TroopMaster Data

# Overview

This document is a work in progress for documenting the TroopMaster format.

TroopMaster provides ways of getting at the TroopMaster data. While most of the data can be exported as CSV the Outing Attendance doesn’t seems to be a choice. Also, there are multiple areas that each need to be exported individually instead of all at once.

Other means of getting at the data could be via the SOAR export but the SOAR also doesn’t contain outing attendance but may be a good choice if want Scout Information minus the outings.

So basically all but the outing attendance you can get by doing an Export all the choices to csv and then doing processing on them.

# TM4 Data

The TroopMaster format on disk is spread out between multiple files. Each file equates to a type of item to enter or in database Terms a table. However, there are differences in the uber format of the file based on which one of the files it is and some of the files (ScoutData and AdultData) has their data encrypted.

The tm4 files appear to all be fixed record formats but there is a mix of some formats allowing to append records and other just fixed size to start with and instead of records written sequentially writes all then as Field1 for all the records, then Field 2, etc

The following table makes up the files in the TM4 data directory minus archives. Lots of files still need to be investigated.

|  |  |  |  |
| --- | --- | --- | --- |
| File | Purpose | Encrypted | Documentation |
| Activity.tm4 | List of Activities | No | [Activity.tm4](#_Activity.tm4) |
| ActivityDesc.tm4 | Activity descriptions | No | [ActivityDesc.tm4](#_ActivityDesc.tm4) |
| ActivityTotals.tm4 |  |  |  |
| AdultData.tm4 | Adults | Yes | [AdultData.tm4](#_AdultData.tm4) |
| AdultFund.tm4 |  |  |  |
| AdultIndex.tm4 |  |  |  |
| AdultMOS.tm4 |  |  |  |
| AdultOA.tm4 |  |  |  |
| Advancement.tm4 | Scout advancement | No | [Advancement.tm4](#_Advancement.tm4) |
| Attend.tm4 | Activity attendance | No | [Attend.tm4](#_Attend.tm4) |
| DBAccess.tm4 |  |  |  |
| DutyToGod.tm4 |  |  |  |
| EagleHonorRoll.tm4 |  |  |  |
| MBCData.tm4 |  |  | [MBCData.tm4](#_MBCData.tm4) |
| MBCIndex.tm4 |  |  |  |
| ParentData.tm4 | Parent Data | Yes | [ParentData.tm4](#_AdultData.tm4) |
| ParentIndex.tm4 |  |  |  |
| POCData.tm4 | Point of Contact Data |  |  |
| POCIndex.tm4 |  |  |  |
| RecharterInfo.tm4 |  |  |  |
| ScoutData.tm4 | Scout Data | Yes | [ScoutData.tm4](#_ScoutData.tm4) |
| ScoutFund.tm4 |  |  |  |
| ScoutIndex.tm4 |  |  |  |
| ScoutMOS.tm4 |  |  |  |
| ScoutOA.tm4 |  |  |  |
| SeaScout.tm4 |  |  |  |
| TourPermit.tm4 |  |  |  |
| UnitInfo.tm4 |  |  |  |
| UserList.tm4 |  |  |  |
| Varsity.tm4 |  |  |  |
| Venture.tm4 |  |  |  |
|  |  |  |  |
|  |  |  |  |

## DBAccess.tm4

Todo: this is a “password” access file not a user file so not sure how important it is in the grand scheme of things since reader/writer could effectively ignore this. Depends on how other software would want to enforce viewing sensitive data fields and write permissions.

Contains the Database access passwords. File is organized so Fields are grouped together.

First field in the files is the Passwords with a length of 20d (including NULL). Fields start at file offset zero but the first record is left empty. (I think this is consistent with others).

