Developers Group

SERVER DOCUMENTATION

TeleFlow TAM Description

Programmers Guide

REVISION: 1.01

Confidential

Last update: October 15, 2003

Summary

Describes the file layout for TeleFlow Server

AUTHOR LIST

Sean Brothers, enGenic Corporation (604.639.6391 x 116)
Vern Baker, enGenic Corporation (604.639.6391 x 101)

SIGNED AUTHENTICATION

This document is verified accurate by:

enGenic
Fax: 604.639.6392

_____ _____
Name (signature) Date

_____
Name (printed)

A representative of

_____
Company Name

1. Only sign once this document is approved in its entirety
2. File to developer folder

CONTENTS

SUMMARY

Developers of TeleFlow Server Software require this document for creating Tam files, which contain the flowcharts. A TAM file is a series of records with a set number of fields.

This document is to serve as a method for sharing ideas and ensuring all aspects of development and documentation have been accounted for. Additional documents related to the project development may also be included as appendices.

During the design process, all parties involved will need to review the document to ensure accuracy. All changes should be ***redlined***, and then ***checked in*** by the other party once the document is returned by email.

FUNCTIONAL OVERVIEW

Definition of TeleFlow file formats

TAM FILE STRUCTURE

The following section describes the TAM file Record structure. There are two sections to each TAM. The first section in the Header, and the second is the Record section.

Header Section:

The header includes the introduction text that describes the current TAM. This section is used by TeleFlow Designer application to document the flowchart, and to record the version number of the file type.

Subsection 1: Intro

The Intro subsection is four lines. Each of the four lines is followed with a CrLf (#13#10) combination.

```
TeleFlow
Copyright 1994-2003 enGenic Corporation. All rights reserved.
ENGTF@TAM0200 TeleFlow Application (module)
☻
```

The first "#2" (Inverse Happy Face) encountered indicates the end of the Header section, and the start of the Record section. For the Server, this section can remain blank with the exception of the "#2#Cr#Lf".

Line 1: Application Name
Line 2: Copyright information
Line 3: Version of file (in this case 0200). Future versions may have a different layout or additional fields. XML will probably be adopted, but following version 0200 conventions closely.
Line 4: End of Header

Subsection 2: Notes

Designer will save a few notes about the TAM. For the Server, this section can remain blank with the exception of the "#Cr#Lf".

```
D:\TeleFlow\Test Apps\Tam\TFTam.Tam
```

Subsection 3: Internal

The internal section of the file contains information about the version of the software, and has provisions for adding additional concepts such as encryption to the TAM files. Ensure this string is “ENGTF@TAM020000000” with 44 spaces following. This section must be wrapped with EOF “1A” (#26) characters. Please refer to the TAM Dump section below.

```
→ENGTF@TAM020000000                                          →
```

Record Section:

The record sections starts immediately following the second EOF “1A” (#26) character. From there records are created with fields that are separated by the “#1” (Normal Happy Face) character. There are a total of 31 fields that are followed with the “#1” character. The 31st field is followed with a “#1”, and then a “#2” to represent the end of the record. The next record starts with the function number defined the Step Functions Definition section. Refer to the TAM Dump section for an example of the file layout.

1. Function

```
> ** This field is used by the Server **
> The Function definition
> See Step Function Definitions section
if = -1 then this icon does not need to be connected to another step
if = 0  then this icon is waiting to be connected to another step
if > 0  then this icon is already connected to step value(3)
```

2. Step Label

```
> ** This field is used by the Server **
> Text label used only by the icLabel function
> Must be blank in all other steps
```

3. Entry Index (Current)

```
> ** This field is used by the Server **
> The entry of the Entry
> Must be a unique number, and can be considered the Primary Key
```

4. Next Entry Pointer (Next)

```
> ** This field is used by the Server **
> Pointer to logically connect next Record/Step
> The Entry Index of the next record goes here.
- if lkDead       or -6    Icon has been removed or deleted
- if lkHide       or -5    Hide action from map
- if lkUnConn     or -4    Unconnectable, will not have a next step
- if lkConnDangle or 0     Has a dangler.
  If Next is populated greater than 0, then the icon is connected
- if lfNotFound   or -1    Indicts a not found
```

5. Parent Entry Pointer (Child)
```
> ** This field is used by the Server **
> Points to owner Entry Index (Parent) by a child step
> Set to 0 if a Parent step
> Also called Action steps
> Indicates Actions that hang from the Parent step
> Refer to Action Steps Reference for appropriate settings
- GetTT = Parent, it has TT1..TT0,TTStar etc
```

