

# FileMaker Pro Development Standards



Last Updated: Jan 22, 2003

# Table of Contents

---

<b>Welcome</b> .....	4
<b>Definitions</b>	
ASCII Codes .....	5
<b>Access Privileges</b>	
Passwords .....	6
Groups .....	6
<b>Project Folders</b>	
Overview .....	7
Folder Structure .....	7
Folder Naming .....	8
<b>Naming Conventions</b>	
General .....	9
Files .....	9
Fields .....	11
Layouts .....	15
Relationships .....	16
Value Lists .....	18
Scripts .....	19
<b>Standard Fields</b>	
Overview .....	21
List of Fields .....	21
<b>Layouts</b>	
General .....	25
Organization .....	25
Consistency .....	25
Cross Platform Considerations .....	25
Navigation .....	26
Fonts .....	26
Buttons .....	27
Portals .....	28
Data Entry Layouts .....	28
List Layouts .....	29
Table View .....	29
Reports .....	30
Find Layouts .....	30
Developer Layouts .....	31
Dialogs .....	31

# Table of Contents

---

<b>Scripting</b>	
Organization .....	32
Opening Scripts .....	32
Browse Mode.....	32
Find Mode .....	33
Restoring Options .....	33
Dependencies.....	33
Previewing.....	33
New Related Records .....	34
Script Steps to Avoid.....	34
<b>Preferences</b>	
General .....	35
<b>Plug Ins</b>	
Recommended Plug Ins.....	36
<b>Start Databases</b>	
Introduction.....	37
Main.FP5.....	38
Developer.FP5.....	38
Utility.FP5.....	39
Defaults.FP5.....	39
<b>Developer Toolkit</b>	
Toolkit Contents	
Graphics Library (Buttons, Icons) .....	40
User Interfaces (Samples).....	40
User Feedback Database.....	40
Drop Rename Utility (Mac Only) .....	40
Toolkit Future Additions	
Screen Sizes.....	41
Standard Fonts .....	41
Scripts & calculations .....	41
Login System .....	41
States/Provinces .....	41
Zip Codes/Postal Codes.....	41
<b>Developer Edition</b>	
Runtime Solutions.....	42
<b>Dates/Y2K .....</b>	<b>43</b>
<b>Web Development</b>	
Variable Names .....	44
Constant Names .....	44

# Welcome

---

I'd like to take this opportunity to say thank you to the many people who contributed to the creation of this standards document. I'd like to especially acknowledge the following people for their valuable contribution; Kieren MacMillan, Steve Hearn, Steve Lane, Andrew Nash, Andy LeCates, Eric Culver, William Moss and Steve Thoms. To Steve Whitlow for his assistance in the Canadian/U.S. translation process - we believe we have removed all phrases starting with or including the words "out and about". If I have left anyone's name out I apologize in advance and will expect your strong email to that effect. These standards are the collective work of these fine people and therefore represent some of the best ideas being used in FileMaker development today. By implementing them in your organization you will benefit from the knowledge and combined experience of all these developers.

The purpose of creating and distributing these standards, free of charge, is to help promote high quality database development in the FileMaker community. Good practices will inevitably result in benefiting the reputation of all FileMaker Pro developers. Everyone is welcome to these standards. Please direct anyone you feel would benefit from it to our website to download their own copy.

<http://www.coresolutions.ca/Resources/standards.lasso>

Understandably there is definite distinction between having standards and using them on a day-to-day basis. The most predominant reason why standards don't get used is simply because they are too rigid and difficult to implement. Over the years, I've worked with many standards and development methodologies — I've seen some succeed and some fail miserably. Standards don't work if they are too rigid and, on the other side of the coin, they fail if they are not comprehensive enough. To be useful and workable they must be flexible, adaptable to any project, easy to follow, easy to implement and not impede upon the developers' creativity or the project development itself.

Developers who follow good standards promote consistency in their database solutions, making the transition easier for other developers to get up to speed on their development projects. If you work in a team environment this is especially important. As an added benefit, you'll reduce the time to develop, debug and support database solutions. Good practices will benefit all FileMaker developers and help promote professional database development.

