STS04A API Specification

Document Number: PR-D2-0769

Revision: 53

Issue Date: 2011/01

Document Information	
Title	STS04A API Specification
Document Number	PR-D2-0769
Document Version	53
Client	Prism Payment Technologies
Synopsis	This document defines the STS API as implemented on the TSM410.
Author(s)	GeoffreyD, GiovanniG, TrevorD, ChrisA

STS04A API Specification

Table of Contents

List of Tables

1. Scope

This document defines the API for the products listed in Table 1-1 below.

Applicable Hardware Products		
Hardware Product Name	**Product Number**	**Version(s)**
TSM410		All
Applicable Software / Firmware Products		
Software / Firmware Product Name	**Product Number**	**Version(s)**
TSM410 Firmware STS04A		Various

Table 1-1 Applicable Products for API

It is to be regarded as the prime source of information for all interface specifications of the product. In the event that the contents of this document contradict the contents of another document in terms of specifying the product, this document shall take precedence, unless the other document is the Product Specification document for this product, in which case it shall take precedence. This document is one of four baseline documents defining the product.

2. API Definition

2.1. Message Formats

All messages interchanged between the host computer and the security module, are terminated by appending a carriage return (CR) character. The message termination character is preceded by a checksum of four characters. The checksum algorithm used is the cyclic redundancy code (CRC) algorithm with a sixteenth order polynomial (x16 + x15 + x2 + 1). Prior to transmitting a message, the checksum is calculated for the message. The checksum yields a 16-bit hexadecimal value (i.e. 4 hexadecimal digits). The hexadecimal digits are converted into 4 ASCII characters (0 - F) representing the hexadecimal digits. The converted checksum is appended to the message. Finally the message termination character is appended to the message prior to transmission.

User command / response data	Checksum	Terminator
Variable length	4 characters	1 character

Table 2-1 Message Formats

The message (command or response) portion comprises a five-byte header, and a variable length data section. The header indicates the message type and destination / source device.

2.1.1. Data Representation

Data is transferred between the host and the security module as a stream of ASCII characters, in the range 0x20 (space) to 0xFF.

The representation (rep) field of the various commands and responses should be interpreted as follows:

Rep	Characters allowed
A	Alphabetic, 'A' - 'Z', 'a' - 'z'
N	Numeric, '0' - '9'
AH	Hex digit, '0' - '9', 'A' - 'F'
S	ASCII symbol, e.g. '?', '!'

Table 2-2 Data Representation

In the case of numbers, the most significant digit is always transferred first.

2.2. Points Of Note

2.2.1. Check Digits Test Pattern

The check digits test pattern will always be binary zeros.

2.3. Diagnostic Commands

2.3.1. Overview

The diagnostic commands are used to perform administrative, diagnostics and user support functions.

2.3.2. SM?ID - Get Identification

Function

The device identification number, the firmware version number and six of the DAK check digits are returned. The remainder of the check digits are set to zero.

Request Format

Description	Rep	Range	Length
Destination device	A	SM	2
Request indicator	S	?	1
Command code	A	ID	2

Response Format

Description	Rep	Range	Length
Source device	A	SM	2
Response indicator	S	!	1
Command code	A	ID	2
Response code	N		2
Device identification number	N	0-9	8
Firmware version number	ANS		8
DAK check digits	AH		16

Description

If the internal DAK register is empty, the DAK check digits are set to '-' characters.

The device identification number is returned in device identification number. The firmware version number is returned in firmware version number. The check digits test pattern is encrypted with the internally stored device authentication key and the resultant key check digits are returned in DAK check digits.

2.3.3. GL?RS - Reset Device

Function

Resets the security module. All current activities are halted and the device is reset to a state in which it is ready to receive and process commands.

This command is included for backwards compatibility – you should not need to issue a GL?RS to a TSM4xx/5xx device.

Request Format

Description	Rep	Range	Length
Destination device	A	GL	2
Request indicator	S	?	1
Command code	A	RS	2

Response Format

Description	Rep	Range	Length
Source device	A	GL	2
Response indicator	S	!	1
Command code	A	RS	2
Response code	N		2

Description

Upon receiving this command, the current activities are halted and the security module is reset to a state in which it is ready to receive and process commands.

NOTE: Allow for a command timeout of 1 minute to allow the TSM410 time to start up.

2.3.4. GL?EC - Echo Data

Function

Echoes data sent to the security module. Used to test communications.

Request Format

Description	Rep	Range	Length
Destination device	A	GL	2
Request indicator	S	?	1
Command code	A	EC	2
Delay	N	00-99	2
No of echo characters	N	000-512	3
Echo data	ANS		Max 512

Response Format

Description	Rep	Range	Length
Source device	A	GL	2
Response indicator	S	!	1
Command code	A	EC	2
Response code	N		2
No of echo characters	N	000-512	3
Echo data	ANS		Max 512

Description

This function returns echo data sent by the host in the request message in echo data in the response message. The number of characters to be echoed is specified by no of echo characters.

Delay contains a value in seconds. This value indicates the number of seconds the response message is delayed before it is returned to the host. If Delay contains 00, no delay is observed and the response message is returned immediately. This feature is useful to provide the user with a crude timer mechanism.

2.3.5. SM?DQ - Query the date and time of the real time clock

Function

Return the modules date/time.

Request Format

Description	Rep	Range	Length
Destination device	A	SM	2
Request indicator	S	?	1
Command code	A	DQ	2

Response Format

Description	Rep	Range	Length
Source device	A	SM	2
Response indicator	S	!	1
Command code	A	DQ	2
Response code	N		2
Date/Time string (YYYYMMDDHHMMSS)	AN		14

2.4. Vending Commands (PCI interface)

2.4.1. Overview

The security module stores default, unique and common vending keys for a number of supply groups in the standard DES key registers; these may be loaded under a key exchange key.

The key types for the vending keys are defined in the table below.

Key type / function	Key type
Default vending key	E
Unique vending key	M
Common vending key	N
Vending Authentication DES Key	P
Sub-Vending Authentication DES Key	Q

Tokens are transferred to and from the module in binary (8.5 bytes, transmitted as 17 hex digits) and text formats. For STS, the text format is 20 decimal digits; for the Conlog proprietary meters this format is 18 hex digits, left justified and padded with two zeros to fill the 20-character field.

Some of the commands below use transfer amounts; these are specified as 16 bit values, coded according to the STS floating point representation. Similarly, the time and date is transferred in the STS format, and converted to proprietary formats as required.

Some of the commands below specify an algorithm type; if this field is not supplied, the STS1 algorithm is assumed. Currently, only the STS algorithm is supported in the STS API

Code	Algorithm
07	STS1

Some of the commands below also specify a token technology; if this field is not supplied, the STS1 magnetic token is assumed. Currently, the token technologies supported are given in the table below.

Code	Algorithm
01	STS1 magnetic token
02	STS1 numeric token

2.4.1.1. STS1 Credit Token

The credit tokens (in clear hexadecimal format) are defined as follows:

Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Value	S	R	I	I	I	I	I	I	A	A	A	A	P	P	P	P

S Sub-Class - set according to the credit function field value:

016	electricity credit
116	water credit
216	gas credit

316	connection time credit
416	currency credit
516	reserved for future STS use
616	reserved for future STS use
716	reserved for future STS use
816	reserved for future STS use
916	reserved for future STS use
A16	reserved for future STS use
B16	reserved for future STS use
C16	reserved for future STS use
D16	reserved for future STS use
E16	reserved for future STS use
F16	reserved for future STS use

R Random Pattern - range 016 - F16

This 4 bit random number is internally generated from the random number generator.

I Token Id digit - range 016 - F16

These 6 hexadecimal digits correspond to token id field value passed in the command as 6 bytes of ASCII hex.

A Transfer Amount digit - range 016 - F16

These 4 hexadecimal digits correspond to the transfer amount field value passed in the command as 4 bytes of ASCII hex.

P CRC digit - range 016 - F16

This credit token is STS encrypted with the dispenser STS key. The encrypted credit token - binary field of the response message is returned as a 17 byte ASCII hex value - the first 2 bits of the 66 bit value form the 2 least significant bits of the first hex digit, with the 2 most significant bits set to zero. The remaining 64 bits or 8 bytes are returned in the following 16 hex digits. The encrypted credit token - text field is returned after conversion from the 66 bit binary value.

2.4.2. XM?TC – Encrypt Credit Function Token

Function

Formats and encrypts a credit transfer function under the dispenser key.

