User Instructions

Ebase Data Transfer Tool, V1.1
(FM5, Ebase V1.0.3 Version)
Larry Bednar
April 22, 2005

Purpose

This document explains the use of the “Ebase data transfer tool”.

Overview

The “Ebase data transfer tool” (EDTT) is designed to assist users in “migration” of data from a “old” copy of Ebase V1.0.x database files to a “new” copy of “standard” Ebase V1.0.3 database files. This process is typically undertaken to move data from a damaged set of Ebase files into a fresh, undamaged set of files after the original FileMaker databases become “corrupted” during a system crash, etc.

If this data migration is performed manually, a very lengthy series of manual steps (button clicks, menu selections, responses to dialog windows, etc) is required to complete the job correctly. Because of the large number of steps, even an expert user can easily overlook some essential task. The process is thus inherently error-prone.

Many of the steps required for “data migration” are easily scripted by FileMaker, and are therefore very amenable to automation. Automation of these steps results in a substantially reduced chance of “operator error” during data migration. Automated steps, once set up correctly, are much more likely to be implemented correctly than steps in a “manual” migration.

EDTT also provides a checklist through which a user can easily keep track of their progress through the data migration process. If the work session is interrupted, the user can much more easily return and carry out the correct steps in sequence to complete the process already started. Multiple data migrations may be tracked in the database, each one represented by a separate record in the provided FileMaker database provided to run the application.

The version of the EDTT documented here is designed to transfer data from damaged Ebase V1.0.x files to a clean set of Ebase V1.0.3 database files. Because EDTT requires 3 “non-standard” Ebase scripts in each new V1.0.3 file targeted to accept data, copies of Ebase V1.0.3 files modified to include those scripts are provided with EDTT.

Requirements

The following requirements should assure that the EDTT application works correctly:

1. A Windows-based PC must be used. (see comments below)
2. A copy of FileMaker Pro V5 or V6 must be installed on the PC where the data transfer is being attempted. (see comments below)
3. The data to be transferred to new databases must be in Ebase V1.0.x database files

At present, EDTT has worked correctly for the author in repeated and varied tests within a computing environment such as that described above.

Several users have reported that EDTT functions properly under the Macintosh operating system. However, this hasn't been tested by the author directly. Please note that commonly used Macintosh archiving software (e.g. "Stuffit",…) is capable of unloading the WinZip file used to provide EDTT.

Optional

A Perl interpreter (such as ActiveState Perl, etc.) must be installed on the computer where the EDTT is being used if the optional Perl utility programs provided with EDTT are to be used.

Components

EDTT is provided in a WinZip self-extracting archive. This file contains:

1. A FileMaker database ("edtt_data_transfer_scripts.fp5") containing the checklist of data migration tasks and scripts designed to automate data migration tasks.
2. A FileMaker database ("edtt_custom_objects.fp5") containing scripts designed to describe fields, layouts, and scripts present in the "source" or "old" ebase files, but not present in the "target" or "new" ebase files.
3. A set of Ebase V1.0.3 files that have been slightly modified. Most of these files contain three "data import" scripts designed to work in coordination with scripts in the "edtt_data_transfer_scripts.fp5" file.
4. A few Perl utility programs that may be used by experts to facilitate the data transfer process on computers with a Perl interpreter installed. Users who are unfamiliar with Perl have no reason for concern – use of these programs is not mandatory.
5. A PDF file named "user_instructions.pdf" (this document).

Installation

To install and use the tool, the following steps are followed:

1. The user receives a self-extracting WinZip archive file that contains the EDTT application.
2. The user places the EDTT archive file in the parent folder of the folder that currently holds the Ebase database files.
3. The user extracts the files from the archive. The extraction should be done in a fashion that recreates paths/subfolders from the archive file. This creates two new subfolders within the "parent" folder. The new subfolders are named "edtt_new" and "edtt_old".