6. Field 1 – Numeric Value N01
```
> ** This field is used by the Server **
> Property value for Step
```

7. Field 2 – Numeric Value N02
```
> ** This field is used by the Server **
> Property value for Step
```

8. Field 3 – Numeric Value N03
```
> ** This field is used by the Server **
> Property value for Step
```

9. Field 4 – Numeric Value N04
```
> ** This field is used by the Server **
> Property value for Step
```

10. Field 5 – Numeric Value N05
```
> ** This field is used by the Server **
> Property value for Step
```

11. Field 6 – Numeric Value N06
```
> ** This field is used by the Server **
> Property value for Step
```

12. Field 7 – Numeric Value N07
```
> ** This field is used by the Server **
> Property value for Step
```

13. Field 8 – Numeric Value N08
```
> ** This field is used by the Server **
> Property value for Step
```

14. Field 1 – String Value S01
```
> ** This field is used by the Server **
> Property value for Step
```

15. Field 2 – String Value S02
```
> ** This field is used by the Server **
> Property value for Step
```

16. Field 3 – String Value S03
```
> ** This field is used by the Server **
```

```
> Property value for Step
```

17. Field 4 – String Value S04
```
> ** This field is used by the Server **
> Property value for Step
```

18. Field 5 – String Value S05
```
> ** This field is used by the Server **
> Property value for Step
```

19. Field 6 – String Value S06
```
> ** This field is used by the Server **
> Property value for Step
```

20. Field 7 – String Value S07
```
> ** This field is used by the Server **
> Property value for Step
```

21. Field 8 – String Value S08
```
> ** This field is used by the Server **
> Property value for Step
```

22. Field 9 – String Value S09
```
> ** This field is used by the Server **
> Property value for Step
```

23. Field 10 – String Value S10
```
> ** This field is used by the Server **
> Property value for Step
```

NOTE: Every field after this point is not required by the Server

24. Entry Attributes
```
> Defines aspects graphical drawings and style
> << Used by Designer only >>
> 0 for Server
```

25. Notes
```
> Notes entered by Designer
> << Used by Designer only >>
> Blank entry for Server
```

26. Entry X Loc
```
> Defines X location of Step on flowchart
> << Used by Designer only >>
> 0 for Server
```

27. Entry Y Loc
```
> Defines Y location of Step on flowchart
> << Used by Designer only >>
> 0 for Server
```

28. Tack X Loc
    > Defines X location of Step's tack (blue tack) on flowchart
    > << Used by Designer only >>
    > 0 for Server

29. Tack Y Loc
    > Defines Y location of Step's tack (blue tack) on flowchart
    > << Used by Designer only >>
    > 0 for Server

30. Connect Shape
    > Defines shape of connecting line to next step
    > << Used by Designer only >>
    > 0 for Server

31. Parent Connect
    > Defines color and shape of connecting line back to Parent step
    > << Used by Designer only >>
    > 0 for Server

STEP PROPERTIES

Step properties include N01-N08 (also referred to as U01-U08) and S01-S10. These properties are entered by the Designer into the records that are within the TAM file. Each Parent step may have property values, but Action steps will not. Properties are definitely for each step. In order to determine what the property name is for each step, you will require TeleFlow Designer to view the field names. To do this, ensure the TFDesigner.Tae file (available for TeleFlow Developers) is put in the same directory as TFDesigner.Exe. This unlocks the F2 key which will pop up the current field information for the Steps Properties page in Designer.

Fields, both numeric and string, are terminated with by the “#1” (Normal Happy Face) character, and by no other means. The record is terminated by a “#2” (Inverse Happy Face). Please refer to the TAP Dump section for examples of records. In this section, the second “1A” character indicates the beginning of the Start step (the green oval that indicates the starting point of the flowchart “icProcStart”). The Start step is a good example of a step with non-populated properties.

String Values

Fields that contain string values may be blank (0 characters), or be populated with any number of characters. Values of string files can be easily identified using the F2 key to related fields to their values in the Step Properties window.

Numeric Values

Fields that contain numeric values must contain a valid number (i.e.: -1,0,1 etc). When Designer creates the N field values it automatically defaults all values to 0. Values for fields are not readily determinable due to conventions for the user interface in the Properties Edit screen. Provided in this section are the values for some of the key steps. To find the values, place focus on a property, and press F2, reference the field number (i.e.: 1: result) to determine through sequential reference what the value should be.

The following code shows constants (denoted with ic????) whose naming conventions are not necessarily indicative of their true functions. Please refer to the Step Definitions section for a proper description of their function.

