

EBASE DEVELOPER CONVENTIONS

Here's what we've been using as a guide to naming tables, fields, layouts, scripts and relationships in ebase v2. This will probably be revised as we go along. The tables at this point reflect a minimum level of compliance with these conventions. Where things get messy is where we haven't made any decisions yet.

In general, names should reflect, as best as possible, the thing it is naming, in plain English, using proper grammar.

Basic interface terminology

Layouts are separated into:

- The **header**
- The **NavBar** (navigation bar on left, also know as the NavPortal because it is actually a portal whose value is conditional upon the user's identity)
- The **Data display** (the center part of the screen)
- The **MetadataBar** (right hand margin)
- The **Footer**

Lists may or may not display the NavBar or the MetadataBar

Layouts may represent:

- **Forms** (input or reports ... looking like a letter)
- **Dialogues** (requiring user input, menus are replaced by buttons)
- **Reports** (forms or tables looking like a spreadsheet)
- **Portals** (lists that appear within forms or reports)

Table naming convention

Tables should be named to accurately describe the nature of the data that they contain. Tables are generally represented as the plural of the data they contain (counter to programming convention), or they are a word that itself implies a plural (Example: "LOG).

Tables will use the current standard FilemakerPro file extension (Example: .fp5)

Location reference convention

For troubleshooting and communication purposes, the place where an issue is identified should be indicated like this:

TABLENAME:LAYOUTNAME:FIELD/BUTTON/SCRIPTNAME

So if there's a problem with adding a contact name and it appears in the first layout, you would refer to it as:

Contacts:AddContact:DupePortal:_ccFullName

to indicate that there is an issue with the field "_ccFullName" in the Duplicate checking portal.

Field naming convention

A person should be able to look at a field name and understand the category of data it contains and the nature of that data. Fields that hold data should be clearly distinguishable from fields that contain derivative data (Example: calculations).

Field names should be in plain English as much as possible, and set up to group like data together when sorted in alphabetic order.

Names will not contain spaces. If a space is required, use an underscore character (_)

Multiword names will have each word initialized (Example: _cdModBy)

Common abbreviations may be used. (See list of abbreviations, below). Abbreviations should be limited to no more than 5 characters; three is preferred.

DATA CATAGORIES SHALL BE REPRESENTED AS FOLLOWS:

No prefix: Raw data fields: no prefix. Like data grouped together.
Example: Names

- NameFirst
- NameLast
- NameMiddle
- NamePrefix
- NameSuffix

_cX	<p>Calculated fields</p> <p>_c = Calculations</p> <p>_ct= Text</p> <p>_cc = Concatenation, text</p> <p>_cn = numeric</p> <p>_cd = date</p> <p>Example: <code>_cdAddBy</code> (A calculation that looks up the current username and inserts it into the field).</p> <p><code>_ccFullName</code> (A concatenation of NameFirst and NameLast</p>
_f	<p>Boolean flags, binary numeric fields with value of 1,0 (true, false) with a value list of Yes/No. Example: <code>_fDelete</code> (the delete flag). Value list: 1/0 represented as Yes/No</p>
_g	<p>Globals: Fields that contain a single value across every record</p> <p><code>g[table][layout][on/off]</code></p> <p>Example: <code>_gContactEntryOn</code></p> <p><code>_gEbaseMenuOn,</code></p>
_k	<p>Key Fields: no matter what their nature (derived or not). Keys are singled out for special treatment to alert developers who want to "clean up" a file to beware when fooling around with key fields.</p> <p>Example: <code>_kTrue</code> (the boolean value "1")</p> <p><code>_kContactID</code></p> <p><code>_kTransactionID</code></p>
_s	<p>Summary fields</p>
_tsi	<p>Filemaker interface hacks (or "Tricks, Stupid interface")</p> <p>Example: <code>_tsiHighlightDeleteRecord</code> (turns the background of a record flagged for deletion red).</p>

~/ Fields which are of questionable utility that need to be examined before deletion

Script Naming Conventions

- ❑ No spaces in names.
- ❑ Script names should describe their function.
- ❑ Script names should follow field naming rules but use the prefixes below to categorize them. Scripts preceded by script class identifier (see below).
- ❑ Scripts should be sorted alphabetically within script classes and separated by a class identifier when you open ScriptMaker.
- ❑ Scripts should be commented as much as possible.

Nav- Navigation to a place or function
 Example: Nav-MainMenu (go to MainMenu in Main.fp5)
 Nav-ContactMainMenu (go to MainMenu in Contact.fp5)
 Nav-Find (go to the Find procedure)

Admin- An administrative procedure (function is kind of fluid)

Delete- Scripts associated with deleting records

Com- Scripts associated with communications functions

Import- Scripts associated with import/export functions

Proc- Data processing procedure: Import, de-duping, CASS certifying, etc. (function is kind of fluid). Proc- scripts are generally not invoked by a user, but are generally sub-routines within other scripts that may invoked by a Nav-script.