   "EDTT_new" subfolder

   This folder contains:

   - modified versions of Ebase V1.0.3 database files.

These modified files are designed specifically to work with EDTT. These files serve as the data migration "target" – the files into which your data will be moved.

NOTE (ebase files modified to work with EDTT): These modified versions of the Ebase database files *must* be used with EDTT – ebase database files that do not contain the specific modifications needed by EDTT will not work correctly as data "targets" for EDTT.

- The "**edtt_data_transfer_scripts.fp5**" file.

  This FileMaker database contains:

  - master scripts that run data imports and check for scripts,
  - fields and layouts in the old copy of Ebase, but not in the new copy of ebase files to which data will be transferred.
  - FileMaker layouts that provide a checklist of data migration steps to be followed by the user. These checklists can be used to track progress through one or more data migration processes.

- The "**edtt_custom_objects.fp5**" file.

  This FileMaker database contains:

  - Scripts for checking source databases for fields, scripts and layouts that are not present in the target databases. After these scripts are run, the **edtt_custom_objects.fp5** database contains a record of each these "orphan objects". These fields, scripts, and layouts must be manually recreated in the target databases if they are needed.

- Perl scripts that streamline some steps in the data migration process.

  Users unfamiliar with Perl may safely ignore these programs completely. Those with expertise in Perl and with Perl interpreters installed on their systems may want to make use of these scripts, however.

"EDTT_old" subfolder

This folder contains:

- A small text file named "placeholder.txt" which serves only to ensure that the "edtt_old" subfolder is created when the archive file is unloaded. .

Instructions for Use

1. Place copies of the current Ebase databases in the "edtt_old" folder.

   EDTT uses these files as the "source" for data to be transferred into the "target" databases located in the "edtt_new" folder.

   During operation, EDTT utilizes only databases in the "edtt_old" and "edtt_new" folders. As a result, there should be no danger whatsoever to your original database files.

NOTE (perl utility scripts): If the PC has a copy of Perl installed, the included script "make_ebase_copies.pl" can be run to perform this task. This script will prompt the user for the name of the folder containing the current Ebase database files. The script will determine whether Ebase V1.0.0, 1.0.2 or 1.0.3 is installed in that folder and will move files carrying the standard Ebase extensions, "100", "102", or "103". (This approach will usually only be wise for an expert user with a knowledge of Perl.)

NOTE (archiving original copy of ebase): Optionally, if a copy of an archiving program is available, it may be prudent to place your original working copy of ebase into an archive file (*.zip, etc.) after performing this step. Once you confirm that the archive has correctly stored your database files, delete the uncompressed copies. In some situations, FileMaker may look for copies of FileMaker databases in the last location where they were found previously. For the purposes of data transfer, the ideal is that the only copies of ebase available to FileMaker are those found in the "edtt_old" and "edtt_new" folders. This step is recommended as a safeguard, but EDTT will typically perform correctly regardless of whether this optional step is taken.

2. The "master" password to the old (i.e. "source") and new (i.e. "target") ebase databases should be set to the same value.

   EDTT automatically uses the password "master" when attempting to open ebase files. To avoid a need to type passwords repeatedly as files are opened by EDTT, the "old" copy of ebase needs to be set to use the value "master" for the most highly privileged password (this is the default ebase setting upon delivery).

   NOTE (password changes): Instructions for changing the ebase passwords may be found in the ebase V1 Administrator's manual. **If you do not have a copy of the Administrator's manual or are not confident of your ability to change the passwords, it is doubtful that you should be undertaking a data migration process without the help of a expert – even with the assistance of EDTT.**

3. Close all ebase databases – both "old" and "new".

   In some cases, database corruption in the "old" database may result in a situation where the user must close FileMaker completely to accomplish this.

4. Open the **edtt_data_transfer_scripts** database in FileMaker

5. In the **edtt_data_transfer_script** database, select the layout appropriate for the "old" version of ebase being used as a data "source".