If a field is written and smaller a null with written in proper length but the remainder of the record will not be cleared out.

a password of aaaaaa ends up being hashed as : WXB1aG2MM. It’s the same independent of record number so strictly based on the string itself.

or in hex (a == 0x61)

57, 58, 42, 31, 61, 47, 32, 4d, 4d (Not sure about the 57??) Seems to change on string length the offset in to the changes are different so like I think like record Number the string length scrambles

Password of bbbbb (b == 0x62) 4b, 62, 36, 34, 55, 30, 71

Password ccccc (c = 0x63) 50, 6c, 47, 50

Character, position, string length,

So I haven’t figured out the decryption of this. The record decrypt and the soar is just copied local for reference. What I do know is that

Hash for a character changes with length of string and position in the string

Other characters in the string do not affect how a single character gets hashed.

Hash for character is same independent of with Record its for.

String above with bbbb end up with b itself in the string so “something” has to end up in some cases being effectively zero either a mask that matches, or subtraction.

The Troopmaster tsi?.dll has a DecryptString which is what I suspect gets called but haven’t tried it.

Record decrypt algorithm is unknow

The decryption algorithm is basically if the byte is zero value then it is still zero, else based on record number and position of byte in the record. Look up a value in the dictionary and subtract the dictionary value from the byte value.

recordDecryptKey {

0x2f, 0x52, 0x66, 0x7a, 0x20, 0x31, 0x6b, 0x4f, 0x35, 0x38, 0x33, 0x59, 0x4c, 0x4d, 0x76, 0x26,

0x32, 0x63, 0x64, 0x65, 0x45, 0x6e, 0x70, 0x57, 0x54, 0x36, 0x50, 0x23, 0x55, 0x49, 0x62, 0x6f,

0x41, 0x51, 0x47, 0x58, 0x73, 0x5a, 0x61, 0x44, 0x74, 0x79, 0x69, 0x42, 0x48, 0x6c, 0x43, 0x37,

0x34, 0x71, 0x72, 0x43, 0x2d, 0x6d, 0x75, 0x53, 0x30, 0x4a, 0x68, 0x6a, 0x56, 0x77, 0x4b, 0x46,

0x39, 0x67, 0x78

};

public static byte[] SoarDictionary = new byte[]

{

0x38, 0x6f, 0x4c, 0x52, 0x6d, 0x36, 0x6a, 0x70, 0x79, 0x66, 0x56, 0x67, 0x41, 0x34, 0x30, 0x71,

0x70, 0x64, 0x4a, 0x48

};

Wonder if it’s the “DescyptString” algorithm.

## ScoutData.tm4

ScoutData list each individual Scout and their information. Data is encrypted ([Encrypted tm4 files](#_Encrypted_tm4_files)).

Each entry is a fixed size record of 0x17d8. Each field also appears to be a fixed length but haven’t broken the data yet into fields for confirmation.

When ScoutData is deleted the entry is nulled out. On entry if there is a nulled out entry it is reused. If not empty entry it’s appended to the end.

Parent information is also included. Just filling out scout parent data doesn’t affect the ParendData.tm4 file but there is a way to “link” but haven’t investigate the interaction.

Note: There is a ScoutIndex.tm4 file which usage is unclear. Maybe it truly is just a db index type file but haven’t seen any real data or reliance on it thus far. Other suspicion that it has to do with the “Linking” feature to relate scouts to adults. Which would explain why there is one for ScoutData, ParentData, and MeritBadge counselors.

## ParentData.tm4

Adult list each individual adult entered and their information. Data is encrypted ([Encrypted tm4 files](#_Encrypted_tm4_files)).

Each entry is a fixed size record of 0x3264 bytes. Each field also appears to be a fixed length but haven’t broken the data yet into fields for confirmation.

When AdultData is deleted the entry is nulled out. On entry if there is a nulled out entry it is reused. If not empty entry it’s appended to the end.

Note: There is a AdultIndex.tm4 file which usage is unclear. Maybe it truly is just a db index type file but haven’t seen any real data or reliance on it thus far.

## AdultData.tm4

Contains information about Adults in the Troop Parent’s may or may not be in Adult data.