Request Format

Description	Rep	Range	Length
Destination device	A	XM	2
Request indicator	S	?	1
Command code	A	TC	2
Dispenser PAN	AN		19
Vending supply key register number	N	001-999	3
Supply group code	N		6
Tariff index	N		2
Key revision number	N	1-9	1
Key expiry number	AH		2
Credit function	N	0-15	2
- *Electricity credit*		0	
- *Water credit*		1	
- *Gas credit*		2	
- *Connection time credit*		3	
- *Currency credit*		4	
- *Reserved for future STS use*		5-15	
Token id	AH		6
Transfer amount	AH		4
Algorithm type (optional, default = STS1)	N	04,07	0 / 2
Token technology (optional, default = 01)	N	01,02,04	0 / 2

Response Format

Description	Rep	Range	Length
Source device	A	XM	2
Response indicator	S	!	1
Command code	A	TC	2
Response code	N		2
Encrypted credit token - binary	AH		17
Encrypted credit token - text	N		20

Description

Formats and encrypts a credit transfer function under the dispenser key. Support is provided for the following functions:

- Electricity credit
- Water credit
- Gas credit
- Connection time credit
- Currency credit

Notes

- The dispenser PAN field contains the primary account number of the dispenser. Like any PAN it is all numeric decimal, but could be of variable length. It is passed left justified and is space padded to the right. A value of all spaces is valid, and represents a null PAN.
- The key expiry number is an 8 bit binary value in the STS, and hence is passed as 2 bytes of ASCII hex.
- The token id is a 24 bit binary value in the STS, and hence is passed as 6 bytes of ASCII hex.
- The transfer amount is a 16 bit binary value (STS floating point format), and hence is passed as 4 bytes of ASCII hex.
- Any format errors should be treated in the standard way by returning a response code of format error.
- Conlog proprietary meters support only electricity credits. The supply group code, tariff index, key revision number, key expiry number fields are ignored, while the credit function field must be zero (otherwise format error).
- The algorithm type and token technology type are optional (either both or neither must be specified).

Implementation

- If the register indicated by vending key register number is empty, a response code of key number error is returned.
- The key specified in this register must be a vending supply key (type M or N). If not, a response code of key type error is returned - specifically; please note that a vending default key (type E) is not valid.
- For STS tokens (algorithm type 07), if the token technology is 02 (STS numeric token), and the vending key is N (common key type), a key type error is returned.
- If the key specified in this register was stored with parity, perform a parity check to ensure the validity of the key. If this check fails, a response code of key integrity error is returned.
- Conlog proprietary meters can transfer at most 6553.5 kWh; if the transfer amount exceeds this, a format error is returned. Also, if the token technology is not 04, a format error is returned.
- If the transfer amount specified in the request is greater than the credit balance amount stored in the security module, a response code of insufficient credit balance is returned.
- Deduct the transfer amount specified from the credit balance amount store din the security module.
- Generate a dispenser key from the specified vending supply key and the data fields passed.
- Note that if the dispenser PAN field is all spaces, the PAN is considered as a null PAN, and for purposes of formatting the PAN block is treated as all zero.
- Format the credit token from the data fields.
- Encrypt the credit token with the dispenser key generated.
- Format the encrypted credit token binary and text fields.
- If the encrypt credit function command completed successfully, the successful response code is returned together with the encrypted credit tokens.

2.4.3. SM?TC - Encrypt Credit Function Token (Legacy)

Function

As per the XM?TC command but with support for registers "01" to "99" only.

Request Format

Description	Rep	Range	Length
Destination device	A	SM	2
Request indicator	S	?	1
Command code	A	TC	2
Dispenser PAN	AN		19
Vending supply key register number	N	01-99	2
Supply group code	N		6
Tariff index	N		2
Key revision number	N	1-9	1
Key expiry number	AH		2
Credit function	N	0-15	2
- *Electricity credit*		0	
- *Water credit*		1	
- *Gas credit*		2	
- *Connection time credit*		3	
- *Currency credit*		4	
- *Reserved for future STS use*		5-15	
Token id	AH		6
Transfer amount	AH		4
Algorithm type (optional, default = STS1)	N	04,07	0 / 2
Token technology (optional, default = 01)	N	01,02,04	0 / 2

Response Format

Description	Rep	Range	Length
Source device	A	SM	2
Response indicator	S	!	1
Command code	A	TC	2
Response code	N		2
Encrypted credit token - binary	AH		17
Encrypted credit token - text	N		20

2.4.4. XM?TV - Verify Encrypted Token

Function

Decrypts and verifies the CRC of a STS dispenser specific management function or credit transfer function token.

Request Format

Description	Rep	Range	Length
Destination device	A	XM	2
Request indicator	S	?	1
Command code	A	TV	2
Dispenser PAN	AN		19
Vending key register number	N	001-999	3
Supply group code	N		6
Tariff index	N		2
Key revision number	N	1-9	1
Key expiry number	AH		2
Encrypted token - text	N		20
Algorithm type (optional, default = STS1)	N	04,07	0 / 2
Token technology (optional, default = 01)	N	01,02,04	0 / 2

Response Format

Description	Rep	Range	Length
Source device	A	XM	2
Response indicator	S	!	1
Command code	A	TV	2
Response code	N		2
Token class	N		1
- Credit transfer token		0	
- Dispenser specific management token		2	
Token sub-class	N	0-15	2
Token id	AH		6
Transfer amount	AH		4

Description

Decrypts and verifies an STS dispenser specific management function or credit transfer function token. Support is provided for all credit function tokens, and all STS dispenser specific management function tokens.

The token class, sub-class (function), id and transfer amount are returned if the token is valid, except for the set 1st and 2nd section dispenser key tokens - where the token id and transfer amount fields do not apply, and are set to zero.

Notes

- The dispenser PAN field contains the primary account number of the dispenser. Like any PAN it is all numeric decimal, but could be of variable length. It is passed left justified and is space padded to the right. A value of all spaces is valid, and represents a null PAN (see later).
- The key expiry number is an 8 bit binary value in the STS, and hence is passed as 2 bytes of ASCII hex.

- Any format errors should be treated in the standard way by returning a response code of format error.
- The transfer amount is a 16 bit binary value (STS floating point format).
- The algorithm type and token technology type are optional (either both or neither must be specified).

Implementation

- If the register indicated by vending key register number is empty, a response code of key number error is returned.
- The key specified in this register must be either a vending default key (type E) or a vending supply key (type M). If not, a response code of key type error is returned.
- If the key specified in this register was stored with parity, perform a parity check to ensure the validity of the key. If this check fails, a response code of key integrity error is returned.
- Generate a dispenser key from the specified vending supply key and the data fields passed.
- Note that if the dispenser PAN field is all spaces, the PAN is considered as a null PAN, and for purposes of formatting the PAN block is treated as all zero.
- Convert the text format token to binary.
- For STS tokens
- Extract the token class 66-bit value, and STA decrypt the 64 bit token with the dispenser STS key generated.
- Calculate the CRC over the decrypted token and compare with the token CRC value - if not equal, return a response code of invalid STS token.
- If the register indicated by vending key register number contains a vending default key (VDDK - type E), and the token class is credit transfer (102), return a response code of invalid STS token, else return a response code of successful.
- If the token is valid, and the token class is a dispenser specific management token (2), and the token sub-class is set 1st or 2nd section dispenser key (3 or 4), return the token id and transfer amount as zeroes, otherwise return the token id and transfer amount with the values in the decrypted token.
- For Conlog proprietary tokens
- Decrypt the token with the dispenser key generated.
- Calculate the CRC over the decrypted token and compare with the token CRC value - if not equal, return a response code of invalid STS token.
- If the token is valid, return the (converted) token id and transfer amount with the values in the decrypted token. Set the token class and sub-class to zero (electricity credit).

2.4.5. SM?TV - Verify Encrypted Token (Legacy)

Function

As per the XM?TV command but with support for registers “01” to “99” only.

Request Format

Description	Rep	Range	Length
Destination device	A	SM	2
Request indicator	S	?	1
Command code	A	TV	2
Dispenser PAN	AN		19
Vending key register number	N	01-99	2
Supply group code	N		6
Tariff index	N		2
Key revision number	N	1-9	1
Key expiry number	AH		2
Encrypted token - text	N		20
Algorithm type (optional, default = STS1)	N	04,07	0 / 2
Token technology (optional, default = 01)	N	01,02,04	0 / 2

Response Format

Description	Rep	Range	Length
Source device	A	SM	2
Response indicator	S	!	1
Command code	A	TV	2
Response code	N		2
Token class	N		1
- Credit transfer token		0	
- Dispenser specific management token		2	
Token sub-class	N	0-15	2
Token id	AH		6
Transfer amount	AH		4

2.4.6. XM?TM - STS Encrypt Management Function Token

Function

Formats and encrypts a dispenser specific management function under the dispenser STS key.