   A different layout is provided for each of the three versions of ebase that might be used as a data "source". The available layouts are:

   "V1.0.0 to V1.0.3 transfer steps"

   "V1.0.2 to V1.0.3 transfer steps"

   "V.1.03 to V1.0.3 transfer steps"

   Each layout provides a shorthand description of steps to be followed to complete a data transfer using EDTT, with "check boxes" to help track progress through those tasks.

NOTE (layout selection): Each of these layouts runs a different set of scripts, so it is important to select the correct layout. (See Fig 1 for representation of "V.1.0.3 to V1.0.3 transfer steps" layout.)

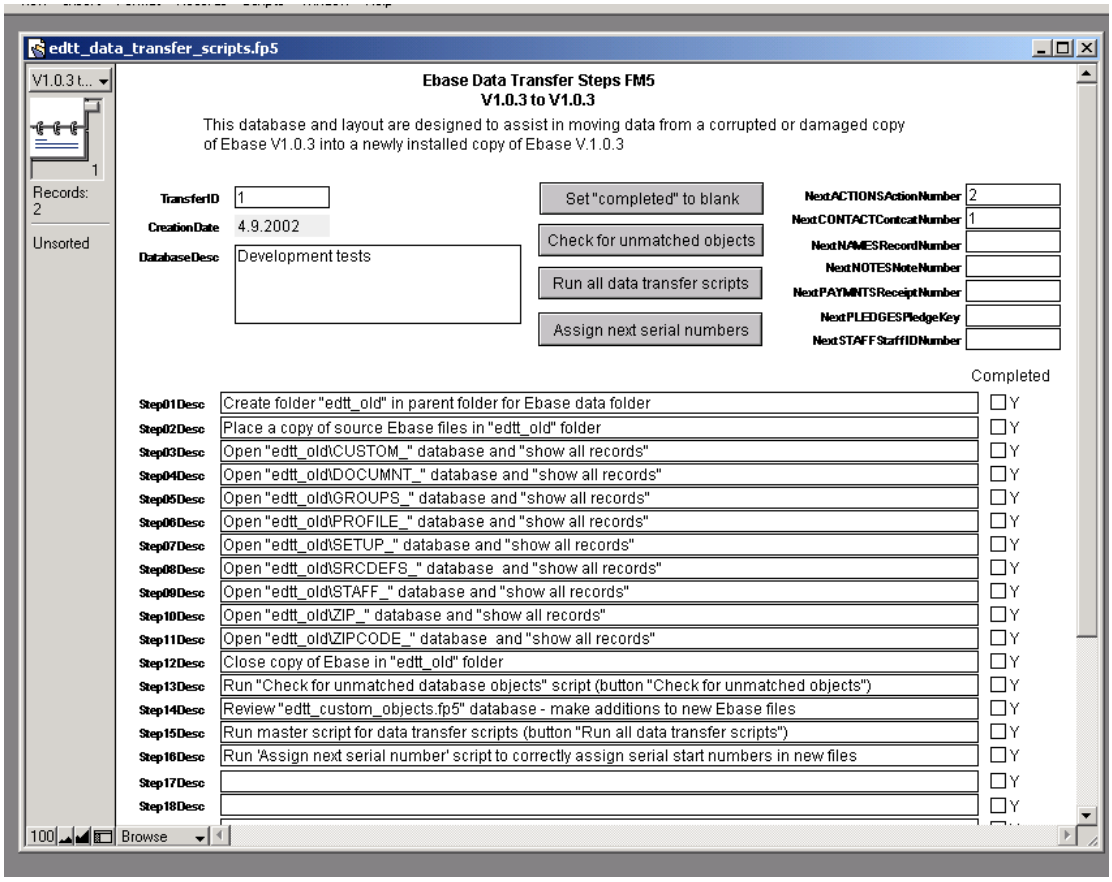

Figure 1. "Data transfer" layout in "edtt_data_transfer_scripts.fp5" database

6. Using the **Ebase data transfer steps** layout, create a new record representing a new data transfer attempt.

7. Using the **Ebase data transfer steps** layout perform all steps *up to, but not including* the **"Run 'Check for unmatched database objects'.."** step.

