Empire Earth
Hex editing guide

Want to edit the units, tech, graphic, etc in Empire Earth, but you don't know what to do? Then this guide is for you!
This guide will show you how to edit this game, but assume you already know how to use Hex editors with EE.
Especially, it require you to read the Alfrqan's, JCFG's and BOOST15673's guides. I need to thank Alfrqan and JCFG for discovering many basic values in the db files (my work started from their results), and BOOST15673 for both discovering new values and grouping all guides in his guide (after all, you could just read BOOST15673's guide, which include the other ones...).

Facts: until 2016 we didn't really know how much this game was moddable...

Now, you know those dat files in EE are a real mess, with many hex values in that horrible format, so you can't just open them with an hex editor and expect to create a new "Cold war" epoch between "WW2" and "Modern".
You need a specific editor for EE to help you. There are few editors on the net, but I suggest you use my editor, which is the best one (sorry for my lack of modesty!): Empire Earth – DB Editor. This guide assume you use this editor from now on.
This editor allow you to open many many dat files and edit all values in integer/float/text formats instead of hex format (including the still unknown ones). It offers a simple graphic interface which allow you to find what you're looking for but retain some "hex" style.

Facts: the Empire Earth – DB Editor is open source, and you can contribute to improve it!

Run this editor and open all the
supported db*.dat files of your game (you need to extract them from the data.ssa with EEStudio and decrypt them with Wolfie's EE Multidecompressor).
This guide assume you read the Manual for Empire Earth – DB Editor before continuing.

You can now do many things, but let's first you need to know something more about the db files we're going to edit:

GENERAL:

- There are no values with lenght 2, 3 or the whole line. In the programming world there are only values which use 1 byte (boolean and char), 2 byte (short), 4 bytes (integer and float) or 8 bytes (long and double). Strings take a variable number of bytes (1 or 2 bytes for character, depending on the language and the char encoding).
  EE only use booleans (1 byte), integers (4 bytes), floats (4 bytes) and strings. So you will never values with 2 or 8 bytes.
  Be careful with float. Did you notice those values like 00 00 20 41, CC CC CC 3F or similar, which seems huge numbers? They are float numbers. They can't be read like integers as they use the exponent + fraction format, but with bytes in reverse order (like all values in these files). Did you remember few years ago I discovered the Range was using an odd exponent formula... that's right! I was headbutting my table trying to decode a float value! Range, Area effect, Speed, etc... are all floats.
  If you read FF FF FF FF it's not a float value, but an integer instead: -1, which is often used by EE as "null" value for links.
  The field ID and Sequence Number must be **UNIQUE** in the file. If you want to change them, ensure no other entry in the file is using that ID or Sequence Number.
- Some dat files have entries with links. These links point to other entries, even (and especially) in other files. A link is simply the ID of the target entry. So, if you alter an entry's ID, remember to update all links to this entry.
  (In other words: if A, B and C have links to B, and you change B's ID, then you must update A's, B's and C's links, because they still point to the old B's ID).
- The editor hide all undefined entries (by default, but you can make it show them), which are all entries with a negative ID and/or Sequence Number. You can safely take and edit/define these undefined entries, BUT you must not use the undefined entry at the top (if any). If the file contains an undefined entry at the top, that entry is used by the game as "default" and must not be touched. Pick any other undefined entry or create a new one (if allowed). If you can't create a new entry, then that file must have a fixed set of entries and you must not create/remove them (but you can still edit the existing ones!).
- Sad but true, there are files which doesn't define an ID field, and their ID is simply the order they are defined in the file, which means you need to either count one by one the entries to find their ID or use Empire Earth – DB Editor (which calculate them for you) to save 20 minutes of your life for each entry you're searching for...

TECH TREE:

- While all dat files define a single "num of entries" followed by all entries, dbtechree.dat use a different structure.
  In dbtechtree.dat there are multiple groups of {"num entries", entries in the group}. Every group represent an epoch, and the "num entries" is always followed by the "epoch" tech, which is then followed by the techs which belong to that epoch.
  If you know how to count, you know the consequence: there are 14 groups (15 in AOC), one for each epoch in order.
  - Side note: my editor manage the num of entries for you and you don't have to worry anymore about these details.
- Everything is a tech: even objects have a tech assigned. After all, the ability to create a unit is a "technology"...
- Many techs are not objects, but alter your civilization. When you research a tech, it fires an event: an event define a list of effects to execute. Effects are just elementary operations.
  - Example: an "upgrade" tech fire an event, which always use 3 effects:
    - effect: disable the old objetc's tech
    - effect: enable the new objetc's tech
    - effect: replace old objects with new objects (upgrade all alive objects on the map to the new version. The new objects will inherit all upgrades and all changes made by other techs to the old objects).

Basic operations:

First, lets do some basic operations, which are required to execute the advanced ones:

Change an entry's value:
Easier than 2+2 (if your answer is not "4", you have a serious problem!): open any db file, select any entry in the list, change a value and click Save Entry, then Save to File.

Cancel changes:
If you want to cancel the changes made to an entry, and you didn't hit the "Save Entry" button, you can click on "Reset Entry" to cancel any change (alternatively, you can select another entry and re-select this entry again). If you already saved to entry... you can still reload the file! If you saved the file, the editor has created a db*.bak file, which you can rename to db*.dat and recover the previous version of the file. Finally, the editor also place a db*.orig file, which is never saved again, until you delete it.

Add/Remove a extra field for an entry:
In dbtechree.dat and dbevents.dat you can create new fields, where you can put extra data (usually, they are ID to other entries, which may be in other files). Open one of these files and add a new field with "Add ID". Then remove it with "Remove ID".