Request Format

Description	Rep	Range	Length
Destination device	A	XM	2
Request indicator	S	?	1
Command code	A	TM	2
Dispenser PAN	AN		19
Vending key register number	N	001-999	3
Supply group code	N		6
Tariff index	N		2
Key revision number	N	1-9	1
Key expiry number	AH		2
Management function	N	0-2, 5-15	2
- Set maximum power load		0	
- Clear credit		1	
- Set tariff rate		2	
- Clear tamper condition		5	
- Set phase power unbalance limit		6	
- Set water factor		7	
- Reserved for future STS use		8-10	
- Reserved for proprietary use		11-15	
Token id	AH		6
Transfer amount	AH		4

Response Format

Description	Rep	Range	Length
Source device	A	XM	2
Response indicator	S	!	1
Command code	A	TM	2
Response code	N		2
Encrypted management token - binary	AH		17
Encrypted management token - text	N		20

Description

Formats and encrypts a dispenser specific management function under the dispenser STS key. This command excludes support for the set 1st section dispenser key and set 2nd section dispenser key management functions - these functions are supported by the STS change key command. Support is provided for the following functions:

- Set maximum power load
- Clear credit
- Set tariff rate
- Clear tamper condition
- Set phase power unbalance limit
- Set water factor

Notes

- The dispenser PAN field contains the primary account number of the dispenser. Like any PAN it is all numeric decimal, but could be of variable length. It is passed left justified and is space padded to the right. A value of all spaces is valid, and represents a null PAN.
- The key expiry number is an 8 bit binary value in the STS, and hence is passed as 2 bytes of ASCII hex.
- The management function can be any value in the range 0 - 15, except for 3 and 4 - these latter values represent a format error.
- The token id is a 24 bit binary value in the STS, and hence is passed as 6 bytes of ASCII hex.
- The transfer amount is a 16 bit binary value in the STS, and hence is passed as 4 bytes of ASCII hex.
- Any format errors should be treated in the standard way by returning a response code of format error.

Implementation

- If the register indicated by vending key register number is empty, a response code of key number error is returned.
- The key specified in this register must be either a vending default key (type E) or a vending supply key (type M). If not, a response code of key type error is returned.
- If the key specified in this register was stored with parity, perform a parity check to ensure the validity of the key. If this check fails, a response code of key integrity error is returned.
- Generate a dispenser STS key from the specified vending key and the data fields passed.
- Note that if the dispenser PAN field is all spaces, the PAN is considered as a null PAN, and for purposes of formatting the PAN block is treated as all zero.
- Note that the STS key type is determined by the key type of the parent vending key - for a vending default key (VDDK - type E), the corresponding dispenser STS key (DDTK) type is 1, and for a vending supply key (VSDK - type M), the corresponding dispenser STS key (DSTK) type is 2.
- Format the management token from the data fields.
- STA encrypt the management token with the dispenser STS key generated.
- Format the encrypted management token binary and text fields.
- If the STS encrypt management function command completed successfully, the successful response code is returned together with the encrypted management tokens.

2.4.7. SM?TM - STS Encrypt Management Function Token (Legacy)

Function

As per the XM?TM command but with support for registers “01” to “99” only.

Request Format

Description	Rep	Range	Length
Destination device	A	SM	2
Request indicator	S	?	1
Command code	A	TM	2
Dispenser PAN	AN		19
Vending key register number	N	01-99	2
Supply group code	N		6
Tariff index	N		2
Key revision number	N	1-9	1
Key expiry number	AH		2
Management function	N	0-2, 5-15	2
- Set maximum power load		0	
- Clear credit		1	
- Set tariff rate		2	
- Clear tamper condition		5	
- Set phase power unbalance limit		6	
- Set water factor		7	
- Reserved for future STS use		8-10	
- Reserved for proprietary use		11-15	
Token id	AH		6
Transfer amount	AH		4

Response Format

Description	Rep	Range	Length
Source device	A	SM	2
Response indicator	S	!	1
Command code	A	TM	2
Response code	N		2
Encrypted management token - binary	AH		17
Encrypted management token - text	N		20

2.4.8. XM?TK - STS Change Key

Function

Encrypts a new STS key under the current STS key.

Request Format

Description	Rep	Range	Length
Destination device	A	XM	2
Request indicator	S	?	1
Command code	A	TK	2
Dispenser PAN	AN		19
Vending key register number - current	N	001-999	3
Vending key register number - new	N	001-999	3
Supply group code - current	N		6
Supply group code - new	N		6
Tariff index - current	N		2
Tariff index – new	N		2
Key revision number - current	N	1-9	1
Key revision number - new	N	1-9	1
Key expiry number - current	AH		2
Key expiry number - new	AH		2

Response Format

Description	Rep	Range	Length
Source device	A	XM	2
Response indicator	S	!	1
Command code	A	TK	2
Response code	N		2
Set 1st section dispenser token - binary	AH		17
Set 2nd section dispenser token -binary	AH		17
Set 1st section dispenser token - text	N		20
Set 2nd section dispenser token - text	N		20

Description

The command is structured to support a number of potential CVS requirements:

- To change the STS key for a given ED from one revision of the parent vending supply key or vending default key to another.
- To change the STS key for a given ED from a vending default key to a vending supply key, or from a vending supply key to a vending default key.
- To change the tariff index of an ED
- To change the supply group code of an ED - i.e. to support the combination of the EDs for two or more supply groups into a single supply group, or the split of EDs from a single supply group into two or more supply groups.
- To allow group coded ED key management.

Notes

- The dispenser PAN field contains the primary account number of the dispenser. Like any PAN it is all numeric decimal, but could be of variable length. It is passed left justified and is space padded to the right. A value of all spaces is valid, and represents a null PAN (see later).
- The key expiry number is an 8 bit binary value in the STS, and hence is passed as 2 bytes of ASCII hex.
- Any format errors should be treated in the standard way by returning a response code of format error.

Implementation

- If the registers indicated by vending key register number - current or vending key register number - new are empty, a response code of key number error is returned.
- The keys specified in each of these registers must be either vending default keys (type E) or vending supply keys (type M). If not, a response code of key type error is returned.
- Vending key register number - current and vending key register number - new must indicate registers in the same domain. If not, a response code of key number error is returned.
- If the keys specified in these registers were stored with parity, perform a parity check to ensure the validity of the key. If this check fails, a response code of key integrity error is returned.
- Generate a dispenser STS key - current and a dispenser STS key - new corresponding to the current data fields passed and the new data fields passed. Additional notes are provided in sections 7.4 and 7.5.
- Note that if the dispenser-PAN field is all spaces, the PAN is considered as a null PAN, and for purposes of formatting the PAN block is treated as all zero.
- Note that the STS key type is determined by the key type of the parent vending key - for a vending default key (VDDK - type E), the corresponding dispenser STS key (DDTK) type is 1, and for a vending supply key (VSDK - type M), the corresponding dispenser STS key (DSTK) type is 2.
- Format the set 1st section dispenser key and set 2nd section dispenser key tokens from the new data fields passed and the dispenser STS key.
- STA encrypt each set dispenser key token with the dispenser STS key - current generated.
- Format the set 1st section dispenser key and set 2nd section dispenser key binary and text fields.
- If the STS change key command completed successfully, the successful response code is returned together with the encrypted set dispenser key tokens.

2.4.9. SM?TK - STS Change Key (Legacy)

Function

As per the XM?TK command but with support for registers “01” to “99” only.

Request Format

Description	Rep	Range	Length
Destination device	A	SM	2
Request indicator	S	?	1
Command code	A	TK	2
Dispenser PAN	AN		19
Vending key register number - current	N	01-99	2
Vending key register number - new	N	01-99	2
Supply group code - current	N		6
Supply group code - new	N		6
Tariff index - current	N		2
Tariff index – new	N		2
Key revision number - current	N	1-9	1
Key revision number - new	N	1-9	1
Key expiry number - current	AH		2
Key expiry number - new	AH		2

Response Format

Description	Rep	Range	Length
Source device	A	SM	2
Response indicator	S	!	1
Command code	A	TK	2
Response code	N		2
Set 1st section dispenser token - binary	AH		17
Set 2nd section dispenser token -binary	AH		17
Set 1st section dispenser token - text	N		20
Set 2nd section dispenser token - text	N		20

2.5. Vending Authorisation Commands (PCI interface)

2.5.1. Overview

The TSM4xx/5xx features security enhancements to limit the number of tokens that may be vended. The Vending Authorisation Commands manage this feature.