   NOTE (folder creation/source file copying): The "Ebase data transfer steps" layouts direct the user to "create the edtt_old folder" and "place a copy of source ebase files in 'edtt_old' folder" which will already have been completed by someone following the "user instructions" document. There is no need to repeat these steps – the layout repeats these directions primarily to *ensure* that these steps have in fact been completed by individuals who are not following the full "user instructions".

   NOTE (manual "find all" steps): Several steps on the **Ebase data transfer steps** layout direct the user to open specific ebase database files from the "old" copy of ebase, and perform a "find all" operation. This is done to ensure that all data in the file is transferred to the "new" copy of ebase. Without this step, only records in the current "found set" of the "old" database will be imported into the corresponding "new" database. If the current

"found set" in these databases is not "all records", many records may unintentionally be excluded from the data transfer operation. Most ebase database files contain a "find all" script that EDTT uses when available. For those databases that do *not* include a "find all" script as a part of the standard ebase distribution, the user must perform these steps manually.

After completing each task, the user "checks off" the task, so the database provides a record of which steps have been completed. As a result, the process can be easily continued after an interruption.

8. Perform the **"Run 'Check for unmatched database objects'…"** step listed on the "Ebase data transfer steps" layout.

   This script checks for fields, layouts, relations, scripts, and value lists that are present in the "old" ebase databases, but not in the corresponding "new" ebase databases. The results are stored in the **edtt_custom_objects.fp5** database.

   The user will be prompted for Ebase passwords as EDTT scripts open "old" and "new" Ebase database files. In each case, the user should enter the Ebase "master" password when prompted for a password.

   NOTE (unmatched db objects checking): This script lists fields, layouts, relations, scripts, and value lists that do not have correspondingly named objects in the matching "new" database. This is based entirely on the *name* of the field, layout, relation, script or value list. The script does not check for *changes* in objects. Changes in placement of fields on a layout, or display of additional fields on a layout are not detected by this check. Neither are additions or changes to script steps, changes in the "custom values" list associated with a value list, etc.

   > For instance, if the "old" PAYMNTS_.10x database includes customizations to the "Payment entry" layout (display of added fields, changes in the spatial arrangement of fields on the layout, etc.), and a the "new" PAYMNTS_.103 database includes a layout named "Payment entry", the "Payments entry" layout will *not* be identified as an unmatched object.
   >
   > However, if the old copy of the PAYMNTS_.10x database has added a "pledge_reference" field that is not present in the "new" PAYMNTS_.103 file, the "pledge_reference" field *will* be identified as an unmatched object. The results provided to the user should be a definite help, but there remains a need for the user to closely check the "old" databases for customizations that may need to be manually recreated in the "new" databases.

9. In the "new" databases, recreate any custom fields that are to be transferred from the "old" databases.

   The results of the **"Check for unmatched objects…"** step may be used to assist the user in identifying fields that should be recreated in the "new" copy of ebase.

   NOTE (field creation order): If the user recreates fields in the "new" databases in the same order they were created in the "old" databases, this can streamline later layout and script modifications. In many cases, layout elements and script steps will correctly reference custom fields ONLY if the creation order in the "new" databases matches that

in the “old” databases. Otherwise, the user may have to manually reset field references after importing scripts and layout elements from the “old” databases.

10. In the “new” databases, recreate any required value lists or value list changes.

    The results of the **“Check for unmatched objects…”** step may be used to assist the user in identifying value lists that should be newly created in the “new” copy of ebase.

    All value lists in the “old” database must be checked against corresponding value lists in the “new” databases to identify those that have been modified. Any customizations revealed during this examination must be recreated manually in the “new” databases.