We realize that this document is not the perfect solution but we believe it has strong merits and benefits. We strongly urge developers to email us with their constructive criticism and development wisdom [standards@coresolutions.ca](mailto:standards@coresolutions.ca). CoreSolutions Development Inc. will collect input and provide updates to the standards so that the information will be available to all FileMaker Developers.

Barney J. Lawn  
President  
CoreSolutions Development Inc.  
<http://www.coresolutions.ca/>  
[blawn@coresolutions.ca](mailto:blawn@coresolutions.ca)

# Definitions

---

## ASCII 32-126

### Standards

The standard 'ASCII 32-126' character set includes the following elements:

(*n.b.* 'Code' is the ASCII value in base-10 decimal notation; 'Char' is the resulting character)

Code	Char	Code	Char	Code	Char
32	SPACE	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	\$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

# Access Privileges

---

## Passwords

### Standards

1. All passwords shall be documented and stored in the Project folder (6b. Developer Docs) allowing easy retrieval while providing protection from access, modification, or other use by unauthorized personnel.
2. The master password(s) for “shrink-wrapped” solutions (*e.g.* programs available for commercial resale) shall not be distributed to non-employees.
3. Except as otherwise noted (*q.v.* §2, above), the distribution of passwords (*e.g.* to users, clients, or other developers) shall be at the discretion of the Project Manager.

### Guidelines

1. Every database should be protected with at least two passwords:
  - a) An “all access” developer password; and,
  - b) A user password known to the user.

All databases should not auto-enter any password unless otherwise dictated by the Project Manager.

## Groups

### Standards

1. Every database shall contain one group named “Developers”, which shall contain the primary (master) developer password(s).

### Guidelines

1. Scripts which perform database or interface “locking” functions (*e.g.* Zoom, Status Bar, *etc.*) should test for `PatternCount(Status(CurrentGroups), “Developers”) = 0` before locking — *i.e.* if the current password is a Developer password, the interface elements should remain unlocked.

# Project Folders

---

## Overview

The file set (*i.e.* databases and all support files) for every project shall be maintained in a consistent manner.

(*n.b.* Projects already completed or in progress are exempt from this requirement — however, in the interest of consistency, each project may be brought up to the standard at the discretion of the Project Manager.)

## Folder Structure

### Standards

Every project shall be maintained in a folder structure that contains at least the following. Folders that contain items should be suffixed with a Plus (+) sign for easy reference, e.g. Estimates+

1. Project Management.
  - 1a. Letters of Engagement
  - 1b. Proposals/Estimates
  - 1c. Correspondence
  - 1d. Contracts
  - 1z. Other
2. Requirements
  - 2a. Needs Analysis
  - 2b. Specs & Data Dictionary
  - 2c. Support Files
  - 2z. Other
3. Client Files
  - 3a. Data
  - 3z. Other
4. Support Files
  - 4a. Artwork
  - 4b. Reference
  - 4z. Other
5. Project Builds  
ClientCode\_WO#\_19990601\_0.0b1
6. Documentation
  - 6a. User Docs
  - 6b. Developer Docs
7. Deliveries
  - 7a. Delivery Report (text file)
  - 7b. Files Delivered
8. System Testing
9. Implementation Conversion
10. Changes
11. Passwords

# Project Folders

---

## Guidelines

1. The order and content of the project folders should conform to the logical progression of standard processes and business rules (*e.g.* sales to design to development to implementation to support).

## Folder Naming

### Standards

1. The numbering of each build of a solution shall adhere to the following conventions:
  - a. a database or solution newly created (*i.e.* in initial development) shall be assigned version number “0.0b1”;
  - b. a database or solution newly placed in production phase shall be assigned version number “1.0v1”;
  - c. the “b” or “v” value of each incremental build shall be increased by 1 (*e.g.* 0.0b2, 0.0b3, ..., or 1.0v2, 1.0v3, ...);
  - d. the version number of each significant new build shall be increased by one or more hundredths or tenths (*e.g.* “0.1”, “0.54”, *etc.*) at the discretion of the Project Manager — the size of the increase shall be based on the importance or quantity of the changes made.