## MBCData.tm4

list each Merit Badge counselor. Data is encrypted ([Encrypted tm4 files](#_Encrypted_tm4_files)).

Record size is Merit Badge Counselors is 0x874 record Size

## Advancement.tm4

Advancement has an entry corresponding to the record number of each scout in ScoutData.

Advancement record size is 0xAB20. Each field has yet to be determined.

Advancement

Out the box and when added first Scout was 0xAB27

When added second scout grew to 0x1564f

When added Scouts three and for grew to 0x2ac9f

Don’t know details of fields but so far:

When added a merit badge the date, counselor information. Merit badge is a word with the BSA Advancement ID used. For fingerprinting is 49d.

What Merit badge number was fingerprinting introduced as?

List of advancement IDs:

<http://meritbadge.org/wiki/index.php/BSA%27s_Merit_Badge_Numbering_System>

When a scout is deleted the corresponding record number in Advancment is null’d out. For all the nulls out I’m not sure if just the fact that the text records are null is enough or there is some way to truly tell if it’s a null/empty record that can be reused.

## Activity.tm4

Activity file with one entry is 75aef. I suspect it’s a fixed size with enough room for 2000 activities. File is laid out so each field is written as chunks. Aka something like Field1Record1, Field1record2, Field2Record1,Field2Record2.

From the TroopMaster docs

This option enables you to clear out your files when you reach the system limit of 2000 activities.

There is then an option to Activities>Set Prior Totals amounts to compensate for the missing activitIies. Kind of a Hack I'd say but prior totals based on the outings being deleted to be applied to the Scouts.

"To avoid having unwanted amounts being added to scout/adult “prior” totals, delete erroneous activities through the Activity Management dialog or by displaying the activity and clicking Delete."

## ActivityDesc.tm4

Haven’t really looked at but looks fixed size file with fixed size records. Doesn’t change as add/remove activities so I suspect it is to have a lengthy description so each record in Activity doesn’t have to be large enough to hold

## Attend.tm4

Keeps track of Activity attendance. This is an interesting one because from what I can tell it is never included in any exports and it’s something you’d want to export somewhere to analyze attendance for something like a golden tent stake.

Each Attend is up to 0xBB7 in distance. Looks like it’s just in order of Scout record but could just be the simple case. So for first Scout, first outing at byte 0 ix 0xA8FFFFFF. No Idea if always that or depends on outing type for roll up.

Added adult to outing put the first adult position at 0x3e8

So scouts get 0 - 0x3e7 with four bytes per scout

Adults get 0x3e8 - 0xbb7

Attendance grows with each outing

Size appears to be bb7 for each attendance but if take the total size of two and divide I get Bbo but one entry is bb7

Maybe there is some overhead bytes/padding but I also think this diff tool may not be showing everything in which case

If that’s the case then 250 scouts are the maximum in Troopmaster?? Which matches what I found online at least for Packmaster. Also the adult size matches the 500 limits on adults.

When deleted outing cleared out the attendance list but left room in the file be interesting to delete one in the middle and see what happens. I suspect the shift everything up.

When deleted an activity in the middle

First word of the Activity file changed from 1 to 2 (not sure if significant or just write number)

Looks like first word in the activity structure is the "Number" and that correlates back to the AttendData?

Data record that attended that outing gets a NULL written to the start of each field so it just nulls it out.

Attend at BB0 got nulled out.

Next outing looks for empty records and fills them in so record number in Attend always matches record in Activity

Note: if the scout doesn’t exist in the ScoutData then there can’t be an entry for the scout in the attend file. This seems to be the reason archived scouts drop out of existence for the outing roster lists. Not sure what would happen on a restore from archive but attend seems to be part of that. However, I don’t see how they would line up the outings other than their description and it would mess up the “totals design” on archive of outings.

## Encrypted tm4 files

ScoutData and AdultData have their records encrypted. Other tables may be encrypted as well. Both ScoutData and AdultData encrypt the same way so more than likely all decryption is done via the samechanism.