2.5.2. XM?IC – Increment Transaction Counter

Function

Increment the number of remaining transactions inside the module.

Request Format

Description	Rep	Range	Length
Destination device	A	XM	2
Request indicator	S	?	1
Command code	A	IC	2
PKC1 Public Key Certificate	AH		512
INTX Instruction	AH		256-416

Response Format

Description	Rep	Range	Length
Source device	A	XM	2
Response indicator	S	!	1
Command code	A	IC	2
Response code	N		2
Vending enabled indicator character	A		1
New effective TX counter (transactions remaining)	N		10
Module serial number	N	0-9	8
Nonce	AH		8

Description:

2.5.3. SM?IC – Increment Transaction Counter (Legacy)

Function

As per XM?IC but only has support for a 6 digit transaction counter.

Request Format

Description	Rep	Range	Length
Destination device	A	SM	2
Request indicator	S	?	1
Command code	A	IC	2
PKC1 Public Key Certificate	AH		512
INTX Instruction	AH		256-416

Response Format

Description	Rep	Range	Length
Source device	A	SM	2
Response indicator	S	!	1
Command code	A	IC	2
Response code	N		2
Vending enabled indicator character	A		1
New effective TX counter (transactions remaining)	N		6
Module serial number	N	0-9	8
Nonce	AH		8

2.5.4. XM?QC – Query Transaction Counter

Function

Returns the current transaction counter value.

Request Format

Description	Rep	Range	Length
Destination device	A	XM	2
Request indicator	S	?	1
Command code	A	QC	2

Response Format

Description	Rep	Range	Length
Source device	A	XM	2
Response indicator	S	!	1
Command code	A	QC	2
Response code	N		2
Vending enabled indicator character	A		1
New effective TX counter (transactions remaining)	N		10
Module serial number	N	0-9	8
Nonce	AH		8

Description:

- A nonce is used in signed instructions (see for example XM?IC) to prevent replay. The nonce value returned from this function is the last nonce seen in a signed instruction; any subsequent nonce must be strictly greater than this value.

2.5.5. SM?QC – Query Transaction Counter (Legacy)

Function

As per XM?QC but only has support for a 6 digit transaction counter.

Request Format

Description	Rep	Range	Length
Destination device	A	SM	2
Request indicator	S	?	1
Command code	A	QC	2

Response Format

Description	Rep	Range	Length
Source device	A	SM	2
Response indicator	S	!	1
Command code	A	QC	2
Response code	N		2
Vending enabled indicator character	A		1
New effective TX counter (transactions remaining)	N		6
Module serial number	N	0-9	8
Nonce	AH		8

2.6. Key Management (Serial or PCI interface)

Operational key management functions address the management of DEA-1 keys. These functions include operations such as the storage, retrieval, generation and checking of DEA-1 keys and can be executed via the operational key management commands.

2.6.1. XM?CK – Clear DEA-1 Key

Function

This command clears the indicated key register and all of its descendant key registers.

Request Format

Description	Rep	Range	Length
Destination device	A	XM	2
Request indicator	S	?	1
Command code	A	CK	2
Key register number	N	001-999	3

Response Format

Description	Rep	Range	Length
Source device	A	XM	2
Response indicator	S	!	1
Command code	A	CK	2
Response code	N		2

Description

This command clears the indicated key register, any key register containing key components of the key in key register and all of its descendant key registers.

2.6.1. SM?CK - Clear DEA-1 Key

Function

This command clears the indicated key register and all of its descendant key registers.

Request Format

Description	Rep	Range	Length
Destination device	A	SM	2
Request indicator	S	?	1
Command code	A	CK	2
Key register number	N	01-99	2

Response Format

Description	Rep	Range	Length
Source device	A	SM	2
Response indicator	S	!	1
Command code	A	CK	2
Response code	N		2

Description

This command clears the indicated key register, any key register containing key components of the key in key register and all of its descendant key registers.

2.6.1. XM?LK - Load Single-Length DEA-1 Key

Function

The load DEA-1 key function allows for a single-length DEA-1 key to be loaded into a key register. The key must be presented to the security module encrypted with the correct variant of the parent key. The variant is determined by the type of the key to be loaded. If the key is loaded successfully, the check digits of the key are returned. Please note that only six of the check digits are returned, the remainder of the digits are set to zero.

Request Format

Description	Rep	Range	Length
Destination device	A	XM	2
Request indicator	S	?	1
Command code	A	LK	2
Key register number	N	001-999	3
Key type	A	C-N	1
Key parity	A	S,C,N	1
Parent key register number[a]	N	001-999	3
Encryption method	A	S,T	1
Encrypted key	AH		16

Response Format

Description	Rep	Range	Length
Source device	A	XM	2
Response indicator	S	!	1
Command code	A	LK	2
Response code	N		2
Key check digits	AH		16

Description

In the case of a clear working key, parent key register number must be 0. The parent key register number must be present for all other load key types. The key held in parent key register number must be a double length key. The parent key and the load key must be in the same domain. If any of these conditions are not met, a response code of key number error is returned.

The key held in parent key register number must be a KEK. If this condition is not met, a response code of key type error is returned.

If the parent key was stored with parity, the security module performs a parity check on each half of the parent key to ensure the validity of the key. If this check fails, a response code of key integrity error is returned.

Each component of the loaded key is checked against the set of DEA-1 weak, semi-weak and potentially weak key values. If this check fails, a response code of key (semi / potentially) weak error is returned.

The security module determines the variant vector associated with key type.

The module triple decrypts encrypted key using the appropriate variant for the double length key held in parent key register number.

Prior to storing the key, the resultant plain text key is adjusted according to key parity. If key parity is C (check parity), the key is checked for odd parity. If any octet has an even parity, a response

[a] not validated for key type I

code of key parity error is returned. If key parity is N (no parity), the key is not adjusted. If key parity is S (set parity), the key is adjusted to odd parity.

If the register indicated by key register number is not empty prior to the command, the register and all its descendants are cleared. In the case of the register indicated by key register number being either half of a double length key, the register set holding the double length key and all its descendants are cleared. The plain text key component, parent key register number and key type are stored in the register indicated by key register number. If the stored key is not a clear working key, the register is added to the linked list of children keys of the register, indicated by parent key register number.

If the load key operation completed successfully, the key loaded successfully response code is returned and the check digits of the loaded key are returned in key check digits. Please note that only six of the check digits are returned, the remainder of the digits are set to zero.

2.6.1. SM?LK - Load Single-Length DEA-1 Key

Function

The load DEA-1 key function allows for a single-length DEA-1 key to be loaded into a key register. The key must be presented to the security module encrypted with the correct variant of the parent key. The variant is determined by the type of the key to be loaded. If the key is loaded successfully, the check digits of the key are returned. Please note that only six of the check digits are returned, the remainder of the digits are set to zero.

Request Format

Description	Rep	Range	Length
Destination device	A	SM	2
Request indicator	S	?	1
Command code	A	LK	2
Key register number	N	01-99	2
Key type	A	C-N	1
Key parity	A	S,C,N	1
Parent key register number[a]	N	01-99	2
Encryption method	A	S,T	1
Encrypted key	AH		16

Response Format

Description	Rep	Range	Length
Source device	A	SM	2
Response indicator	S	!	1
Command code	A	LK	2
Response code	N		2
Key check digits	AH		16

Description

In the case of a clear working key, parent key register number must be 0. The parent key register number must be present for all other load key types. The key held in parent key register number must be a double length key. The parent key and the load key must be in the same domain. If any of these conditions are not met, a response code of key number error is returned.

The key held in parent key register number must be a KEK. If this condition is not met, a response code of key type error is returned.

If the parent key was stored with parity, the security module performs a parity check on each half of the parent key to ensure the validity of the key. If this check fails, a response code of key integrity error is returned.

Each component of the loaded key is checked against the set of DEA-1 weak, semi-weak and potentially weak key values. If this check fails, a response code of key (semi / potentially) weak error is returned.

The security module determines the variant vector associated with key type.

The module triple decrypts encrypted key using the appropriate variant for the double length key held in parent key register number.

Prior to storing the key, the resultant plain text key is adjusted according to key parity. If key parity is C (check parity), the key is checked for odd parity. If any octet has an even parity, a response

[a] not validated for key type I

code of key parity error is returned. If key parity is N (no parity), the key is not adjusted. If key parity is S (set parity), the key is adjusted to odd parity.