2. The actual database files shall be stored in the Project Build folder, and shall be named according to the following convention(s):

ClientCode\_WorkOrderNo\_YYYYMMDD\_BuildNumber

*e.g.* WID\_550\_20000630\_0.17b5 (Widget project 550, as of 30 June 2000, version 0.17 build 5)

*n.b.* all years shall be four-digit years and the length of the folder name should not exceed 31 characters)

## Guidelines

[No guidelines have been set for this item.]

# Naming Conventions

---

## General

### Standards

1. The name of an object shall be descriptive of the contents or purpose of that object — where this is not possible, the object shall be properly and thoroughly commented or otherwise documented.

### Guidelines

1. Object names should consist of one or more words.
2. Object names should use only elements in the 'ASCII 32-126' character set.  
*(n.b. Using the 'ASCII 32-126' character set will avoid many potential cross-platform problems.)*

## Files

### Standards

1. File names shall use only the following ASCII characters:
  - a. the twenty-six lowercase and twenty-six uppercase letters of the alphabet {a...z, A...Z};
  - b. the ten numeric digits {0...9};
  - c. one underscore character {`_`} (used to separate the name from the extension); and,
  - d. one period {`.`} (used to begin the extension).
2. All FileMaker Pro files shall include the appropriate extension: “.FP3” for v3 and v4 files, “.FP5” for v5 files, etc. (e.g. Login.FP5).
3. In FileMaker Pro v3 or v4, database files not required to be viewed in the Hosts dialog (i.e. over a network) shall have an underscore (“\_”) preceding the suffix (e.g. Contact\_.FP5).

*(n.b. FileMaker Pro 5 files are exempt from this requirement, as there exists the option to share files as “Multi-User (hidden)”.)*

4. The length of the file name shall not exceed thirty-one (31) characters.  
*(n.b. The Windows 3.1 operating system is not supported — therefore, the “8.3” DOS-style naming limitation does not apply.)*

# Naming Conventions

---

## Files (Cont'd)

### Guidelines

1. All words in file names should be in title case (*e.g.* PeopleWeKnow.FP5, not PEOPLEWEKNOW.FP5 or peopleweknow.FP5).
2. File names should use singular or collective nouns in place of plural alternatives (*e.g.* Company.FP5 instead of Companies.FP5).
3. Files that contain records that act as line items for records in another file should be named appropriately (*e.g.* InvoiceItem.FP5 or WorkOrderLine.FP5).
  - a. Files that act as “join files” (*i.e.* files which contain records to link records from two or more distinct files) should be given a name that is descriptive of the purpose of the join (*e.g.* a file linking records in Student.FP5 and Class.FP5 might be called Roster.FP5 or Enrollment.FP5 or Registration.FP5). If it is impossible to find an appropriate name, the file should be named with the names of the two files and an appropriate word to indicate the function of the third (*i.e.* current, joining) file  
(*e.g.* File1File2Join.FP5 or File1File2Link.FP5).

# Naming Conventions

---

## Fields

### Standards

1. Field names shall use only the following ASCII characters:
  - a. the twenty-six lowercase and twenty-six uppercase letters of the alphabet {a...z, A...Z};
  - b. the ten numeric digits {0...9}; and,
  - c. the underscore character {`_`} (used to separate the field name from a prefix or suffix).
2. Prefixes shall precede field names according to the following conventions:
  - a. Developer field names shall begin with "z\_" (e.g. z\_counter).
  - b. Some developer fields will have two letter prefixes according to the following conventions.
    - i. **zc – CONTROL.**
      - a) Fields the developer uses to manipulate data. This is a very general category, it covers fields that don't fit in any other category and which the user normally doesn't put data into.
      - b) The prefix should be followed by a short name or reference: e.g. zc\_reportComment.

These fields shall follow a separator in Define Fields named: "ZC --- CONTROL ----".

- ii. **zi – INTERFACE.**

- a) Any fields the developer uses in the creation of the User Interface, to contextually provide the user with information or to store graphic elements for the User Interface. For example, these could be globals, containers and calculation fields used to create buttons and button graphics.
            - b) The prefix should be followed by the name of the primary function of the button, e.g. zi\_new.