The decryption algorithm is basically if the byte is zero value then it is still zero, else based on record number and position of byte in the record. Look up a value in the dictionary and subtract the dictionary value from the byte value.

For Troop520 the dictionary and what it was on a box I installed the trial the dictionary was the same:

Note: I had to fix a typo in the test program at one point; not sure it made it into the doc

public byte[] DecryptDictionary = new byte[]

{

0x2f, 0x52, 0x66, 0x7a, 0x20, 0x31, 0x6b, 0x4f, 0x35, 0x38, 0x33, 0x59, 0x4c, 0x4d, 0x76, 0x26,

0x32, 0x63, 0x64, 0x65, 0x45, 0x6e, 0x70, 0x57, 0x54, 0x36, 0x50, 0x23, 0x55, 0x49, 0x62, 0x6f,

0x41, 0x51, 0x47, 0x58, 0x73, 0x5a, 0x61, 0x44, 0x74, 0x79, 0x69, 0x42, 0x48, 0x6c, 0x43, 0x37,

0x34, 0x71, 0x72, 0x43, 0x2d, 0x6d, 0x75, 0x53, 0x30, 0x4a, 0x68, 0x6a, 0x56, 0x77, 0x4b, 0x46,

0x39, 0x67, 0x78

};

Or in hex dump from TroopMaster.exe

0:000> db 100c3858

100c3858 2f 52 66 7a 20 31 6b 4f-35 38 33 59 4c 4d 76 26 /Rfz 1kO583YLMv&

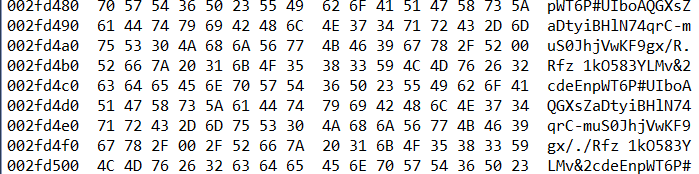
100c3868 32 63 64 65 45 6e 70 57-54 36 50 23 55 49 62 6f 2cdeEnpWT6P#UIbo

100c3878 41 51 47 58 73 5a 61 44-74 79 69 42 48 6c 4e 37 AQGXsZaDtyiBHlN7

100c3888 34 71 72 43 2d 6d 75 53-30 4a 68 6a 56 77 4b 46 4qrC-muS0JhjVwKF

100c3898 39 67 78 00 00 00 00 00-00 00 00 00 e8 35 25 00 9gx

I’d assumed the dictionary changes based on licensing or other data but since demo had the same key and in fact a matching key is in the binary so more than likely the key is the same for any Troopmaster data file.



### Decrypt Example

General algorithm in c# code to decrypt length number of bytes in “textToDecrypt”.

The bytes in textToDecrypt are encrypted when passed to Decrypt and on return they are unencrypted. The recordNumber is zero based and corresponds to the record position in the file.

public void Decrypt(int recordNumber /\* zero indexed record number \*/

, byte[] textToDecrypt, int length)

{

for (int index = 0; index < length; index++)

{

byte curByte = textToDecrypt[index];

if (0 != curByte)

{

byte decryptBytePos = (byte)

((index + decryptOffset) % DecryptDictionary.Length);

byte unEncryptByte = (byte)(curByte - DecryptDictionary[decryptBytePos]);

textToDecrypt[index] = unEncryptByte;

}

}

}

Test code to load and decrypt the scoutdata is.

public void SimpleDecryptExample()

{

int recordCount = 0;

int recordSize = 0x17d8; // size of each Scout record is 0x17d8 bytes

byte[] recordData = new byte[recordSize];

// path do scoutdata. this is based on c drive.

using (FileStream file = new FileStream(

@"C:\Troopmaster Software\TM4\DATA\scoutdata.tm4", FileMode.Open))

{

while (file.Position < file.Length) // loop reading recordSize bytes

{

file.Read(recordData, 0, recordSize);

Decrypt(recordCount, recordData, recordSize);

// data in recordData after Decrypt is clearText

++recordCount; // increment recordCount for next record.