Please note in the case of vfaFields, the following values are used:

```
vfrNMSVOX   = '1';
vfrDIAVOX   = '9';    // Supported
vfrMUWAV    = '2';    // Supported
vfrNMSVCE16 = '3';
vfrNMSVCE24 = '4';    // Supported
vfrNMSVCE32 = '5';
```

```
 vfrNMSVCE64 = '6';
 vfrNMSVCEMU = '7';
 vfrNMSVCEA  = '8';
 vfrEXT      = '0';    // Supported
 vfrGSM610   = '11';   // AI-Logix



// Reference for N field values

icAppl:
  begin
       case uFieldNo of
       1: result := PropertyFieldTranslate(uDirection, uNumber, GlobBoardOrderArray);
       end;
  end;
icAnswerPhone:
  begin
       case uFieldNo of
       1: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
       end;
  end;
icPickUpLine:
  begin
       case uFieldNo of
       1: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
       end;
  end;
icPlaceCall:
  begin
       case uFieldNo of
       1: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
       2: result := PropertyFieldTranslate(uDirection, uNumber, [1,0]);
       3: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
       4: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
       end;
  end;
icHangUp:
  begin
       case uFieldNo of
       1: result := PropertyFieldTranslate(uDirection, uNumber, [1,2]);
       end;
  end;
icFaxOut:
  begin
       case uFieldNo of
       1: result := PropertyFieldTranslate(uDirection, uNumber, [0,1,2,3,4]);
       end;
  end;
icTTGetLetter:
  begin
       case uFieldNo of
       1: result := PropertyFieldTranslate(uDirection, uNumber, [1,2,3]);
       2: result := PropertyFieldTranslate(uDirection, uNumber, [1,2]);
       end;
  end;
icFileCopy:
  begin
       case uFieldNo of
       1: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
       end;
  end;
icVoxPlay:
  begin
       case uFieldNo of
       1: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
       2: result := PropertyFieldTranslate(uDirection, uNumber, vfaFIELDS);
       4: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
       end;
  end;
icVoxSpeak:
  begin
       case uFieldNo of
       1: result := PropertyFieldTranslate(uDirection, uNumber, [0,1,2,7,8,3,6,4,5]);
```

```
        2: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
        end;
  end;
icVoxRec:
  begin
        case uFieldNo of
        1: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
        2: result := PropertyFieldTranslate(uDirection, uNumber, [1,0]);   //[0,1]
        3: result := PropertyFieldTranslate(uDirection, uNumber, vfaFIELDS);
        4: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
        5: result := PropertyFieldTranslate(uDirection, uNumber, [1,0]);
        end;
  end;
icDictate:
  begin
        case uFieldNo of
        1: result := PropertyFieldTranslate(uDirection, uNumber, [0,1,2,3,4]);
        2: result := PropertyFieldTranslate(uDirection, uNumber, [1,0]);   //[0,1]
        3: result := PropertyFieldTranslate(uDirection, uNumber, vfaFIELDS);
        4: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
        5: result := PropertyFieldTranslate(uDirection, uNumber, [1,0]);
        6: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
        7: result := PropertyFieldTranslate(uDirection, uNumber, [0,1,2,3,4]);
        end;
  end;
icVoxBeep:
  begin
        case uFieldNo of
        1: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
        end;
  end;
icVoxVolume:
  begin
        case uFieldNo of
        1: result := PropertyFieldTranslate(uDirection, uNumber, [3,2,1,0,-1,-2,-3]);
        end;
  end;
icTextSpeech:
  begin
        case uFieldNo of
        1: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
        2: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
        end;
  end;
icTrigRecord:
  begin
        case uFieldNo of
        1: result := PropertyFieldTranslate(uDirection, uNumber, vfaFIELDS);
        2: result := PropertyFieldTranslate(uDirection, uNumber, [1,0]);
        3: result := PropertyFieldTranslate(uDirection, uNumber, [1,0]);
        4: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
        end;
  end;
icStreaming:
  begin
        case uFieldNo of
        1: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
        end;
  end;
icConfOpen:
  begin
        case uFieldNo of
        1: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
        2: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
        3: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
        4: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
        end;
  end;
icConfJoin:
  begin
        case uFieldNo of
        1: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
        end;
  end;
icVarMath:
```

```
  begin
       case uFieldNo of
       1: result := PropertyFieldTranslate(uDirection, uNumber, [1,2,3,4,5,8,7,6,9]);
       end;
  end;
icVarDateInfo:
  begin
       case uFieldNo of
       1: result := PropertyFieldTranslate(uDirection, uNumber, [1,2,3,4]);
       2: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
       end;
  end;
icVarCmp:
  begin
       case uFieldNo of
       1: result := PropertyFieldTranslate(uDirection, uNumber, [1,2,3,4,5,6,7,9,8]);
       end;
  end;
icFinCalc:
  begin
       case uFieldNo of
       1: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
       end;
  end;
icSetGlobal:
  begin
       case uFieldNo of
       1: result := PropertyFieldTranslate(uDirection, uNumber, [1,2]);
       end;
  end;
icParse:
  begin
       case uFieldNo of
       1: result := PropertyFieldTranslate(uDirection, uNumber, [1,3,4,2]);
       2: result := PropertyFieldTranslate(uDirection, uNumber, [1,2,4]);
       3: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
       4: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
       end;
  end;
icSQLEndTr:
  begin
       case uFieldNo of
       1: result := PropertyFieldTranslate(uDirection, uNumber, [1,2]);
       end;
  end;
icMessage:
  begin
       case uFieldNo of
       1: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
       2: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
       end;
  end;
// Telos
icDigRelCall:
  begin
       case uFieldNo of
       1: result := PropertyFieldTranslate(uDirection, uNumber, [1,2]);
       end;
  end;
// eMail
icSendMail:
  begin
       case uFieldNo of
       1: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
       end;
  end;
// Webpage
icInternet:
  begin
       case uFieldNo of
       1: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
       end;
  end;
// XML
icXMLRequest:
```

```
  begin
       case uFieldNo of
       1: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
       end;
  end;
icXMLFetch:
  begin
       case uFieldNo of
       1: result := PropertyFieldTranslate(uDirection, uNumber, [0,1,2]);
       2: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
       end;
  end;
icXMLLoad:
  begin
       case uFieldNo of
       1: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
       end;
  end;
icXMLWrite:
  begin
       case uFieldNo of
       1: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
       end;
  end;
icSecHash:
  begin
       case uFieldNo of
       1: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
       end;
  end;
icSecEncrypt:
  begin
       case uFieldNo of
       1: result := PropertyFieldTranslate(uDirection, uNumber, [0,1]);
       end;
  end;
```