These fields shall follow a separator in Define Fields named: "ZI --- INTERFACE ----".

- iii. **zk – KEYS.**

- a) Fields that are used expressly to form relationships.
              - b) The type of key is designated by a suffix. Refer to section 3b for additional information.
              - c) These fields shall follow a separator in Define Fields named: "ZK --- KEYS ----".

# Naming Conventions

---

## Fields (Cont'd)

### iv. **zl – LOG.**

- a) Fields the developer uses for field modification tracking, e.g. a 2<sup>nd</sup> set of lookup fields to store before and after values when a field is modified.
- b) The prefix should be followed by a short name or reference: e.g. zl\_address.
- c) These fields shall follow a separator in Define Fields named: "ZL --- LOG ----".

### v. **zv – VARIABLES.**

- a) Any fields the developer uses to store variable data. For example, counters and error codes.
- b) The prefix should be followed by the name of the primary function of the button, e.g. zv\_counter\_g.
- c) These fields shall follow a separator in Define Fields named: "ZV --- VARIABLES ----".

c. User fields shall have no prefix.

3. Suffixes shall be appended to field names according to the following conventions:

- a. all suffixes shall begin with a single underscore ("\_");
- b. key fields should use a letter to designate the type of key, i.e. "p" for Primary Key, "c" for Compound Key, "f" for Foreign Key, "a" for an Alternate Key, e.g. zk\_reference\_p; zk\_name\_f.

A Primary key is one that has been designated as the main identifier for the records in a file.

A Compound key is a calculation field that concatenates two or more fields generally used to form a Primary key. It's often used in the Join file of a many-to-many relationship.

A Foreign key is a field in a related child file that refers to a uniquely distinguishable record in the parent file.

An Alternate key is a field that is normally used in a secondary relationship (for use in a filtered portal for example). It is usually a calculation and may appear on either the right or left side of a relationship.

c. global fields shall be indicated by a "g" (e.g. zc\_nameCompany\_g);

# Naming Conventions

---

## Fields (Cont'd)

- d. Lookup fields shall be indicated by an "l" (e.g. `zc_nameCompany_l`);
  - e. fields which return values in a format not obvious from the field name shall be indicated by a letter which indicates the format returned: "b" for boolean, "d" for date, "i" for time, "n" for number, "o" for container, "s" for summary, "t" for text, "x" for index, "l" for lookup;
  - f. unstored calculations shall be indicated by a "u" (e.g. `zc_phoneFormatted_u`)
  - g. when multiple suffixes are necessary, they shall be appended in order of the conventions listed above (e.g. `zc_dateTime_tu`)
4. User fields are fields that the user interacts with, e.g. data entry fields or fields the user may sort in the sort dialog. Whenever possible user fields should not be named starting with an "X", "Y" or "Z" as it can confuse the sorting order. If this is not possible the developer should move any fields that begin with these letters above the Developer fields.
  5. For the purpose of grouping fields (*i.e.* in the Define Fields dialog), "separator" fields should be created with names that adhere to the convention outlined below.

- a) " ----- USER FIELDS -----" (The first character is a space).  
User fields are positioned below this separator and will appear first in certain dialogs, e.g. Define Fields and Sort Dialogs. This makes it easier for the user to locate the fields they will work with most.

All of the separators that follow are used for locating various developer only fields.

- b) "ZA --- USER CALCS ---"
- c) "ZC --- CONTROL ----"
- d) "ZI --- INTERFACE ----"
- e) "ZK --- KEYS ----"
- f) "ZL --- LOG ----"
- g) "ZV --- VARIABLES ----"

These fields should be created as Number or Date Globals to minimise their overhead. This makes it easier to find fields and sends a message to the user that certain fields are 'not their stuff'.

6. The first word of a field name shall be in lower case and any subsequent words shall be in title case (e.g. `dateInvoiceSent`, `nameFirst`, `nameLast`); acronyms and abbreviations within the file name shall be in all uppercase (e.g. `customerID`, `taxNO`).
7. "Constant" fields (*i.e.* fields acting as a universal, persistent, and predictable key to all records within a file) shall be named "zk\_constant".