    NOTE (value list creation order): If the user recreates value lists in the “new” databases in the same order they were created in the “old” databases, this can streamline later layout and script modifications. In many cases, field displays on layouts will correctly reference custom value lists ONLY if the creation order in the “new” databases matches that in the “old” databases. Otherwise, the user may have to manually reset value list references after importing layout elements from the “old” databases.

11. In the “new” databases, recreate any relations that are to be transferred from the “old” databases.

    The results of the **“Check for unmatched objects…”** step may be used to assist the user in identifying relations that should be newly created in the “new” copy of ebase.

    All relations in the “old” database should be checked against corresponding relations in the “new” databases to identify those that have been modified. Any customizations revealed during this examination must be recreated manually in the “new” databases.

    NOTE (relation creation order): If the user recreates relations in the “new” databases in the same order they were created in the “old” databases, this can streamline later layout and script modifications. In many cases, field references on layouts will correctly reference related fields ONLY if the creation order in the “new” databases matches that in the “old” databases. Otherwise, the user may have to manually reset related field references after importing layout elements or scripts from the “old” databases.

12. In the “new” databases, recreate required layouts from the “old” databases.

    All pre-existing layouts in the “new” databases must be checked closely to verify that field, script, and value list references are correct. In some cases, some of these

    The results of the **“Check for unmatched objects…”** step may be used to assist the user in identifying scripts and layouts that should be imported to the “new” copy of ebase.

    NOTE (relation/field/value list modifications before layout modifications): Field, value list, and relation modifications required in the new databases may refer to custom fields, value lists or relations. As a result, it is generally easier to postpone layout customization until all field, value list, and relation modifications have been completed.

    NOTE (data transfer without layout recreation): Strictly speaking, this step is not required to complete the “data transfer” itself – once the “new” databases contain all required fields and value lists, and value list modifications, the actual movement of data into the new databases will proceed correctly. However, if there are layout modifications to be recreated, it is usually an advantage to perform those tasks at this stage. Repeated runs of

the “check for unmatched objects…” scripts will perform more quickly if the “new” databases are empty of data.

NOTE (script reference errors in layout buttons): Any buttons that are “cut and pasted” from layouts in an “old” database to layouts in the “new” database will not refer to the correct scripts until

- script import has been completed (see step listed below)
- the user resets the script reference (see “NOTE (references to scripts in layout buttons)” below for more detailed explanation)

13. In the “new” databases, import any scripts required from the “old” databases.

    Imported scripts and layouts must be checked closely to verify that field, script, and layout references are correct.

    The results of the **“Check for unmatched objects…”** step may be used to assist the user in identifying scripts that should be imported to the “new” copy of ebase.

    NOTE (data transfer without script recreation): Strictly speaking, this step is not required to complete the “data transfer” itself – once the “new” databases contain all required fields and value lists, and value list modifications, the actual movement of data into the new databases will proceed correctly. However, if there are script modifications to be recreated, it is usually an advantage to perform those tasks at this stage. Repeated runs of the “check for unmatched objects…” scripts will perform more quickly if the “new” databases are empty.

    NOTE (references to subscripts within scripts): It is wise to double-check all scripts for subscript references to scripts newly imported from the “old” databases into a “new” database. The same explanation provided above under “NOTE (references to scripts in layout buttons)” applies to subscript references.

14. Recheck layout buttons for correct script references.

    NOTE (references to scripts in layout buttons): It is wise to double-check all buttons on layouts for script references after script import/modification has been completed. Newly added buttons that reference custom scripts may not reference the correct scripts if they are simply “cut and pasted” from layouts in the “old” database into the “new” database.