}

}

}

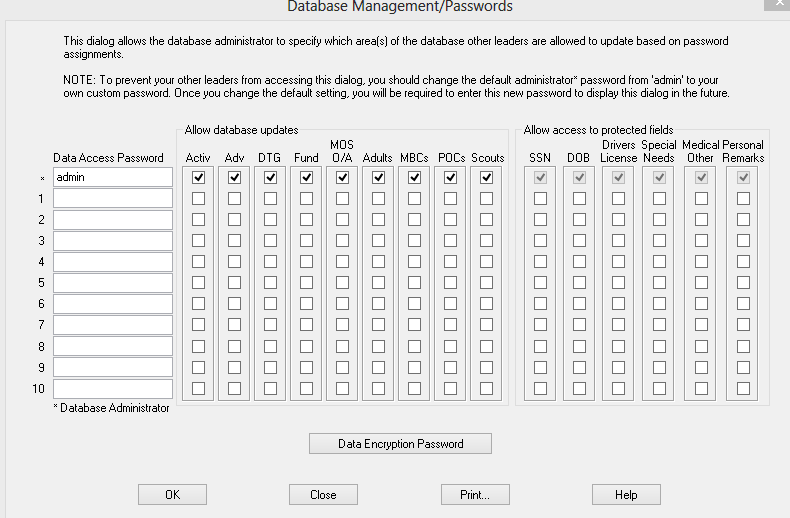
## Archiving

Troopmaster has a concept of archiving which serves as both a way to remove scouts from the day to day view and clear up entries that may be needed if the Unit is hitting the Troopmaster limits.

Haven’t delved into archiving but probably some interesting cases when restoring from archive with outings

# Data Access

Troop master allows data access and update permissions based on database managed/passwords



So you can give a data Access password for what roles you want people or a unique one for each person.

The toplevel password is the “root admin” and is the one allowed to even pull up the Data Access. From above TroopMaster has a hack that if the password is admin it treats the same as no password.

Passwords are in the DbAcess.tm4 file. I suspect if you just alter the dbaccess.tm4 file with a known hash of a password you can then just open Troopmaster and enter that one. Maybe just hash it back to admin.

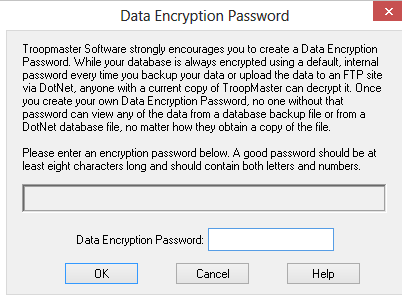
Note: As a test I changed the Database Encryption password to make sure it didn’t affect the hash and it does not.

If want to see what happens enter the following at dbaccess.tm4 offset 0x14

0x6e 0x72 0x50 -0x32 0x78 0x00

## Database Encryption Password

According to the dialog



This is the password for encrypting on ftp and backups. Haven’t verified but if dialog is true local files are always encrypted with the built in key but wanted to enable folks doing an FTP or backup for import or fetching from a remote site need to use.

The Database Encryption password doesn’t seem to be stored in the DBAccess nor do I see other files changed in the “Troop Master” install directory. It would make sense that it wouldn’t and is probably in the Windows Crypt store or something the PolyCrypt.dll is stashing away

# FTP Transfer File Format (TMD4DATA.bak)

When transferring over the wire (and possibly other places) Troop master uses a TMD4Data.bak and TM4Archive.bak file.

## PolarCrypto

The File is encrypted with encryption provided by polarcrypto.dll. I’m assuming that the encryption key for this is the magic key you define in .net. I’m not sure though how Troopmaster gets this if it’s part of the registration.