TAM RECORD

The following pascal code illustrates how to build a single record

```
// ************************************************************
// *** Build a TAM record to save                           ***
// ************************************************************
function TEngFlow.FCMODBuildRec(ArrayEntry: Integer): String;  // ArrayEntry
is a record number
const
     // Characters
     TTChrSep = #1;   // Field separator
     TTChrRec = #2;   // Record separator
var
   mIFunction: Integer;
begin
   // Build TAM record
   result := // Field 1
             IntToStr(TTEntryArr[ArrayEntry,itFunction])+
             TTChrSep+
             // Field 2
             FCGetField(TTEntryRec[ArrayEntry],
                        FCLocateField(TTEntryRec[ArrayEntry],
                                      FFTAMEntryLabel, eFTAM))+
             TTChrSep+
             // Field 3
             IntToStr(TTEntryArr[ArrayEntry,itCurrent])+
             TTChrSep+
             // Field 4
             IntToStr(TTEntryArr[ArrayEntry,itNext])+
             TTChrSep+
             // Field 5
             IntToStr(TTEntryArr[ArrayEntry,itParent])+
             TTChrSep+
             // Field 6
             FCGetField(TTEntryRec[ArrayEntry],
                        FCLocateField(TTEntryRec[ArrayEntry],
                                      FFTAMEntryN01, eFTAM))+
             TTChrSep+
             // Field 7
             FCGetField(TTEntryRec[ArrayEntry],
                        FCLocateField(TTEntryRec[ArrayEntry],
                                      FFTAMEntryN02, eFTAM))+
             TTChrSep+
             // Field 8
             FCGetField(TTEntryRec[ArrayEntry],
                        FCLocateField(TTEntryRec[ArrayEntry],
                                      FFTAMEntryN03, eFTAM))+
             TTChrSep+
             // Field 9
             FCGetField(TTEntryRec[ArrayEntry],
                        FCLocateField(TTEntryRec[ArrayEntry],
                                      FFTAMEntryN04, eFTAM))+
             TTChrSep+
             // Field 10
             FCGetField(TTEntryRec[ArrayEntry],
```

```
            FCLocateField(TTEntryRec[ArrayEntry],
                        FFTAMEntryN05, eFTAM))+
TTChrSep+
// Field 11
FCGetField(TTEntryRec[ArrayEntry],
            FCLocateField(TTEntryRec[ArrayEntry],
                        FFTAMEntryN06, eFTAM))+
TTChrSep+
// Field 12
FCGetField(TTEntryRec[ArrayEntry],
            FCLocateField(TTEntryRec[ArrayEntry],
                        FFTAMEntryN07, eFTAM))+
TTChrSep+
// Field 13
FCGetField(TTEntryRec[ArrayEntry],
            FCLocateField(TTEntryRec[ArrayEntry],
                        FFTAMEntryN08, eFTAM))+
TTChrSep+
// Field 14
FCGetField(TTEntryRec[ArrayEntry],
            FCLocateField(TTEntryRec[ArrayEntry],
                        FFTAMEntryS01, eFTAM))+
TTChrSep+
// Field 15
FCGetField(TTEntryRec[ArrayEntry],
            FCLocateField(TTEntryRec[ArrayEntry],
                        FFTAMEntryS02, eFTAM))+
TTChrSep+
// Field 16
FCGetField(TTEntryRec[ArrayEntry],
            FCLocateField(TTEntryRec[ArrayEntry],
                        FFTAMEntryS03, eFTAM))+
TTChrSep+
// Field 17
FCGetField(TTEntryRec[ArrayEntry],
            FCLocateField(TTEntryRec[ArrayEntry],
                        FFTAMEntryS04, eFTAM))+
TTChrSep+
// Field 18
FCGetField(TTEntryRec[ArrayEntry],
            FCLocateField(TTEntryRec[ArrayEntry],
                        FFTAMEntryS05, eFTAM))+
TTChrSep+
// Field 19
FCGetField(TTEntryRec[ArrayEntry],
            FCLocateField(TTEntryRec[ArrayEntry],
                        FFTAMEntryS06, eFTAM))+
TTChrSep+
// Field 20
FCGetField(TTEntryRec[ArrayEntry],
            FCLocateField(TTEntryRec[ArrayEntry],
                        FFTAMEntryS07, eFTAM))+
TTChrSep+
// Field 21
FCGetField(TTEntryRec[ArrayEntry],
            FCLocateField(TTEntryRec[ArrayEntry],
                        FFTAMEntryS08, eFTAM))+
TTChrSep+
// Field 22
FCGetField(TTEntryRec[ArrayEntry],
```

```
                    FCLocateField(TTEntryRec[ArrayEntry],
                                FFTAMEntryS09, eFTAM))+
            TTChrSep+
            // Field 23
            FCGetField(TTEntryRec[ArrayEntry],
                    FCLocateField(TTEntryRec[ArrayEntry],
                                FFTAMEntryS10, eFTAM))+
            TTChrSep+
            // Field 24
            IntToStr(TTEntryArr[ArrayEntry,itAttributes])+
            TTChrSep+
            // Field 25
            FCGetField(TTEntryRec[ArrayEntry],
                    FCLocateField(TTEntryRec[ArrayEntry],
                                FFTAMEntryNotes, eFTAM))+
            TTChrSep+
            // Field 26
            IntToStr(TTEntryArr[ArrayEntry,itX])+
            TTChrSep+
            // Field 27
            IntToStr(TTEntryArr[ArrayEntry,itY])+
            TTChrSep+
            // Field 28
            IntToStr(TTEntryArr[ArrayEntry,itTackX])+
            TTChrSep+
            // Field 29
            IntToStr(TTEntryArr[ArrayEntry,itTackY])+
            TTChrSep+
            // Field 30
            IntToStr(TTEntryArr[ArrayEntry,itConnectShape])+
            TTChrSep+
            // Field 31
            IntToStr(TTEntryArr[ArrayEntry,itParentConnect])+
            TTChrSep+
            // Record Separator
            TTChrRec;
end;  // Build record
```

