prototype Definition

```
int sm_t38gw_abort_job(struct sm_t38gw_abort_job_parms *abort_parms)
```

Parameters

`*abort_parms`

a structure of the following type:

```
typedef struct sm_t38gw_abort_job_parms {
   tSMT38GWJobId job;/* in */
} SM_T38GW_ABORT_JOB_PARMS;
```

Description

Attempts to abort the currently executing T.38 gateway job, *`job`*. When the job terminates due to a user abort, `termination_reason` in the <?> will be set to `tSMT38GWJobTerminationUser`.

NOTE

It is not possible to abort a job unless it has been added to a session and the session is currently executing.

Fields

Job

A `tSMT38GWJobId` obtained from a call to `sm_t38gw_create_job()`.

Returns

0 if call completed successfully, otherwise a standard error such as:

`ERR_SM_DEVERR` - device error

5.10 sm_t38gw_destroy_job

Prototype Definition

```
int sm_t38gw_destroy_job(tSMT38GWJobId job)
```

Parameters

Job

A `tSMT38GWJobId` obtained from a call to `sm_t38gw_create_job()`.

Description

Destroys the T.38 gateway job specified by *`job`* and invalidates the `tSMT38GWJobId`.

NOTE

Once a job has been added to a session, the job must not be destroyed until the status ‘`tSMT38GWJobStatusTerminated`’ has been reported.

Returns

0 if call completed successfully, otherwise a standard error such as:

`ERR_SM_DEVERR` - device error

5.11 sm_t38gw_destroy_job_info

Prototype Definition

```
int sm_t38gw_destroy_job_info(tSMT38GWJobId job, U32 *info)
```

Parameters

Job

A `tSMT38GWJobId` that has been previously created by a call to sm_t38gw_create_job().

Description

Destroys the T.38 gateway job specified by *`job`* and invalidates the `tSMT38GWJobId`.

NOTE

Once a job has been added to a session, the job must not be destroyed until the status ‘`tSMT38GWJobStatusTerminated`’ has been reported.

Returns

0 if call completed successfully, otherwise a standard error such as:

`ERR_SM_DEVERR` - device error

Unsigned 32 bit information parameter field (5.12) written to ‘info’.

5.12 T.38 Gateway job information field.

A 32 bit information field is returned by two of the T.38 Gateway API calls.

sm_t38gw_job_status

sm_t38gw_destroy_job_info

sm_t38gw_job_status returns the information field at the present stage of the fax transfer.

sm_t38gw_destroy_job_info returns the full information field of the completed job.

Bit	Mask	Description
0	T38GW_STATUS_CNG	At least one CNG received.
1	T38GW_STATUS_CED	At least one CED received.
2	T38GW_STATUS_DIS	At least one DIS received.
3	T38GW_STATUS_DCS	At least one DCS received.
4	T38GW_STATUS_CFR	At least one CFR received.
5	T38GW_STATUS_FTT	At least one FTT received.
6	T38GW_STATUS_MPS	At least one MPS received.
7	T38GW_STATUS_EOP	At least one EOP received.
8	T38GW_STATUS_MCF	At least one MCF received.
9	T38GW_STATUS_RTP	At least one RTP received.
10	T38GW_STATUS_RTN	At least one RTN received.
11	T38GW_STATUS_DCN	DCN received.
12	T38GW_STATUS_PPR	At least one PPR received.
13	T38GW_STATUS_CTR	At least one CTR received.
14	T38GW_STATUS_CTC	At least one CTC received.
15	T38GW_STATUS_PPS_NULL	At least one PPS_NULL received.
16	T38GW_STATUS_PPS_MPS	At least one PPS_MPS received.
17	T38GW_STATUS_PPS_EOP	At least one PPS_EOP received.
18		
19	T38GW_STATUS_SPOOFED	T.30 endpoint spoofed.
20	Reserved.	Aculab use only.
21	Reserved.	Aculab use only.
22	Reserved.	Aculab use only.
23	Reserved.	Aculab use only.
24		
25	T38GW_STATUS_JOB_KILLED	T.38 job terminated, see log.
26	T38GW_STATUS_MEMORY_ERROR	*Contact Aculab.*
27	T38GW_STATUS_RESOURCE_ERR	Memory allocation error.
28		
29		
30		
31		

6 T38.Gateway trace.

To enable tracing to stdout set the variable 'T38GWtrace' (which is exported) to one of the following trace levels:

T38GW_TRACELVL_LOW

T38GW_TRACELVL_MED

T38GW_TRACELVL_HIGH

A full list of the individual trace masks can be found in *t38gwtypes.h*

Trace can be modified or redirected by defining a new trace function and re-assigning the trace function pointer.

The signature for the trace function is:

*int fn(const char *fmt, va_list ap)*

and the function pointer is:

T38GW_showtrace

e.g.

To trivially output a timestamp and then the standard trace:

```
int timedtrace(const char *fmt, va_list ap)
{
    DWORD tm = GetTickCount();
    printf("%u: ", tm);
    vprintf(fmt, ap);
    return 0;
}
...
```

Then inside the main function (preferably before calls to any

T.38 gateway API functions):

```
{
    ...
    T38GW_showtrace = timedtrace;
    ...
}
```

NOTE

Temporary buffers allocated to hold data from the logging function require to be at least 6180 bytes long to allow for the longest possible data that can be outputted.

6.1 T.38 Gateway Input and Output data logging.

The T.38 gateway logging defaults to displaying the number of bytes read and sent on each data path as :-

(job number) READ (n).

(job number) SEND (n).

Full display of all data received and sent can be enabled by 'oring'

T38GW_TRACELVL_INDATA

T38GW_TRACELVL_OUTDATA

with the trace level used.

For example

T38GWtrace = (T38GW_TRACELVL_MED | T38GW_TRACELVL_INDATA);

displays all data received on both paths by the gateway.

Full data logging increases log file size and results in greater system load.

7 Appendix A : References

[1] Prosody Version 2 API Guide.

The above document can be found online at http://www.aculab.com/ under technical documents.

[2] ITU-T Recommendation T.38 – Procedures for real-time Group 3 facsimile communication over IP networks

[3] ITU-T Recommendation T.30 – Procedures for document facsimile transmission in the general switched telephone network

[4] T.38 Gateway introduction.

[5] T.38 Gateway logging.

8 Appendix B: Firmware Requirements

For basic operation the T.38 gateway library requires the following firmware modules:

General:

datafeed

T.30 Fax

hdlctx

hdclrx

fsktx

fskrx

fskpll

v27tx

v27rx

inchan

outchan

td

tonegen

synctx

syncrx

prefsuf

T.38 Fax

fmptx

fmprx

ifprx

ifprx

It is normally desirable to include these additional modems to improve performance

T.30 Fax:

v29rx

v29tx

v17tx

v17rx

If a user requires the ability to handle T.30 fax over RTP then these additional modules are also required:

T.30 Fax over RTP:

vmptx

vmprx

vtdet