# Naming Conventions

---

## Fields (Cont'd)

8. Whenever possible, the name of a primary key should match the name of the corresponding foreign key in all related files  
(e.g. Clients::zk\_clientID\_p and Phone::zk\_clientID\_f, not Clients::zk\_clientID\_p and Phone::zk\_clientNumber\_f).

## Guidelines

1. Within the names of fields that are apparent and accessible to end users (especially fields used in user-specified sorts or merge texts), words should be ordered from the general to the specific (e.g. nameFirst and nameLast, rather than firstName and lastName).
2. Fields should be arranged into groups and ordered according to an apparent and logical scheme (*n.b.* this ordering will only be apparent when fields being displayed in "Custom Order").
  - a. If a file contains multiple "constant" fields, each field should be named appropriately.
3. Primary Keys.
  - a) It is recommended that primary keys be of type Text. Though there is a theoretical speed benefit to using numeric keys, real-world tests show the difference is undetectable when compared to other FileMaker speed issues such as network lag. A text key allows a developer always to use text comparisons on keys and avoid any miscues due to type mismatches. It also allows application of the following guideline.
  - b) Though many developers use a sequential serial number to key their records, this scheme is vulnerable to corruption during imports. To reduce the chance of such corruption and increase the "uniqueness" of keys, it is recommended that text keys be composed by prefixing a serial number with a letter or two denoting the type of record being keyed. So invoices would have keys INV0001, INV0002, employees would be EMP0011, EMP0002 (this can easily be accomplished with FileMaker's serial number option in a text field). In addition to increasing uniqueness, this also makes the keys more legible if they are displayed for debugging purposes during development.

# Naming Conventions

---

## Layouts

### Standards

1. Layout names shall adhere to the following conventions:
  - a. Layouts, which are meant to be used or accessed by the Web Companion or any other CGI, interface, or external application other than FileMaker Pro, shall use only the following ASCII characters:  
the twenty-six lowercase and twenty-six uppercase letters of the alphabet {a...z, A...Z};  
the ten numeric digits {0...9}; and,  
the underscore character {\_}.
  - b. All other layouts shall use only characters from the 'ASCII 32-126' character set (*q.v.* the Definitions section of this document for more details).
2. Layout names shall be preceded by the following prefixes, according to the purpose or type of the layout:
  - a. "Edit\_" for layouts where data entry is permitted;
  - b. "Read\_" for read-only layouts;
  - c. "Find\_" for interactive layouts for specifying Find criteria;
  - d. "Rpt\_" for a previewed or printed report;
  - e. "Dev\_" for a developer-only layout;
  - f. "www\_" for layouts designed for external access (*e.g.* AppleScript, Java, or Web)

### Guidelines

1. Layouts should be arranged according to an apparent and logical scheme.

# Naming Conventions

---

## Relationships

### [Proposed] Standards

1. The name of each relationship shall be distinct enough from the names of all other relationships that it remains unambiguous and easily comprehended in all contexts (*e.g.* in a Define Calculation dialog box containing multiple related fields across multiple relationships).
2. Whenever possible, the name of each relationship shall be reasonably descriptive of the function of the relationship.
3. Every relationship shall be documented in a location that is easily accessible by all persons who require information about the relationship definition.

### Guidelines

1. [No guidelines have been set for this item.]

# Naming Conventions

---

## Relationships (Cont'd)

### [Fallback] Standards

1. Only the lowercase and uppercase alphabet, the numeric digits (0...9), the underscore character (“\_”), and the pound (“#”) shall be used in relationship names.
2. Relationships joining a file to an external file shall be named with the Target File in Title case), followed by the pound (“#”), followed by the name of the initiating key, followed by the pound (“#”), followed by the name of the target key; i.e. TargetFile#InitiatingKey#TargetKey (e.g. Letters#zk\_contactID\_p#zk\_contactID\_f).
3. The names of relationships joining a file to itself (i.e. a “self-join” relationship) shall be named according to the rules for external files (*q.v.* §2, above), where the “Target File” is replaced with the word “Self” (in title case (e.g. Self#zk\_nameLast\_g#zk\_nameLast\_f).