Interesting exports are:

000> x PolarCrypto\*

bp PolarCrypto!pcDeriveKey

bp PolarCrypto!pcReadDataFromFile

bp PolarCrypto!pcGenerateKey

bp PolarCrypto!pcCreateHash

bp PolarCrypto!pcCalculateKeyHash

bp PolarCrypto!pcSetKeyProperty

bp PolarCrypto!pcSetHashLength

PolarCrypto header file is attached:



Basic flow is that a call is make to a call to pcDeriveKey() to get a key and then pcReadDataFromFile to get the decrypted data.

Exact values passed to pcDeriveKey() isn’t known nor is the decrypt method. Current guess based on debugging is AES with either a 128 or 256 keyLength. According to Troopmaster docs/website it is 256.

The file is decrypted and written in unencrypted format in a tempDecrypt folder.

## TMD4Data.bak format

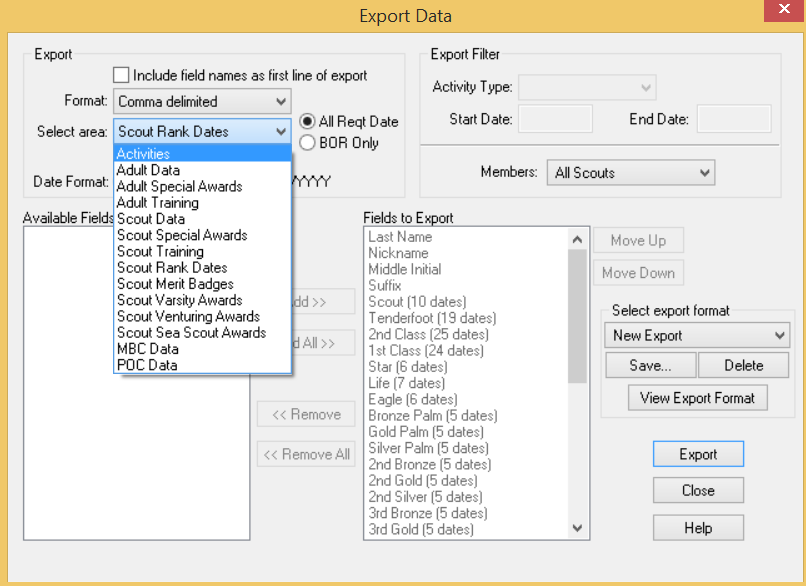
Decrypted tmd4data.bak file is made up of chunks and each chunk is uncompressed and placed with its corresponding name in the TM4\Data directory.

Format or the file for how to extract/decompress isn’t known. I suspect that the decompression is using Zip based on the export function names.

# TroopMaster Supported Export Formats

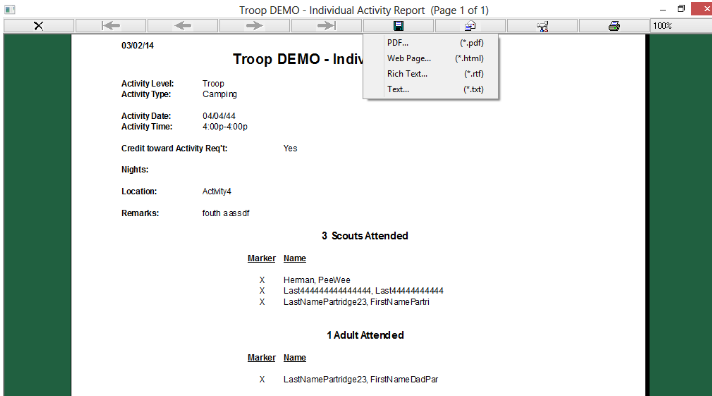
## CSV

Troopmaster provides support for exporting some of its information in CSV format.



## Reports

If you want to get creative you can also generate Reports and save as Text which could be parsed if you wanted but it would be custom parsing and since I can’t find a bulk create reports for all outings you’d have to do it one by one.



## Activity List (iCalendar)

Allows export of the current Activity List into ICalendar format.

## Automated District Management System (ADMS)

District Management export.

Beginning of file is: District Transfer V1.01.