TAM DUMP

The following dump shows the construction of a very simple TAM file. The yellow indicates various sections of header section, and green the records section. Blue characters are End of Records, and purple are End of Files.

```
54 65 6C 65 46 6C 6F 77-0D 0A 43 6F 70 79 72 69   TeleFlow..Copyri
67 68 74 20 31 39 39 34-2D 32 30 30 33 20 65 6E   ght 1994-2003 en
47 65 6E 69 63 20 43 6F-72 70 6F 72 61 74 69 6F   Genic Corporatio
6E 2E 20 41 6C 6C 20 72-69 67 68 74 73 20 72 65   n. All rights re
73 65 72 76 65 64 2E 0D-0A 45 4E 47 54 46 40 54   served...ENGTF@T
41 4D 30 32 30 30 20 54-65 6C 65 46 6C 6F 77 20   AM0200 TeleFlow
41 70 70 6C 69 63 61 74-69 6F 6E 20 28 6D 6F 64   Application (mod
75 6C 65 29 0D 0A 02 0D-0A 44 3A 5C 54 65 6C 65   ule)..☻..D:\Tele
46 6C 6F 77 5C 54 65 73-74 20 41 70 70 73 5C 54   Flow\Test Apps\T
61 6D 5C 54 46 54 61 6D-2E 54 61 6D 0D 0A 1A 45   am\TFTam.Tam...E
4E 47 54 46 40 54 41 4D-30 32 30 30 30 30 30 30   NGTF@TAM02000000
30 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20   0
20 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20-20 20 20 20 20 1A 33 30                .30
01 01 31 01 33 01 30 01-30 01 30 01 30 01 30 01   ..1.3.0.0.0.0.0.
30 01 30 01 30 01 30 01-01 01 01 01 01 01 01 01   0.0.0.0........
01 01 30 01 49 6E 66 6F-72 6D 61 74 69 6F 6E 20   ..0.Information
61 62 6F 75 74 20 54 61-6D 01 34 30 30 01 33 30   about Tam.400.30
30 01 31 33 30 30 01 35-35 30 01 31 01 30 01 02   0.1300.550.1.0.☻
34 30 01 01 32 01 2D 34-01 30 01 30 01 30 01 30   40..2.-4.0.0.0.0
01 30 01 30 01 30 01 30-01 30 01 01 01 01 01 01   .0.0.0.0.0.....
01 01 01 01 01 30 01 01-34 30 30 01 31 31 30 30   .....0..400.1100
01 30 01 30 01 31 01 30-01 02 31 30 30 01 01 33   .0.0.1.0.☻100..3
01 32 01 30 01 31 01 30-01 30 01 30 01 30 01 30   .2.0.1.0.0.0.0.0
01 30 01 30 01 01 01 01-01 01 63 3A 5C 74 65 73   .0.0......c:\tes
74 2E 77 61 76 01 01 01-01 01 30 01 01 34 30 30   t.wav.....0..400
01 37 30 30 01 31 31 30-30 01 31 30 30 30 01 31   .700.1100.1000.1
01 30 01 02 39 33 30 01-01 34 01 2D 35 01 33 01   .0.☻930..4.-5.3.
30 01 30 01 30 01 30 01-30 01 30 01 30 01 30 01   0.0.0.0.0.0.0.0.
01 01 01 01 01 01 01 01-01 01 2D 34 01 01 36 30   ..........-4..60
30 01 37 30 30 01 30 01-30 01 31 01 31 01 02      0.700.0.0.1.1.☻
```

The first “02” indicates the end of the Intro section. The First “1A” (#26) indicates the end of the Notes section. The Second “1A” indicates the end of the Internal section. The following blocks consists of four records as follows.