    > At a minimum, the fact that the customized ebase V1.0.3 files provided with EDTT include 3-4 scripts required by EDTT may cause “offsets” in script references. Internally, a layout button reference to a script is based on a number assigned to the script at the time of it’s creation. The script name is only a “handle” provided to allow the user to easily choose the correct script – the actual internal reference is based on the creation order of scripts in the database. If a custom script in the “old” database is script #100, the imported script in the “new” database may be script #104 due to the fact that 4 EDTT scripts have been added to the “new” database before any other modifications have been made. A layout button that runs script #100 will still attempt to start script #100 after being “cut and pasted” into a layout in the new database – and will run the wrong script until the user alters the button definition to correctly reference the correct script.

15. At the **"Review 'edtt_custom_objects.fp5' database…"** step, the user double-checks that all required fields, scripts, and layouts added to the "old" copy of ebase as customizations have been manually recreated in the "new" copy of ebase.

    After completing this review and verifying that the "target" database has all the desired customizations, the user should "check off" this task.

    NOTE (orphaned objects): Fields, layouts, relations, scripts, or value lists that are not standard features of ebase V.1.0.3 will not be present in the "new" databases unless they have been manually created there.

    NOTE (repeated use of unmatched objects check): The 'Run "Check for unmatched database objects"…' step may be rerun several times if needed to repeatedly check for unmatched objects. When the user is finally satisfied with the results they can proceed to the next step.

16. Close any "old" or "new" ebase files that have been opened in FileMaker.

    NOTE (closing ebase files): It is desirable that the automated parts of EDTT be started with no Ebase files open. If the user opens Ebase files for any reason (to perform "find all" commands, to check data contents, etc.), all Ebase files opened during these actions should be closed before running "data import" scripts. Please note that opening of one Ebase file will sometimes result in additional, related ebase files also being opened. Ideally, *only* EDTT databases will be open when the user starts the automated data transfer scripts.

17. Start the **"Run master script for data transfer scripts…"** step.

    For each ebase file, EDTT will:

    - Open the "target"/"new" database that will receive data

      The user will usually be prompted for passwords for both the "old" and "new". databases.

    - Ask the user if any "custom" fields are to be imported for this database

      If the user indicates that custom fields are to be imported, the user will be prompted with the usual data import dialogs used by FileMaker during a manually performed data import. This is necessary so the user can specify the mapping of data from custom fields in the "old" database to custom fields newly added to the "new" database. If the fields added to the "new" database use the same name, data type, etc. as the corresponding fields in the "old" database, the user need only indicate "import order" by "matching fields". If the "new" database uses different field names than the "old" database, the user will have to individually indicate the mapping of data from custom fields in the "old" database to newly added fields in the "new" database.

      NOTE (exact field names): It is recommended that custom fields in the "new" database use exactly the same name, data type, etc that was used in the "old" database. This *greatly* simplifies the data transfer process. If there is a need to rename fields for clarity, etc. it may be wise to postpone that renaming until after data transfer operations are complete and verified correct.

- Attempt an appropriate import of data into the “new” database file from the corresponding “old” database file.

18. Start the **“Assign next serial numbers”** step.

    For those files where a serial number field in the “new” database must be manually set, EDTT will assist the user in performing this task by performing the following actions:

    - Display a window indicating that a serial number field’s “next value” option must be reset,
    - Calculate the value to which the serial number field’s “next value option must be reset (displayed on the “Ebase data transfer steps” layout)
    - Open the “Define Fields” window of the database to facilitate the user’s manual task in resetting the serial number field’s “next value” option. .

    To ensure data integrity, the user *must* manually reset the “next value” setting of serial number fields when this is indicated by EDTT.

    In early issues of FileMaker V5, there was no script step available to completely automate this type of task. However, EDTT was set up to perform as many actions as possible to streamline this task for the user.

    It is essential that this step is performed correctly, or serious data problems could result if newly added records are assigned serial number values that are already assigned to preexisting records.

    NOTE (serial number options): If you do not understand the use of serial number functions or are not familiar with FileMaker “serial number” options, it may be wise to seek expert assistance rather performing a data transfer by yourself - even with the assistance of EDTT.