If the register indicated by key register number is not empty prior to the command, the register and all its descendants are cleared. In the case of the register indicated by key register number being either half of a double length key, the register set holding the double length key and all its descendants are cleared. The plain text key component, parent key register number and key type are stored in the register indicated by key register number. If the stored key is not a clear working key, the register is added to the linked list of children keys of the register, indicated by parent key register number.

If the load key operation completed successfully, the key loaded successfully response code is returned and the check digits of the loaded key are returned in key check digits. Please note that only six of the check digits are returned, the remainder of the digits are set to zero.

2.6.1. XM?GS - Get DEA-1 Key Status

Function

The status of a DEA-1 or double length (TDES) key (in an extension register) is returned, and this command can operate with 3 digit register numbers.

Similar to SM?GS, but the “Key register number” in the request and “Parent key register number” in the response are length 3, for a range 001-999. In addition, if the key in question is a double length key, then the check digit test pattern is TDES encrypted with the key in order to form the check digits. This enforces a better approach to the security of the keys.

Request Format

Description	Rep	Range	Length
Destination device	A	XM	2
Request indicator	S	?	1
Command code	A	GS	2
Key register number	N	001-999	3

Response Format

Description	Rep	Range	Length
Source device	A	XM	2
Response indicator	S	!	1
Command code	A	GS	2
Response code	N		2
Key type	AN	C-N	1
Key parity	A	S,N	1
Parent key register number[a]	N	001-999	3
Load mode (origin)	A	A,M,R,C	1
Encryption method	A	T	1
Key check digits	AH		16

[a]undefined for key type I

2.6.1. SM?GS - Get DEA-1 Key Status

Function

The status of a DEA-1 key is returned. The status consists of the type of the key, the key register number of the parent of the key and the key check digits.

Request Format

Description	Rep	Range	Length
Destination device	A	SM	2
Request indicator	S	?	1
Command code	A	GS	2
Key register number	N	01-99	2

Response Format

Description	Rep	Range	Length
Source device	A	SM	2
Response indicator	S	!	1
Command code	A	GS	2
Response code	N		2
Key type	AN	C-N	1
Key parity	A	S,N	1
Parent key register number[a]	N	01-99	2
Load mode (origin)	A	A,M,R,C	1
Encryption method	A	T	1
Key check digits	AH		16

Description

If the register indicated by key register number is empty, a response code of key number error is returned.

The security module performs a parity check on the key to ensure the validity of the key. If this check fails, a response code of key integrity error is returned.

The check digits test pattern is encrypted with the key in the register indicated by key register number. The resultant cipher-text is returned in key check digits. Please note that only six of the check digits ate returned, the remainder of the check digits are set to zero.

The type of the key is returned in key type.

If the key is stored with parity, key parity contains S. If the key is not stored with parity, key parity contains N.

If the key was loaded manually, a value of M is returned in load mode. If the key was loaded automatically a value of A is returned in load mode. If the key was generated (random) a value of R is returned in load mode. If the key was loaded from a smart card a value of C is returned in load mode.

If the key was not loaded manually, encryption method contains the encryption method that was used in the load or generation procedure. S is returned for single encryption and T is used for triple encryption.

In the case of a clear working key or a key that was entered manually, the return value of parent key register number is undefined. For all other keys, this field is set to the parent key register number of the key indicated by key register number.

[a]undefined for key type I

2.6.2. SM?GD - Generate Device Authentication Code

Function

Generate a device authentication code given a challenge message.

Request Format

Description	Rep	Range	Length
Destination device	A	SM	2
Request indicator	S	?	1
Command code	A	GD	2
No of hexadecimal digits	N	001-512	3
Device authentication message	AH		Max 512

Response Format

Description	Rep	Range	Length
Source device	A	SM	2
Response indicator	S	!	1
Command code	A	GD	2
Response code	N		2
Device authentication code	AH		8

Description

The DAK is retrieved from the internal DAK register and used to authenticate the security module firmware.

If the internal DAK register is empty, a response code of key number error is returned.

A 32-bit device authentication code (DAC) is calculated over the offered message in device authentication message using the ANSI X9.9/X9.19 algorithm (without data stream editing), the DAK as key and an initialisation vector consisting of 64 zero bits. The resultant DAC is returned in device authentication code.

2.7. KMC Only (Serial Interface Only)

KMC only functions address the DEA-1 operations, as well as the generation of the initial DEA-1 root key. These functions can be executed via the KMC only commands.

The security of the STS API is greatly improved in comparison to previous versions of pre-paid firmware. However, due to constraints of backwards compatibility, the STS API remains predisposed to being insecure as, originally, keys were loaded across the command interface in the clear. In addition to this, it was possible to query the check digits of each half of a double length key as though they were single length keys. While the threat of this attack was limited due to only 3 bytes worth of check digits being returned, the strength of the TDES key were reduced thereby possibly making a brute force attacks on the full-length keys feasible. In order to minimise these vulnerabilities sensitive commands are now only available over the serial interface of the TSM410. This is the same serial interface that is usually used to load key components via a Key Component Entry Device (KCED). More specifically, the initial key management commands SM?IK, SM?LK and SM?GS will only be permitted over the serial interface. It should also be noted that since this interface is therefore dedicated to key loading, no vending commands are permitted over the serial interface.

2.7.1. SM?CA - Clear All DEA-1 Keys

Function

This command clears all the key registers.

Request Format

Description	Rep	Range	Length
Destination device	A	SM	2
Request indicator	S	?	1
Command code	A	CA	2

Response Format

Description	Rep	Range	Length
Source device	A	SM	2
Response indicator	S	!	1
Command code	A	CA	2
Response code	N		2

Description

This command clears all the key registers.

2.7.2. SM?IK – Initialise Single Length DEA-1 Key

Function

Initialises a single-length DEA-1 key register.

Request Format

Description	Rep	Range	Length
Destination device	A	SM	2
Request indicator	S	?	1
Command code	A	IK	2
Key register number	N	01-99	2
Key type	AN	C,I	1
Key parity	A	S,C,N	1
Key component	AH		16

Response Format

Description	Rep	Range	Length
Source device	A	SM	2
Response indicator	S	!	1
Command code	A	IK	2
Response code	N		2
Key check digits	AH		16

Description

If the register indicated by key register number is not empty prior to the command, the register and all its descendants are cleared. The plain text key component and the key type are stored in the register indicated by key register number.

The key is checked against the set of DEA-1 weak, semi-weak and potentially weak key values. If this check fails, a response code of key (semi / potentially) weak error is returned.

Prior to storing the key, the resultant plain text key is adjusted according to key parity. If key parity is C (check parity), the key is checked for odd parity. If any octet has an even parity, a response code of key parity error is returned. If key parity is N (no parity), the key is not adjusted. If key parity is S (set parity), the key is adjusted to odd parity.

No parent register is associated with the key. A key entry method of manual loading is associated with the key.

The check digits test pattern is encrypted with the key in the register indicated by key register number. The resultant cipher-text is returned in key check digits. Note that only six of these check digits are returned, the following ten digits are set to zero.

2.7.1. SM?AK – Add Double Length DEA-1 Key Components

Function

This function XORs (Exclusive OR) a single-length DEA-1 key component with a single length key residing in an existing key register. The original key component should have been initialised using the SM?IK command.

Request Format

Description	Rep	Range	Length
Destination device	A	SM	2
Request indicator	S	?	1
Command code	A	AK	2
Key register number	N	1-N	2
Key component	AH		16

Response Format

Description	Rep	Range	Length
Source device	A	SM	2
Response indicator	S	!	1
Command code	A	AK	2
Response code	N		2
Key component check digits	AH		16
Key check digits	AH		16

Description

This command performs an exclusive-OR operation on the key in key register number using the supplied component. The key in key register number (and the key extension for the subsequent register number, if the key is a double length key) should have been loaded via an Initialise Key command (SM?IK). The table below shows how 3 key components with their 2 component parts are used to create a resultant key.

SM?IK	Key - Component 1
SM?IK	Key Extension – Component 1
SM?AK	**XORed**
	Key - Component 2
	Resultant Key
SM?AK	**XORed**
	Key Extension - Component 2
	Resultant Key Extension
SM?AK	**XORed**
	Key – Component 3
	Resultant Key
SM?AK	**XORed**
	Key Extension - Component 3
	Resultant Key Extension

Table 2-3 XORed Key Construction (SM?AK)

If no key is present in the key register indicated by key register number, a response code of key number error is returned. If the key in the key register indicated by key register number was not loaded via the manual key entry method, a response code of key number error is returned.

If the key parity specified in the Initialise Key command was C (to be checked), the key component is checked for even parity. If any octet has an odd parity, a response code of key parity error is returned.