STEP FUNCTION DEFINITIONS

This is a list of the Step Functions that are used by TeleFlow. This list may not be up to date as new steps are added during the on going development of the product. The bolded steps indicate basic IVR function steps. In the TeleFlow TAM file, leading zeros are removed.

The following list provides constants whose naming conventions are not necessarily indicative of their true functions. These constants are apt to change. Please refer to the third column for a description of their proper function. For naming, internal constants, we recommend “stepWaitForCall” etc

Parent Steps:

icAnswerPhone: 0010 'Wait for Call'
icPickUpLine: 0011 'Pick Up'
icPlaceCall: 0015 'Place Call'
icLineFlash: 0017 'Flash Hook'
icLineTone: 0018 'Send TT'
icHangup: 0020 'Hang Up'
icBlockCall: 0021 'Block Port'
icUnBlockCall: 0022 'Unblock Port'
icSwitchCall: 0025 'Switch Call'
icSwitchCallDis: 0026 'Switch Disconnect'
icProcStart: 0030 'START'
icProcEnd: 0040 'FINISH'
icTTGetKey: 0050 'Get TT'
icTTGetString: 0060 'Get TT String'
icTTGetLetter: 0063 'Get Letter'
icTTClearBuf: 0065 'Clear TT'
icTTPending: 0075 'Pending TT'
icLabel: 0080 'Go to Label'
icFatal: 0081 'Fatal Stop'
icPoint: 0082 'Connector Point'
icUnknown: 0083 'Reminder'
icComment: 0085 'Comment'
icWait: 0090 'Wait'
icMessage: 0095 'Write Text'
icVoxPlay: 0100 'Play'
icSetVoxVar: 0105 'Set Play Variable'
icVoxBeep: 0110 'Beep'
icVoxSpeak: 0115 'Say'
icVoxRec: 0120 'Record'
icDictate: 0121 'Dictate'
icFileDelete: 0130 'Delete File'
icVoxVolume: 0140 'Set Volume'

```
icFileCopy:         0150 'Copy File'
icTextSpeech:       0180 'Speak Text'
icTrigListen:       0190 'Call Trigger'
icTrigRecord:       0195 'Start Record'
icTrigRecStop:      0196 'Stop Record'
icStreaming:        0197 'Streaming'
icConfOpen:         0200 'Open Conference'
icConfClose:        0201 'Close Conference'
icConfJoin:         0202 'Join Conference'
icConfLeave:        0203 'Leave Conference'
icReloadOn:         0220 'Reload On'
icReloadOff:        0230 'Reload Off'
icVarControl:       0290 'Var Control'
icVarSet:           0300 'Set Global Variable'
icVarEdit:          0301 'Edit Var'
icVarRandom:        0302 'Random Number'
icVarSetPriv:       0305 'Set Local Variable'
icVarCmp:           0310 'Compare'
icVarMath:          0320 'Math'
icVarDateInfo:      0324 'Date Info'
icFormat:           0340 'Format'
icFinCalc:          0325 'Financial Calc'
icCreateDir:        0350 'Create Folder'
icScanDir:          0355 'Scan Folder'
icFileOpen:         0360 'Load Text'
icParse:            0380 'Parse'
icSendMail:         0385 'Send E-Mail'
icCreditTrans:      0386 'Credit Transaction'
icInternet:         0390 'Load Web Doc'
// VR
icXMLRequest:       0391 'Simpull Request'
icXMLDiscon:        0392 'XML End'
icXMLFetch:         0393 'XML Fetch'
icXMLLoad:          0394 'XML Load'
icXMLWrite:         0395 'XML Write'
icXMLParams:        0396 'Simpull Parameters'
// VR
icVocRecInit:       0400 'VR Acquire'
icVocRecRelease:    0410 'VR Release'
icVocRecGetDigits:  0420 'VR Get Digits'
icVocRecGetYN:      0430 'VR Get Yes/No'
icVocRecGetCmd:     0440 'VR Get String'
// Application
icAppCall:          0500 'Run Flowchart'
icAppRunTap:        0505 'Run TAP'
icAppGoto:          0510 'Go to Label'
icAppExe:           0520 'Run Program'
icAppDllLink:       0521 'Link DLL'
icAppDllUnlink:     0522 'Unlink DLL'
icAppDllCall:       0523 'DLL Function'
```

```
icComCall:          0525 'Run COM Object'
icComProcOn:        0526 'Process COM On'
icComProcOff:       0527 'Process COM Off'
icComProcChanges:   0528 'Process Changes'

// Security
icSecHash:          0540 'Hash'
icSecEncrypt:       0541 'Encrypt'
icSecDecrypt:       0542 'Decrypt'
// Telos (custom solution) customization 83567
icDigAcqLine:       0550 'Acquire Port'
icDigRelLine:       0555 'Release Port'
icDigReqCall:       0560 'Request Call'
icDigAnsCall:       0565 'Answer Call'
icDigAcptCall:      0567 'Accept Call'
icDigCanReq:        0570 'Cancel Call'
icDigRejCall:       0575 'Reject Call'
icDigRelCall:       0580 'Release Call'
// SQL
icFaxOut:           0600 'Fax Send'
icFaxIn:            0610 'Fax Receive'
icFaxQueueAdd:      0620 'Add to Fax'
icFaxQueueClr:      0630 'Fax Clear'
icSetGlobal:        0700 'Global Event'
icSetLanuage:       0705 'Set Language'
icSQLConnect:       0800 'DB Connect'
icSQLDiscon:        0805 'DB Disconnect'
icSQLBeginSt:       0810 'SQL Statement'
icSQLLock:          0815 'DB Lock'
icSQLBeginTr:       0820 'SQL Begin Transaction'
icSQLEndTr:         0825 'SQL End Transaction'
icSQLEndSt:         0830 'SQL End'
icSQLFetch:         0835 'SQL Fetch'
```

