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 other guides in his guide (after all, you could just read BOOST15673's guide, which include the other ones...).
When you read the other guides, you're ready to read this one.

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 a 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 allows you to open 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, so you can easily switch between my editor and hex editor if you need.

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

Before you continue: from now on, this guide will assume you read the Manual for Empire Earth – DB Editor and knows how to perform the basic actions described there.

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 length 3, 5 or the whole line. In the programming world numbers only use one of 1, 2, 4, 8 (and sometimes 16) bytes. Strings take a variable number of bytes (1 or 2 bytes for character, depending on the char encoding). EE only use boolean/integer (1 byte), integer (4 bytes), float (4 bytes) and string. There are no numbers 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 instead. 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). Maybe you remember: few years ago I discovered the Range was using an odd exponent formula... that's right! I was headbutting the edge of my table while 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 (but sometimes it uses 0, or big number outside any range… someone was smoking good stuff while programming this game…).
- The fields 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. It seems Sequence Number must be sequential, while ID has no such limitation.
- Some entries have links: they point to other entries, sometimes in other files. A link contains simply the destination file and ID.
- By default, the editor hides all undefined entries, which are all entries with a **negative** ID and/or Sequence Number. These entries are ignored by the game, so you can safely alter these entries (and even turn them into valid entries by giving them a valid ID/Sequence number), BUT you must never touch the undefined entry at the top (if any). If the file contains an undefined entry at the top, usually that entry is used by the game as "default" or "null" and so it 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).
- 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 count the entries one by one to find their ID. Empire Earth – DB Editor will calculate these ID for you.

TECH TREE:

- Many dat files define the field Num. entries at the beginning of the file, followed by all entries of the file.
  dbtechree.dat instead is divided into group, one for each epoch: every group starts with its own counter Num entries, followed by the "Epoch tech" (which is not included in the counter), followed by all techs which belong to that epoch.
  So, there are 14 groups (15 in AOC), and they appear like this in the file:
  - [Num. Paleo epoch techs], [Paleo epoch], [List of Paleo epoch techs], [Num. Stone epoch techs], [Stone epoch], [List of Stone epoch techs], [Num. copper epoch techs], …
- Every epoch tech defines a range of 1000 valid IDs, starting from the value in (44) Epoch IDs.
  All technologies which belong to this epoch must have an ID in this range, **else the game will crash**.
  - Example: Copper epoch has Epoch IDs == 3000, so all techs must have an ID in the range [3000, 3999].
- Basing on the field (23) Tech type, a tech may create an object, upgrade a unit, improve your civilization, etc.
  When you research a tech of type 0 – Research, it fires an event (dbEvents.dat), which contains a list of effects to execute (dbEffects.dat).
  Effects are elementary operations, like enable/disable tech, change button graphic, change an attribute, etc.…
  - Example: an "upgrade" tech fire an event, which usually contains 4 effects:
    - effect: disable the old object's tech
    - effect: enable the new object'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)
    - effect: enable the next "upgrade" tech for this object (sometimes upgrades doesn't contain this effect)

Advanced actions:

Require: Base actions from the Empire Earth - DB Editor Manual

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 and give it new ID and Sequence Number.
I suggest you reuse these entries so you won't further increase the file size (and some files seems to have a limit), but as I already said, you must never touch the undefined entry at the very top of the list (if any).
Anyway, now you have an entry with default/invalid 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, but the Sequence Number and ID must be unique in the file. When you create a new entry with the editor, it will automatically find and assign these 2 fields, but you can change them if you want.
Finally assign all other fields (you can take inspiration from similar entries, if you don't know what to put there).

Create new Techs and Objects:

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

- "Upgrade" technology (example: upgrade "Composite bow" to "Long bow"):
  - Requirements:
    - A button entry for the upgrade tech (if the object has other upgrades, use the same button as those "upgrade techs")
    - The object and tech entries of the object to upgrade
    - The object and tech entries of the upgraded object
  - Procedure:
    - Create a new tech in the list and assign it Name, ID, Sequence Number, Language ID, all resources costs, Build time, Starting Epoch and Ending epochs (even if we're not sure Starting Epoch is used at all by the game...).
    - Assign the above button to the tech.
    - Set Is object? to 0 and Type of tech to 7: Upgrade class. Copy the other NOT-Link fields from other similar techs.
    - In Build from object extra fields place the objects which will be able to research this tech. If you don't use the editor, you'll have to manually copy the last extra field to Last build object.
    - Go to the tech of the upgraded object and go to the field Unlocked by tech. You must set this field to the 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/Define the effects "Disable tech", "Enable tech" and "Replace objects. There are tons of effects like these in the file, so if you don't know how, to create them, take inspiration from the other effects.
      - Create/Define an event, and add 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 object(s) which must research this tech and append the new tech in the list of fields Can build tech at the end. The field (224) Num. techs it can build is automatically updated by the editor. Without editor, you must update this field yourself.

- "Improvements" technology (example: increase priest's range, +15% wood gather rate, ...):
  - Very similar to the above one, except for few differences:
    - Object ID is empty
    - There's no upgraded object
    - Type of tech is 0
    - The effects are different (obviously!)

- "Object" technology (example: the technology associated to the object Marine, which define its cost, build time, epoch, etc....):
  - Requirements:
    - All object/tech data (including graphic, sounds, gfx effects, language, button, etc...)
  - Procedure:
    - Create/Define a new object and assign the values: Name, ID, Sequence Number, HP, Attack, all links, etc....
    - Create/Define a new tech and assign the values: Name, ID, Sequence Number, Costs, Build time, Epochs, etc....
      - You need to assign the Language ID in the dbObject.dat entry, not in the tech.
      - Set Is object to 1 and Type of tech to 1: Object. 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. If you use the editor, it will update Last built object automatically.
    - Return to dbObjects.dat, right click on (225) Tech ID and refresh the list of techs. Now assign the tech in this field.
    - Finally, open dbobjects.dat, find the object(s) which must research this tech and append the new tech in the list of fields Can build tech at the end. The field (224) Num. techs it can build is automatically updated by the editor. Without editor, you must update this field yourself.
    - If this object is an upgrade of another object (or vice versa), now you must follow the "Upgrade" procedure