Add/Remove an entry in the list of entries:
Some db files define a limited set of entries and can't use this feature, but many db files allow you to add/remove entries. If the file support this operation, you can right click on the list of entries to show a menu. If the menu doesn't appear, the file doesn't support this feature. So, open a db file which support this operation and add a new entry. Then right click on it and remove it.

Change an entry's ID:
This is similar to "Change an entry's value", except the ID is not a common value that you can change without thinking.
If you change this ID, and some entries (even in other files) have links to this entry, you need to update those links too, because they still point to the old ID. So open all entries which point to this object and update their links.

> Example: for techs, you need to update the object (only if the tech represent an object), the building which can build the object/research the tech, any tech which require this one (like: "Upgrade to Long sword" require "Upgrade to Short Sword") and any effect (in dbeffects.dat) which alter this tech.

For this reason, only change an ID if necessary and if you know what' you're doing.

Move tech to another epoch:
This is (a bit) more complex. Go to any epoch, select the entry you want to move to another epoch, rigth click and select "Move to epoch" and you need to select the new epoch. The tech will be moved there and it will automatically receive a new ID within the epoch's range of IDs (needed change, else the game crash). Now, don't forget to update the links (look at “Change an entry's ID). dbtechtree entries are only used by dbobjects.dat and dbeffects.dat, so check these ones.

Define an entry:
You know how to add an entry, right? Well, if you want to "define" an entry, is another story.
First you need to add a new entry, or take an existing "undefined" entry. If you uncheck "Hide undefined entries" you'll see a bunch of <undefined> entries in the file. These entries are just trash for the game, which doesn't even consider them, UNLESS you define them (when you give them a sequence number and an ID, the game will start to consider them). I suggest you reuse these entries so you won't further increase the file size (unless you have a reason to add a new entry). Anyway, you'll have an entry with default values.
Now: almost every entry (with very rare exceptions in few files) have a Name, Sequence Number, ID.
The name is not really important: it's useful to you, but it's not used at all by the game. The sequence number and ID must be unique and in order in the file. When you create a new entry, the editor automatically find and assign unique sequence number and ID.
Finally assign all other fields.

Now, you have everything you need to start doing something greater!
The next section will raise the difficult from Easy to Hard, but if you manage to understand how the whole system works, you can really become an amazing modder for this game!

Advanced operations:

Create new Techs and Objects:

This is a very difficult task: the procedure vary, depending on what are you planning to create: upgrade, improvement or object?

- "Upgrade" technology (example: upgrade "Composite bow" to "Long bow"):
  - Requirements:
    - The tech of the object to upgrade
    - The tech of the upgraded object
  - Procedure:
    - Of course, create the new tech in the list and assign it name, ID, seq num, language ID, resources cost, build time, starting and ending epochs (even if starting epoch is not really used by the game...).
    - Set "Is object?" to 0 and "Type of tech" to 7 (Upgrade). Copy the other NOT-Link fields from other similar techs.
    - Assign the "Object which can build it" extra fields (as you may guess, you can research this tech from these objects, which are usually buildings), then copy the last extra field in "Last build object".
    - Take/Create a button for this upgrade. If there's already a sequence of ugrades for the object (like Clubman → Maceman → Short Sword → Long sword), you must reuse the same “upgrade to” button. Assign the button to the new tech.
    - Go to the tech of the upgraded object, find the field "Unlocked by tech". Set this field to this "upgrade" tech you're creating. This ensure the new object won't become available unless you research this tech (or a trigger add it).
    - Now you need to create the tech's event and its effects.
      - Create the effects "Disable tech", "Enable tech" and "Replace objects. There are tons of effects like these in the file. You could even duplicate the existing ones and replace their objects/techs ID fields with your objects/tech IDs.
      - Create an event, add 3 extra fields and select these 3 effects (the order doesn't really matter).
      - Assign the event to the tech.
    - Ensure the tech is enabled (toggle the buttons at the end like this: ON, OFF, ON, ON)
    - Finally, open dbobjects.dat, find the objects which must research this tech, increase their "Num buildable techs" by 1 and append the new tech in the list of fields "Can build tech" at the end.

- "Improvements" technology (example: increase priest's range, +15% wood gather rate, ...):
  - Very similar to the above one, except for:
    - "Object ID" stay empty
    - "Type of tech" is 0
    - the effects are differents (obviously!)

- "Object" technology (example: the technology associated to the object Marine, which define its cost, build time, epoch, ...):
  - Requirements:
    - All object data (graphic ID, sounds ID, effects ID, family ID, etc...)
  - Procedure:
    - Create the new object and assign all those basic values (Name, ID, Seq num, HP, Attack, all Links, etc...)
    - Create a new tech for the object and assign all basic values (Name, ID, Seq num, Cost, Build time, epochs, ...)
      - You need to assign the Language ID to the object and not to the tech.
      - Set "Is object" to 1 and "Type of tech" to 1. Copy the other NOT-Link fields from other similar techs.
      - Set "Object ID" to the object you created above and set "Build from object" and "Last build object" to the objects (usually buildings) which will be able to create this object.
    - Return to dbobjects and upgrade the list of techs (right click on the field "Tech ID" and... you'll see...)
    - Now you can assign the tech you've created to the object you've just created (in field "Tech ID").
    - Finally, open dbobjects.dat, find the objects which must create this object, increase their "Num. buildable techs" by 1 and append the new tech in the list of fields "Can build tech".