Child (Action) Steps:

```
icFirstAction   // Action (Child Steps) follow 0880

icTTOther           0890 'TT Others'
icTT0               0900 'TT 0'
icTT1               0901 'TT 1'
icTT2               0902 'TT 2'
icTT3               0903 'TT 3'
icTT4               0904 'TT 4'
icTT5               0905 'TT 5'
icTT6               0906 'TT 6'
icTT7               0907 'TT 7'
icTT8               0908 'TT 8'
icTT9               0909 'TT 9'
```

```
icTTStar            0910 'TT Star'
icTTPound           0911 'TT Pound'
icTTA               0915 'TT A'
icTTB               0916 'TT B'
icTTC               0917 'TT C'
icTTD               0918 'TT D'
icTimeOut           0920 'Time Out'
icSilenceWait       0921 'Silence Wait'
icSilenceTimeout    0922 'Silence Timeout'
icThreshold         0925 'Threshold'
icFail              0930 'Failure'
icBusy              0935 'Busy'
icCheck             0940 'Check'
icEx                0950 'Ex'
icYes               0955 'Yes'
icNo                0956 'No'
icActLabel          0960 'Label'
icActComment        0970 'Notes'
icActPoint          0980 'Point'
icTest              0999 'Test'
```

ACTION (CHILD) STEPS REFERENCE

This section describes what child steps, and the order that they must be created in following the Parent step. Action steps are defined identical to the Parent step, but must also include the reference back to the Entry number representing the Parent, and never have properties assigned (N01-N08,S01-S10). TeleFlow convention requires that Action steps are created in order immediately behind the Parent step.

In the list, the first step is the Parent step, and the following indented steps are the Child steps that are created in the order shown.

```
icAnswerPhone
     icTimeOut

icPickUpLine
     icTimeOut

icPlaceCall
     icTimeOut
     icBusy
     icFail

icBlockCall
     icFail

icFaxOut
     icTimeOut
     icBusy
     icFail

icTTGetString
     icTimeOut

icTTGetLetter
     icTimeOut
     icTTStar
     icTTPound

icFileDelete
     icFail

icFileCopy
     icFail

icVoxPlay
```

```
     icFail

icSetVoxVar
     icFail

icVoxRec
     icTimeOut
     icFail
     icSilenceWait

icDictate
     icTimeOut
     icFail
     icSilenceTimeout

icCreateDir
     icFail

icScanDir
     icFail

icTextSpeech
     icFail

icTrigListen
     icTimeOut

icTrigRecord
     icFail

icTrigRecStop
     icFail

icConfOpen
     icFail

icConfJoin
     icFail

icVarControl
     icFail

icVarSet
     icFail

icVarRandom
     icFail

icVarCmp
     icCheck
```

```
     icEx

icVarSetPriv
     icFail

icParse
     icFail
     icCheck
     icEx

icFileOpen
     icFail

icSendMail
     icFail

icCreditTrans
     icFail
     icYes
     icNo

icInternet
     icTimeOut
     icFail

icAppDllLink
     icFail


icAppDllUnlink
     icFail

icAppDllCall

     icFail
icComCall
     icFail

icSecHash
     icFail

icSecEncrypt
     icFail

icSecDecrypt
     icFail

icSQLConnect
     icFail
```

```
icSQLBeginSt
     icFail

icSQLLock
     icFail

icSQLFetch
    icFail

icVocRecInit
    icFail

icVocRecGetDigits
    icFail
    icThreshold

icVocRecGetYN
    icFail
    icThreshold
    icYes
    icNo

icVocRecGetCmd
    icFail
    icThreshold

icTTGetKey
    icTimeOut
    icTTOther
    icTT1
    icTT2
    icTT3
    icTTA
    icTT4
    icTT5
    icTT6
    icTTB
    icTT7
    icTT8
    icTT9
    icTTC
    icTTStar
    icTT0
    icTTPound
    icTTD

icXMLRequest
    icTimeOut
    icFail
```

```
icXMLFetch
    icFail

icXMLLoad
    icFail
```