The resulting key is checked against the set of DEA-1 weak, semi-weak and potentially weak key values. If this check fails, a response code of key (semi / potentially) weak error is returned.

Prior to storing the key, it is adjusted according to key parity specified in the Initialise Key command. If key parity is C (check parity), the key is checked for odd parity. If the key has even parity or no parity, a response code of key parity error is returned. If key parity is N (no parity), the key is not adjusted. If key parity is S (set parity), the key is adjusted to have odd parity.

The check digits test pattern is encrypted with the key component. The resultant cipher-text is returned in key component check digits.

The check digits test pattern is encrypted with the resultant key. The resultant cipher-text is returned in key check digits.

Please note that only six of the check digits are returned for both of the previous check digit fields, the remainder of the digits are set to zero.

2.7.2. SM?GF - Generate Firmware Authentication Code

Function

The firmware authentication code is returned for a given plain text firmware authentication key.

Request Format

Description	Rep	Range	Length
Destination device	A	SM	2
Request indicator	S	?	1
Command code	A	GF	2
Key register number	N	01-99	2

Response Format

Description	Rep	Range	Length
Source device	A	SM	2
Response indicator	S	!	1
Command code	A	GF	2
Response code	N		2
Firmware authentication code	AH		8

Description

The FAK is retrieved from the register indicated by key register number and used to authenticate the security module firmware.

If the register indicated by key register number is empty, a response code of key number error is returned.

If the register indicated by key register number does not contain a clear working (type I) key, a response code of key type error is returned.

A 32-bit firmware authentication code (FAC) is calculated over the board firmware (i.e. all processor instruction codes plus all constant values) using the ANSI X9.9/X9.19 algorithm (without data stream editing), the key indicated by key register number and an initialisation vector consisting of 64 zero bits. The resultant FAC is returned in firmware authentication code.

2.7.1. SM?GK - Generate Single-Length DEA-1 Key

Function

A single-length DEA-1 key is generated and stored in a key register.

Request Format

Description	Rep	Range	Length
Destination device	A	SM	2
Request indicator	S	?	1
Command code	A	GK	2
Key register number	N	01-99	2
Key type	A	C-N	1
Key parity	A	S,N	1
Parent key register number[a]	N	01-99	2
Encryption method	A	T	1

Response Format

Description	Rep	Range	Length
Source device	A	SM	2
Response indicator	S	!	1
Command code	A	GK	2
Response code	N		2
Encrypted key	AH		16
Key check digits	AH		16

Description

In the case of a clear working key (type I), parent key register number is ignored. The parent key register number must be present for all other generated key types. If the encryption method is T (triple), the key held in parent key register number + 1) must be present and in the same domain as the key held in parent key register number. If the parent key (pair) is not present, a response code of key number error is returned. If the parent key (pair) and the load key are not in the same domain, a response code of key number error is returned.

A MEK (type A) may not be generated by this command. If a KEK (type B), MEK extension (type J) or KEK extension (type K) is to be generated, the key held in parent key register number key must be a MEK (type A) or a KEK (type B). For all other valid key types, this key must be a KEK. If triple encryption is used, the key in held in parent key register number + 1) must be a MEK extension (type J) if the key held in parent key register number is a MEK, and a KEK extension (type K) if the key held in parent key register number is a KEK. If any of these conditions are violated, a response code of key type error is returned.

If the parent key was stored with parity, the security module performs a parity check on the parent key to ensure the validity of the key. If this check fails, a response code of key integrity error is returned.

The security module generates a 64 bit random value and checks this value against the set of DEA-1 weak or dual values. If the key is weak or dual, the process is repeated.

If key parity is N (no parity), the key is not adjusted. If key parity} is S (set parity), the key is adjusted to odd parity.

If the register indicated by key register is not empty prior to the store of the generated key, the register and all its descendants are cleared. The resultant key, parent key register number and key

[a] not validated for key I

type are stored in the register indicated by key register number. If the stored key is not a clear working key (type I), the register is added to the linked list of children keys of the register indicated by parent key register number.

If the generate key operation completed successfully, the key generated successfully response code is returned. If the key is a single length key with no parent key or is not a single length key, a zero vector is encrypted with the key set. For single length keys with parent keys a MAC shall be done on the key in the register indicated by key register number, using the key's parent key as the MACing key. The resultant check / MAC digits are returned in key check digits. Please note that only six of the check digits are returned, the remainder of the check digits are set to zero.

2.7.2. SM?FK – Fetch Single-Length DEA-1 Key

Function

A single-length DEA-1 key is fetched from the specified key register.

Request Format

Description	Rep	Range	Length
Destination device	A	SM	2
Request indicator	S	?	1
Command code	A	FK	2
Key register number	N	01-99	2
Key parity	A	S,N	1

Response Format

Description	Rep	Range	Length
Source device	A	SM	2
Response indicator	S	!	1
Command code	A	FK	2
Response code	N		2
Encrypted key	AH		16
Key check digits	AH		16

3. Appendix A - TSM410 Product Overview

3.1. Introduction

The Incognito TSM410 is a secure cryptographic coprocessor that offers high-performance, high-security services in the arena of transaction security.

Designed to address top-end transaction processing requirements, the Incognito TSM410 is capable of 50 pre-paid vending transactions per second (TPS). Each model is installed into a Windows-based server as a PCI peripheral.

The TSM410 forms part of Prism's Incognito Transaction Security family. The Incognito range addresses end-to-end transaction security requirements.

3.2. Security

The TSM410 is a Tamper-Responsive Security Module (TRSM) that complies with ANSI, ISO and FIPS (Federal Information Processing Standards) requirements.

The TSM410 has been certified to FIPS 140-2 Level 3 software and Level 4 hardware.

Note: FIPS 140-2 Level 3 software certification has been awarded to the TSM410 boot-loader application. This gives customers the assurance that only authorised applications may be executed on the TSM410. However, API security for the loaded application must be taken into account separately. In an attempt to maximise the security of the STS API certain restrictions have been implemented. These restrictions relate mostly to the distinction between initial key management and operational key management. The API definition indicates these restrictions.

3.3. Performance

The Incognito TSM410 is capable of delivering a performance of 50 TPS in the pre-paid dispensing environment.

3.4. Application Services

Applications integrate with a TSM410 through a network service called Conductor. The programming interface to the 410's cryptographic services is the STS API (detailed in this document).

Conductor provides device management functions for the TSM410, and offers client applications a consistent communication interface. With Conductor it is easy to configure a cold standby module.

3.5. Application Programmer Interface

NOTE: The following is applicable to communication with the TSM410 via the serial port only. For operational communication with the TSM410 over the PCI interface, it is necessary to install and communicate with Conductor. Please refer to "EDP V1 Specification.pdf" (PR-D2-Dat-1003) on the provided support CD for further information.

The user can access the functions supported by the security module via a message based application programmer interface (API). Careful consideration has been given to ease of programmability and the rapid integration of the security module into the target application environment. Each command used to invoke a function on the security module consists of a request/response message pair. To execute a command, the host computer sends the request associated with the command over the asynchronous communications link. Upon completion of the command, the security module returns the response message associated with the command over the communications link. If the security module cannot recognise the request message received, it returns a general error response message to the host computer. All request messages to the security module are validated before the command is executed.

All characters in the message are in ASCII representation, terminated by a message termination character (0x0D - carriage return). This means that the user does not need to write any code to evaluate any features of the security module, but could interact with the security module using a dumb asynchronous terminal or using a terminal emulation package on a personal computer. Using a message termination character ensures that message synchronisation recovery from lost characters is guaranteed.

All requests to and responses from the security module are prefixed by standard fixed headers. The request header contains the destination device identification, the command code and a request indicator. The response header contains the source device indication, the command code, the returned status code and a response indicator. The header information and the fact that the messages are in ASCII representation promote ease of monitoring, tracing and analysing (via e.g. a data scope) messages to and from the security module.

4. Appendix B - Communications Interface

The TSM410 communicates with external equipment either via an asynchronous serial communications channel, or via the PCI bus interface.

NOTE: The following is applicable to communication with the TSM410 via the serial port only. For operational communication with the TSM410 over the PCI interface, it is necessary to install and communicate with Conductor. Please refer to “EDP V1 Specification.pdf” (PR-D2-Dat-1003) on the provided support CD for further information. While communication with Conductor will still use the same message formatting as for the serial interface, it may be required that an additional header be pre-pended to the command.

4.1. Communication Parameters

The communications parameters used by the TSM410 serial interface are given in the table “Table 4-1 Communication Parameters” below.