19. Check the contents of the new database files closely to verify that data has transferred correctly.

Databases checked for custom objects

Scripts in the “edtt_custom_objects.fp5” database will check for custom objects in the following databases:

actions_.10x
contact_.10x
custom_.10x
groups_.10x
impenh_.10x
invoices_.10x (only for V1.0.2 and V1.0.3 data sources)
names_.10x
notes_.10x
paymnts_.10x
pledges_.10x
profile_.10x

setup_.10x
solicit_.10x
srcdefs_.10x
staff_.10x
summary_.10x
zip_.10x
zipcode_.10x

These scripts identify any fields, layouts or scripts that do not have counterparts in the new Ebase databases.

Transferred Data

The “Ebase data transfer tool” should import data into the following Ebase database files from the corresponding source databases::

actions_.103
contact_.103
custom_.103
deleted_.103
document_.103
groups_.103
impenh_.103
names_.103
notes_.103
paymnts_.103
pledges_.103
profile_.103
setup_.103
solicit_.103
srcdefs_.103
staff_.103
summary_.103
zip_.103
zipcode_.103

Corruption problems during “imports”

Some cases have been encountered where a corrupted file just couldn't be used as a source during an "import" operation. It seems that this is sometimes one of the side effects of database corruption. Since the ebase data transfer tool uses “import” steps in it’s scripts, a file that cannot be used as a source for an import operation presents a problem.

You can test for this circumstance by following these steps:

1. Create a copy of your "new" file using a different file name.
2. Use FileMaker to open the file just created
3. Start a "manual" import operation using the "old" file as the data source.

If the import operation fails when attempted manually like this, the problem is probably best attributed to corruption in the source file. If a manual "import" won't work with a file, there is no reason a scripted “import” such as those performed by the data transfer tool should be expected to work.

In at least some cases, the data can nonetheless be moved into the new files using an alternative, “manual” approach:

1. open the "old/bad" file (let's designate this file "A_old"),
2. export all data from "A_old" to a new database file (we’ll designate this file "A_old_export")
3. open my "new" database (we’ll designate this file "A_new")
4. do a manual import from file "A_old_export" into file "A_new".
5. reset serial numbers, etc. as needed

Typically, Ebase then seems to incorporate the data that was manually imported in the correct way and seems to function correctly. Apparently, the new file created at step 2 sometimes does not include the corruption that interferes with the import, even after the old data is placed in that file.

Assistance and Use Agreement

The application is provided for use free of charge, and the user is free to distribute or modify the tool under the following conditions:

1. The distributed product must retain mention of the original author (available below) and contact information for the author.
2. Distributions of modified versions of the product must indicate that they have been modified, who performed the modification, and when the modifications were implemented.
3. Those distributing the tool must not license it, register it, or in any other way restrict the distribution of the tool by others. You obtained it without charge and with the right to modify and distribute – those who get it from you should have the same rights.

The user accepts full responsibility for evaluating and preventing problems that might arise from use of the application. The author cannot guarantee that the tool will perform correctly in every possible environment where it might be used.

Copies of the tool and/or assistance in use or modification may be obtained from the author:

Larry Bednar
Bednar Consulting
3226 NE 24th Ave.
Portland, OR 97212
503.493.8542
larry@quimdas.com

The author welcomes and encourages feedback and information about use and modification of the tool, which will be used for continued improvement of the product.

Due to time restrictions, suggestions for modification will be prioritized together with the author's other ongoing projects. As a result, even suggestions of high value and high ease of implementation may *not* be implemented immediately. However, the author will maintain a formal record of problem reports/suggestions to ensure that comments and ideas for product improvement will not be accidentally lost.

Acknowledgments

Thanks to Tony Kaperick and Allen Poole for reporting "bugs", suggesting or implementing fixes, trying EDTT on hardware and operating systems not available to the developer, and pointing out errors and possible improvements in the documentation.