Data is either compressed and/or encrypted.

<http://www.troopmaster.com/flyers/ADMS.pdf>

## SOAR

Soar format is an encrypted file that contains CSV formatted sections for ScoutData and Activity

The unencrypted content is:

SCOUTS,Last Name,First Name,Middle Name,Suffix,

43,Alayo,John,C,,John,

ADVANCEMENT

43,,,,,,,04/18/11,04/30/12,,,,,,Art,08/19/11,Fingerprinting,08/19/11,First

PARTIAL MB

43,

The data is encrypted with a simple XOR pattern

public static byte[] SoarDictionary = new byte[]

{

0x38, 0x6f, 0x4c, 0x52, 0x6d, 0x36, 0x6a, 0x70, 0x79, 0x66, 0x56, 0x67, 0x41, 0x34, 0x30, 0x71,

0x70, 0x64, 0x4a, 0x48

};

using (FileStream records = new FileStream(@"C:\Troopmaster Software\Export\soar.xfr", FileMode.Open))

{

using (FileStream output = new FileStream(@"C:\temp\soar\soarDecrypted.txt", FileMode.Create))

{

int recordSizetest = (int) records.Length;

byte[] decryptBytes = new byte[recordSizetest];

records.Read(decryptBytes, 0, decryptBytes.Length);

for (int offset = 0 ; offset < decryptBytes.Length; offset++)

{

byte curByte = decryptBytes[offset];

int posIndecrypt = offset % SoarDictionary.Length;

byte decryptByte = SoarDictionary[posIndecrypt];

byte newValue = (byte) (curByte ^ decryptByte); // ^ is the symbol for XOR

decryptBytes[offset] = newValue;

}

output.Write(decryptBytes, 0, decryptBytes.Length);

}

}

}

Scouts are assigned an ID that is used as a way to reference the Scout for Advancement and other sections. ID matches. id given looks like it matches the position of the Scout in the TroopMaster Scout.tm4 file. Adults are also given an ID

Categories exported are

SCOUTS

ADULTS

ADVANCEMENT

INIDIVDUAL PROGRESS

PARTIAL MB

## Troop Ledger

Allows export to Troop Ledger. Seems to just be ScoutData without advancement, etc.

## TroopMaster Traveler

Looks like export or sync to PocketPC and Palm.

# TroopMaster Limits

The way the Troopmaster data is stored it imposes limits on the amount of data that can be stored. The following are the limits documented in TroopMaster manual.pdf in the Troopmaster\tm4 folder

Scouts

- 250 scouts

- 100 special awards per scout

- 100 training classes per scout

- 300 scout training course definitions

- 136 merit badges per scout

- 15 partial merit badges per scout

- 15 leadership history positions per scout

Activities

- 24 unit activity types

- 30 unit activity levels

- 2000 total activities

Other

- 20 email message attachments

- 100 vehicles per tour permit

- 100 special awards per adult

- 100 training courses per adult

- 500 merit badge counselors

- 500 points of contact (POCs)

- 20 skills/interests per POC

# TroopMaster TSI exports

Troopmaster uses a TSI.dll which has the following exports:

Some interesting ones are:

bp TSI!CompressFileToZIP

bp TSI!UncompressZIPFile

bp TSI!UncompressOldZIPFile

bp TSI!CompressFileA

bp TSI!CMessage::ExtractFile

bp TSI!EncryptData

bp TSI!DecryptData

bp TSI!EncryptXOR

bp TSI!AttachPassword

bp TSI!DetachPassword

bp TSI!ValidateDecryption

bp TSI!EncryptString

bp TSI!DecryptString

bp TSI!EraseTempDecryptFolder

bp TSI!EncryptRecord

bp TSI!DecryptRecord

bp TSI!ReadText

bp TSI!ReadLine

bp TSI!ReadField

bp TSI!CheckPassword

bp TSI!GenerateFullPassword

# Useful Tools

vBinDiff - For comparing binary files

windbg – For debugging code