Find- Find a set of records

Sort- Sort a set of records

~/ Scripts which are of questionable utility that need to be examined before deletion

Layout Naming Conventions

Not many so far. But they should have no spaces, be descriptive, and ideally have the same name as the scripts that call them out. Example: MainMenu (called out by Nav-MainMenu)

Find (called out by Nav-Find)

Generic naming convention is to describe the layout function, then the layout subtype

- MenuContact
- MenuItem
- EntryShort
- ListShort
- ListLong etc.

- x Prefix used to indicate hidden, utility layouts that do not appear but **SHOULD NEVER BE DELETED**. The names of these layouts should relate to the script(s) that call them.
- ~\ Indicates questionable layouts that need examination before deletion.

Relationship naming conventions

Relationship names should reflect which tables are joined by what keys in the following format:

CLIENTTABLE_KEYNAME\HOSTTABLE_KEYNAME

Table names should be abbreviated to 3 characters (if possible), but you may use up to 5 if you need to distinguish between various tables.

Example: To express a True=True relationship between Contacts.fp5 and Transactions.fp5 you would use

CON_kTrue\TRANS_kTrue

Note that the name does not express the ordinality or sort orders that may be defined in the relationships.

Primary menu structure

- Main Menu (central navigation point, reports metrics)

- ❑ Contacts (work with individuals)
- ❑ Payments (work with money)
- ❑ Reports (work with reports...duhh!)
- ❑ Communications (activist outreach vehicles: email, letters, add data to web site)
- ❑ Import/Export (move data in and out of ebase)
- ❑ Admin (administrative functions, only seen by ebase administrator)

Table names and functions

(This is incomplete and not up to date. Refer to the document entitled “ebaseentities” for a complete list of files, their functions, and which build versions they appear in.)

DATA STORAGE

Contact.fp5	Stores one-one data on individuals
Location.fp5	Stores location data (address, geocode)
ContactLocation.fp5	Join table between contact and location that stores the role and organizational information of a person located at a specific location.
Items.fp5	Stores many kinds of data related to Contacts.
Friends.fp5	Stores information on relationship of Contacts to each other.

REPORTING/FUNCTIONAL PORTALS

Reports.fp5	Draws data from many tables to create reports.
Communications.fp5	Portal to communications transactions
Payments.fp5	Portal to payment transactions

UTILITY STORAGE

FoundContacts.fp5	Utility table, resident on client machine, that stores transient found sets containing ContactID's
FoundTransactions.fp5	Utility table, resident on client machine, that stores transient found sets containing TransactionID's

NAVIGATION/SECURITY/SYSTEM SETTINGS

Navigator.fp5	Contains transient data on where users are in the collection.
Nav_Admin.fp5	Contains user account and menu options data.
Ebase.fp5	Stores global information about the organization and license data.
Main.fp5	Stores global information about the system. Acts as a "switchboard" between modules.

IMPORT EXPORT TOOLS

TBD

WEB TABLES

WebContacts.fp5	Serves as an interface between Contacts.fp5 and the web interface.
WebInfo.fp5	Stores dynamic content to be served up by the web interface.

DEVELOPER TOOLS

EbaseUnlocked.fp5	Contains ebase.fp5 minus proprietary plug-in registration codes.
DevLogin.fp5	Developer Login file allows Admin to change/modify layouts in various table normally locked to users.

Transaction coding conventions

The 8 data buckets abstract data using the following categories:

- Class
- Owner filter
- Category
- Item
- Version
- Iteration
- (user defined)
- LegacyCode

--Alternatively----

- Class
- Owner (indicates which user level can view the data contained in this transaction)
- Project (links a transaction to a project transaction)
- Category
- Item
- Version
- Iteration
- LegacyCode

1. The first three buckets are required to define a data type.
2. Transaction codes must contain a UNIQUE and descriptive title, ideally no with more than 20-25 characters
3. Transaction Description must contain information that segregates THIS code from others.

4. Transactions contain data fields of various data types that can repurposed to conform to the transaction code. For example, a transaction with the code

Payment/Development/Membership/Family

may use `_ccAddDate` to indicate when the payment transaction occurred, but another transaction coded:

Communication/Activist/Membership/Family

may use another date field in the transaction record to indicate when a conversation happened.

It's up to each organization to define the nature of the data fields contained in the `Transaction.fp5` table. ebase provides a raw structure that can trigger dynamic reports, but each organization will need to adopt transaction codes that accurately model it's business practices and information storage needs.

Each transaction code bucket should use unique data within the bucket structure (so you don't use "ebase" in both the Owner and Version buckets for example).

List of Abbreviations

Exp	Export
Com	Communication
Imp	Import
Mod	Modified
PCL	Primary Contact Location