Parameter	Values
Transmission rate (bps)	9600
Character length (bits)	8
Parity	None
Stop bits	1
Flow control	None
Checksum	On

Table 4-1 Communication Parameters

4.2. Recommended Interface

1. Set retry-count to zero.
2. Flush the receive buffer. Send the [command + CRC + CR] to the security module.
3. Wait for the response. If there is a response within the allowed time (TIMEOUT), go to step 4. Otherwise, increment the retry count. If the retries exceeds MAX_RETRIES, return a device failure. Terminate interface.
4. If the response CRC is incorrect, send [GL?RR + CRC + CR] to the security module, and go to step 3.
5. If the response header matches the command header (but with a '!' instead of the '?'), return the response.
6. Increment the retry count. If the retries exceeds MAX_RETRIES, return a device failure. Go to step 2.

Notes: **The value for TIMEOUT depends on the STS command being executed.**
MAX_RETRIES should be at least two.

5. Appendix C - CRC Computation (Sample Code)

The C code given below can be used to compute the CRC of a message:

```
static unsigned BitsSet (unsigned char ch)
{
   unsigned n;

   n = 0;
   while (ch)
   {
      n += (ch & 1);
      ch >>= 1;
   }
   return(n);
}

unsigned CRCof (const char *message, unsigned len)
{
   unsigned i;
   unsigned crc;
   unsigned char k;

   crc = 0;
   for (i=0; i<len; i++)
   { k = (unsigned char)(message[i]) ^ crc;
      crc = (crc / 256) ^ (k*128) ^ (k*64);
      if ((BitsSet(k) & 1) != 0)
         crc ^= 0xC001;
   }
   return(crc);
}
```

6. Appendix D - Cryptographic Algorithm Support

The TSM410 supports the Data Encryption Standard (DES), as adopted by ANSI and ISO, with the electronic code book (ECB), cipher block chaining (CBC) and cipher feedback (CFB) modes of operation. This algorithm is hereafter referred to as DEA-1.

6.1. DEA-1 Cryptographic Support

The security module supports DEA-1 secret cryptographic keys and functions on or with these keys. All keys are stored in a physically secure non-volatile area in memory. No support is provided for secure key storage external to the security module.

6.1.1. DEA-1 Key Support

6.1.1.1. DEA-1 Key Classes

Three classes of DEA-1 keys are supported. This classification facilitates a three (or more) level hierarchical key management scheme with respect to the exchange of keys and/or data with other compatible cryptographic facilities over non-secure communications or storage media. These classes are:

Working keys:	Working keys are used to perform cryptographic functions on user data.
Key exchange keys:	Key exchange keys (KEKs) are key encrypting keys used to encrypt working keys when they are transmitted from one cryptographic facility to another over a non-secure communications medium. A KEK may also be used to encrypt another KEK for transmission.
Master exchange keys:	Master exchange keys (MEKs) are key encrypting keys used to encrypt KEKs when they are transmitted from one cryptographic facility to another over a non-secure communications medium. A MEK may also be used to encrypt another MEK.

6.1.1.2. DEA-1 Key Types and Variant Vectors

As already established, a minimum of three classes or levels of DEA-1 keys are provided for. The number of levels may be arbitrarily extended, by replacing a MEK with a new MEK encrypted with its predecessor. Additional KEKs can also be encrypted under each other.

This hierarchical system makes use of the fact that the key universe is partitioned, according to function, into different key spaces. A key type defines each key space.

Key Type Description	Key Type Abbreviation	Base Key Type	Extension Key Type
Master exchange key	MEK	A	J
Key exchange key	KEK	B	K
Message working key	MWK	C	
PIN encryption key*	PEK	D	L
STS Default Vending Key		E	
STS Unique Key		M	
STS Common Key		N	
File working key*	FWK	F	
PIN generation key*	PGK	G	
PIN verification key*	PVK	H	O
Clear working key	CWK	I	

Table 6-1 Key Type Description

* Key Type not used in the STS API

Keys of different types cannot be used interchangeably. A key of a specific type can only be used with functions corresponding to its type. To enforce this rule, keys (excluding clear keys) are, when exchanged, encrypted under a parent key that is modified according to the type of the encrypted key under consideration. A 64-bit value is associated with each key type. These values are called variant vectors.

6.1.1.3. Key Form (Length)

The STS API supports both single-length and double-length key operations. Single length keys are used with the single encryption method of DEA-1, and double-length keys with the triple encryption method of DEA-1. A double-length key consists of two keys, usually referred to as the base key, left key or left half and the extension key, right key or right half. Triple encryption of data, where the left half and right half of a double length key are the same key is equivalent to single encryption with that key.

The Single Length Key Types A to M are associated with single-length keys. The Key Extension Types J to O are used to form double length keys with the respective single length base Key Type. In the case of single encryption, the parent key is modified by performing an exclusive-OR operation with the variant vector associated with the key type of the encrypted key. In the case of triple encryption, the double-length parent key is modified by performing an exclusive-OR operation on each 64-bit half of the parent key with the variant vector associated with the key type of the encrypted key (half). Such a modified key is referred to as a key variant.

6.1.1.4. DEA-1 Key Storage

The DEA-1 keys are maintained in the non-volatile, physically secure environment of the security module.

The security module stores DEA-1 keys and their supporting control information in direct access key storage registers. Key register numbers (1 - N) identify these registers, where N is ‘999’ for the STS04A implementation. The DEA-1 functions, used to perform various DEA-1 key management and cryptographic operations, referenced these keys by this key register number.

For double-length keys, the two halves of a double-length key are stored in 2 adjacent key registers - the left half in register n, and the right half in register n+1. This means that for any key register (n) that has a double-length key stored in it the next register (n+1) is used and should not be referenced directly. Key operations will however work on these (n+1) key registers as per normal and double length keys can be deleted a key register position n if key register position n+1 is clear / modified / etc.

A key register contains the following user accessible information:

- The encrypted key value
- The key type
- The *parent-key register* number.

In addition:

- The parity with which the key was stored
- The method of key loading (i.e. manual components or automatic encrypted)
- The encryption method with which the key was loaded (i.e. single or triple encrypted)

When loading a key into a key register, the user can specify whether or not the key's parity should be checked and whether or not the key should be stored with parity. If the parity check fails, the TSM410 aborts the current operation and returns a response code indicating a key parity error. If a key is stored with parity, a parity check is performed on the key whenever it is subsequently used. If this parity check fails, the TSM410 aborts the current operation and returns a response code indicating a key integrity error.

The key registers can be partitioned into a number of domains at production time. A domain consists of a range of consecutive key registers. No key register may fall in more than one domain. A variant vector table and a capability table are associated with each domain. The variant vector table defines the variant vectors associated with the key types of the keys within the specified domain. The capability table defines the cryptographic operations allowed using the keys within the specific domain. The variant vector table and the capability table for each of the domains are set to user specified values during production time.

6.1.2. DEA-1 Function Support

6.1.2.1. DEA-1 Key Storage Functions

A DEA-1 key can be loaded into a key register in one of the following ways:

- As separate key components which are combined (exclusive-OR) within the designated key register (manual)
- Encrypted under a DEA-1 key (automatic)
- Encrypted under a DEA-2 key (automatic).

A DEA-1 key is loaded into a key register encrypted under a parent DEA-1 key according to the type and variant vector rules previously outlined. The key type, key parity, key register number (in which the key is to be loaded) and the parent key register number must be given. A load of a DEA-1 key automatically overwrites the current contents of the key register, destroying in the process all descendent (child) keys of the overwritten key. This includes key registers containing the right half of a double-length key; in this regard a double length key can be cleared by clearing / modifying the contents of the register containing either the left or right half of the double-length key.

When a key is used for a specific function, the key register number and the associated key type are used to verify whether or not the desired function can be performed with the selected key.

When loading a MEK (type a), the destination key register number must be the same as the parent key register number, and the parent key register must contain a MEK. This is necessary since the new MEK must replace the old MEK.

A key register may be cleared. If this happens, all descendent key registers of that register are cleared.

All keys loaded into the security module encrypted under a parent key, may be fetched from the key register. Such a key is returned to the user encrypted under its parent key.

A new key of a specified type (excluding a MEK) may be generated under a designated parent key.

By encrypting the 64-bit check digit test pattern with the key the validity of a key can be checked. This process yields the check digits of the key. The check digits test pattern is set to a value of all zeroes at production time. Please note that only 6 check digits will ever be returned from the module. The remaining 10 digits will be set to zero in order to avoid the possibility of an adversary gaining information concerning the keys by obtaining the check digits.