PUTTING A TAM TOGETHER

This section describes the steps to programmatically put together a basic TAM file by referencing this document. The fastest approach to creating the header section would to be populate a string directly with the following characters:

```
54 65 6C 65 46 6C 6F 77-0D 0A 43 6F 70 79 72 69   TeleFlow..Copyri
67 68 74 20 31 39 39 34-2D 32 30 30 33 20 65 6E   ght 1994-2003 en
47 65 6E 69 63 20 43 6F-72 70 6F 72 61 74 69 6F   Genic Corporatio
6E 2E 20 41 6C 6C 20 72-69 67 68 74 73 20 72 65   n. All rights re
73 65 72 76 65 64 2E 0D-0A 45 4E 47 54 46 40 54   served...ENGTF@T
41 4D 30 32 30 30 20 54-65 6C 65 46 6C 6F 77 20   AM0200 TeleFlow
41 70 70 6C 69 63 61 74-69 6F 6E 20 28 6D 6F 64   Application (mod
75 6C 65 29 0D 0A 02 0D-0A 20 20 20 20 20 20 20   ule)..☻..
20 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20-20 20 20 20 0D 0A 1A 45               ...E
4E 47 54 46 40 54 41 4D-30 32 30 30 30 30 30 30   NGTF@TAM02000000
30 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20   0
20 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20-20 20 20 20 1A
```

The section between the “02 0D 0A” and “0D 0A 1A” is variable and could be reduced to blank. Server does not require information in this section.

The rest of the TAM File consists of Step records each terminating with a “02” character. The construction is as follows:

1) Start step
2) 1 step
   + mandatory action (child) steps
3) 2
   + mandatory action (child) steps
4) 3
   + mandatory action (child) steps
   .
   .
   .
5) n
   + mandatory action (child) steps
6) End step

Start step record

```
33 30 01 01 31 01 33 01 30 01 30 01 30 01 30 01    30..1.3.0.0.0.0.
30 01 30 01 30 01 30 01 30 01 01 01 01 01 01 01    0.0.0.0.0.......
01 01 01 01 30 01 01 30 01 30 01 30 30 01 30 01    ....0..0.0.0.0.0
01 30 01 02                                        .0.☻

3 = Entry number of next step
```

The Start step function number is 30, and Entry (third field) is 1 (the serial number or primary key of the step)

End step record

```
34 30 01 01 32 01 2D 34 01 30 01 30 01 30 01 30    40..2.-4.0.0.0.0
01 30 01 30 01 30 01 30 01 30 01 01 01 01 01 01    .0.0.0.0.0......
01 01 01 01 01 30 01 01 30 01 30 01 30 01 30 01    .....0..0.0,0.0.
30 01 30 01 02                                     0.0.☻

-4 = Indicates end of flowchart.  Will not have a next step
```

The Start step function number is 40, and Entry (third field) is 2 (the serial number or primary key of the step). The Next (fourth field) of the last step to run in the flowchart, must have an entry of 2, which points to this step, which indicates the end of the flowchart.

All other records

Every addition record must then be added after the Start and End records. Records are formatted in the same way as the start record, and additional Child steps are created when required (refer to the Action (Child) Steps Reference section)

The steps for creating the next records are as follows:

1) Increment the variable that defines the Entry (third field) by one.
2) Decide what the next step Function (first field) will be.
3) Account for any action (child) steps that that will need to be created
4) Take the number of action steps, and at it to the current Entry variable to define the following step that might be pointed in the Next step. You will need to consider this as you develop the flowchart.

5) Action steps are generally hidden by the Designer if they are Failure (bomb) actions. For the most part, you will not need to worry about many actions, and failures will be picked up and dealt with by the Server.
6) Populate the properties (fields) with the appropriate values. You will need to use the Designer Property sheet and F2 button to determine what values will need to be populated. Also refer to the Step Properties section for N (or U) values which are hidden from the user.
7) Continue to create any Action steps that are necessary. Refer to the Action (Child) Steps Reference section for the list. Remember to increment the Entry number and point back to the parent's Entry number in the Parent Entry Pointer (field five).
8) Repeat until the flowchart is created
9) Remember to point back to the second record (End step) for all end paths

Because it is hard to anticipate the step numbers, it may be ideal to create an array of steps and then populate the numbers after the fact. This array could then be easily looped and saved with alternating field, end of field ("01"), and eventually end of record values ("02").

GLOSSARY

IVR: Interactive Voice Response system. The culmination of computer hardware, voice hardware and software programming to automate a telephone conversation and provide information or specific functionality.

PROVIDED BY

This document and information provided within has been provided by both the company as listed on the title page as well as:

enGenic Corporation
618 – 1111 Melville St
Vancouver, BC, V6E 3V6

T: 604.639.6391 x 101
F: 604.639.6392

Document: TeleFlow TAM Layout.doc
Printed: Oct. 17, 03

 End.