7. Appendix E – Data Dictionary

Data Dictionary

Abbr./Acr./Def.	**Abbreviation, Acronym or Definition**
API	Application Programmer Interface
ATR	Answer To Reset
CBC	Cipher Block Chaining [6]
DAC	Device Authentication Code
DAK	Device Authentication Key
DEA	Data Encryption Algorithm [5]
DES	Data Encryption Standard [5]
ECB	Electronic Code Book [6]
EPS	Electronic Payment System
EPT	Electronic Payment Terminal [2]
FAC	Firmware Authentication Code
FAK	Firmware Authentication DES Key
FWK	File Working Key
ISA	Industry Standard Architecture
ISO	International Organisation for Standardisation
IV	Initialising Variable [6]
KEK	Key Exchange Key
MAC	Message Authentication Code
MEK	Master Exchange Key
MWK	Message Working Key
NBS	National Bureau of Standards
PEK	PIN Encryption Key
PAN	Primary Account Number
PCA	Prism Cryptographic Adapter
PCM	Prism Cryptographic Module
PIN	Personal Identification Number
POS	Point of Sale/Service
PVK	PIN Verification Key
PVV	PIN Verification Value
TRSM	Incognito Tamper Responsive Security Module
TSM410	High Performance, High Security Incognito Transaction Security Module

8. Appendix F – Return Codes

Return (Error) Codes

Code	Description
0	**SUCCESSFUL** The command executed successfully.
1	**DEVICE FAILURE** There was some hardware or general failure.
2	**FORMAT ERROR** The format of the data in the command is incorrect.
3	**COMMAND TIMED OUT** The command could not complete within the specified time.
4	**KEY NUMBER ERROR** The specified key register is empty or does not exist.
5	**KEY TYPE ERROR** The type of key is invalid for the operation to be performed.
6	**KEY INTEGRITY ERROR** The key has been corrupted.
7	**KEY PARITY ERROR** The parity of the key is incorrect.
8	**REDEFINITION FAILED** The particular attribute may not be redefined.
9	**INPUT DISPLAY FAILED**
10	**KEYPAD ENTRY CANCELLED**
11	**INVALID FUNCTION KEY MASK**
12	**INVALID AMOUNT FORMAT**
13	**READER NOT SUPPORTED**
14	**CARD NOT READ**
15	**MESSAGE CYCLE BUFFER VIOLATION**
16	**INVALID PIN**
17	**DEA2 PUBLIC KEY ERROR**
18	**INVALID PASSWORD**
19	**INVALID REPRESENTATION**
20	**CHECKSUM ERROR** The CRC of the command is incorrect, and the command should be retransmitted. This code is only used with a GL!ER response.
21	**INVALID REQUEST HEADER** The header in invalid, or the command is not supported. This sometimes happens if the command has been corrupted. This code is only used with a GL!ER response.

Return (Error) Codes	
Code	**Description**
22	**MAX KEY COMPONENTS**
23	**INVALID DOMAIN**
24	**KEY TOKEN ERROR**
25	**WEAK KEY ERROR**
26	**RSA / DES KEYS NOT CLEARED**
27	**PASSWORD LOCKOUT**
28	**PASSWORD NOT SET ERROR**
29	**PASSWORD INTEGRITY ERROR**
30	**INVALID STS TOKEN** The STS token CRC is incorrect.
31	**INSUFFICIENT CREDIT BALANCE** The module does not have sufficient credit to execute the command.
32	**CREDIT UPDATE BLOCK FORMAT ERROR** The format of the credit update block is incorrect.
33	**INVALID SECURITY MODULE ID** The operation cannot be performed on this module, as the module ID specified in different.
34	**INVALID CREDIT UPDATE SEQUENCE NO** The sequence number in the credit update block is incorrect.
35	**CREDIT OVERFLOW** The maximum STS credit balance has been exceeded.
37	**TARIFF / CREDIT INTEGRITY ERROR** The tariff / credit has become corrupt.
38	**NO TARIFF CREDIT DATA** No tariff / credit data exists.
39	**MAX CREDIT DOMAINS REACHED** The maximum number of credit domains allowed for the module has been reached.
40	**INVALID DATE ERROR**
41	**PAM CREDIT UNDERFLOW**
42	**PAM TARIFF EXPIRED**
50	**KEYPAD ENTRY TERMINATED**
51	**KEYPAD ENTRY CLEARED**
52	**KEYPAD ENTRY ERROR**
60	**INVALID TERMINAL** The terminal number is invalid.
61	**RSA BLOCK ERROR** The data in the RSA block is invalid.
64	**BALANCE ERROR** The balance amount is incorrect.
65	**TRANS SEQUENCE NUMBER ERROR** The transaction sequence number is incorrect.
66	**INVALID CARD TYPE** The card type is incorrect.
67	**INVALID OPTION** The option code is not supported.

Return (Error) Codes	
Code	**Description**
68	**RANDOM NUMBER ERROR** The random number is incorrect.
69	**PROTOCOL SEQUENCE ERROR** The command has been executed out of sequence.
70	**REVERSAL ERROR** The transaction reversal could not be performed.
71	**PIN BLOCK ERROR** The PIN block data is not in the correct format.
72	**KEY REGISTER ERROR** There is some problem with a key register. Usually it means the register is empty of corrupt. This error code is not used for the standard key registers.
73	**INVALID AMOUNT** The amount is invalid.
74	**PIN ERROR** The PIN is invalid.
75	**TOKEN INTERFACE ERROR** The security module cannot communicate with the smart card (or other token).
76	**TOKEN FORMAT ERROR** The format of the smart card (or other token) is incorrect.
77	**AUTHENTICATION ERROR** The authentication function failed.
78	**CHECK DIGIT ERROR** The check digits for the key are incorrect.
79	**NO CARD PRESENT** No smart card is present.
80	**INVALID TOKEN** The token is invalid.
81	**SIGNATURE ERROR** The signature was not verified.
82	**KEY PROTOCOL ERROR** The key protocol rules were broken.
83	**PUBLIC KEY ALREADY PRESENT** The Public Key component is already in key storage, or key storage in use
84	**PUBLIC KEY NOT LOADED ERROR** The Public key failed to load
97	**COMMAND DISABLED ERROR** This Error code will generally be returned if a command is not available over a particular interface.

9. Appendix G – New data formats

This section describes the new data formats that are used in conjunction with the STS04A firmware.

9.1. PKC1 Public Key Certificate

Field	**Length (bytes)**	**Type & Contents**
Header	1	‘4B’
Padding	Variable	‘BB’
Boundary	1	‘BA’
Message:		
Magic	4	4AN = ‘PKC1′
Expiry	4	4 bytes YYYYMMDD in BCD[1] indicates the last day on which the certificate is valid
Public exponent	4	4 bytes in big endian order, must be odd
Modulus	128–208	128 to 208 bytes in big endian order, top byte must be >= 0×80
SHA-256 hash	32	32 byte hash over *message*
Trailer	2	‘34CC’

9.2. INTX Increment Transaction Counter Instruction

9.2.1. INTX for XM?IC

Field	**Length (bytes)**	**Type & Contents**
Header	1	‘4B’
Padding	MODLEN-62	‘BB’
Boundary	1	‘BA’
Message:		
Magic	4	4AN = ‘INTX’
Module SN	8	8N
Nonce	8	8AH (use [clock seconds] to create)
TX increment	10	10N = amount by which to increase the transaction count
SHA-256 hash	32	32 byte hash over *message*
Trailer	2	‘34CC’

9.2.2. INTX for SM?IC (Legacy)

Field	**Length (bytes)**	**Type & Contents**
Header	1	‘4B’
Padding	MODLEN-62	‘BB’
Boundary	1	‘BA’
Message:		
Magic	4	4AN = ‘INTX’
Module SN	8	8N
Nonce	8	8AH (use [clock seconds] to create)
TX increment	6	6N = amount by which to increase the transaction count
SHA-256 hash	32	32 byte hash over *message*
Trailer	2	‘34CC’

10. Appendix G – API Revision History

Version	Description	Person	Date
28-A	Migrated document from PR-D2-0579 STS04A26 API Guide to create PR-D2-0769 rev 28-A. - Added SM?GD command, required by KMC and TeStMe application. - Allow SM?GS to be called over the PCI interface (previously was serial only). Firmware upgrade 1.28.0.0 (STS04A28) - Fixed bug where empty key registers may incorrectly report a key integrity error (06) instead of an invalid key (04).	TD/GG	2008/02/22
29	- Allow SM?FK to be called over the serial interface (previously disabled).Firmware upgrade 1.29.0.0 (STS04A29)	GG	2009/10/07
53	Added XM?IC and XM?QC which has support for 10 digit transaction counter values	CA	2011/01/26