### Guidelines

1. As long as all three elements of the relationship (*i.e.* the target file, the initiating key field, and the target key field) remain unambiguous, relationship names should be made as short as possible by:
  - a. omitting the suffix from the initiating key, the target key, or both keys, if there are no similar field names in the respective files (e.g. PhoneMain#zk\_phone#zk\_phone instead of PhoneMain#zk\_phone\_a#zk\_phone\_f;
  - b. omitting the name of the target key and the preceding pound (“#”) if the names of the initiating and target keys are identical, or the difference is arbitrarily small (e.g. Contact#contactID instead of Contact#contactID#contactID);
  - c. substituting obvious abbreviations for file and field names (e.g. Con#conID is appropriate unless the file set includes two files that begin with “Con”, in which case Ctct#conID might be more appropriate).

# Naming Conventions

---

## Value Lists

### Standards

1. Only the lowercase and uppercase alphabet, the numeric digits (0...9), the underscore character (“\_”), and the pound (“#”) shall be used in value list names.
2. Suffixes shall be appended to value list names according to the following conventions:
  - a. all suffixes shall begin with a single underscore (“\_”);
  - b. Custom value lists in the current file will be designated by “c”;
  - c. Value lists which reference values from a field(s) will be designated by “f”;
  - d. Value lists which reference value lists from another file will be designated by “v”;
3. Value lists which reference field values or value lists in another file shall be appended by a pound sign and the name of the file (e.g. Country\_v#Default)

# Naming Conventions

---

## Scripts

### Standards

1. Only the standard “ASCII 32-126” character set shall be used in script names.

### Guidelines

1. Script names — especially those visible to the end-user — should contain spaces and other formatting elements to increase comprehensibility.
2. Script names should begin with an indicator descriptive of the function or triggering mechanism of the script. Some examples would be as follows:
  - a. “BTN: ” for scripts attached to buttons on layouts;
  - b. “RPT: “ for scripts that print reports;
  - c. “PRT: “ for scripts that execute Print Setup formatting steps;
  - d. “SRT: “ for scripts that perform Sort routines;
  - e. “FND: “ for scripts that perform Find routines;
  - f. “IMP: “ for scripts that perform Import routines;
  - g. “EXP: “ for scripts that perform Export routines;
  - h. “NAV: “ for scripts that navigate (use “NAVX: “ for scripts that navigate to an external file);
  - i. “DEV: “ for scripts used by the developer for “housekeeping” or other programming purposes;
  - j. “OPS: “ for scripts specifically used for database operations (e.g. “OPS: Startup Script”);
  - j. “INT: “ for scripts which control user interface aspects;
  - k. “FILE: “ for scripts which perform file opening or saving routines;
  - l. “REC: “ for scripts which manipulate individual records (use “RECX: “ for records in external files);
  - m. “SUB: “ for subscripts (*i.e.* scripts that are called only from other scripts).
3. The names of multipart scripts should contain an appropriate indicator (e.g. SUB: Create Invoice Part 2”).
4. The names of scripts that contain an unconditional “Halt Script” step should have an appropriate indicator (e.g. “SUB: No Records Found [HALT]”).

# Naming Conventions

---

## Scripts (Cont'd)

5. The names of scripts attached to buttons that are formatted to do something other than “Exit” the current script (*e.g.* buttons in a dialog would likely be formatted to “Pause” or “Resume” the current script) should contain an appropriate indicator (*e.g.* “BTN: Get CustomerID [PAUSE]”).
6. Scripts should be arranged into groups and ordered according to an apparent and logical scheme.
7. For the purpose of grouping scripts (*i.e.* in the ScriptMaker™ dialog), “separator” scripts should be created with names that adhere to the following convention;

```
-  
----- SCRIPT NAME -----
```

*i.e.* 4-5 dashes on either side of the words. A script named as “-” (*i.e.* a dash) immediately before the beginning and just after the end of each section to separate the section from the scripts that follow.

These scripts should contain comments that describe the respective script group.

# Standard Fields

---

## Overview

The fields listed here are designated as standard fields that should be used in the development of all databases developed. For ease of implementation, these standard fields should be incorporated into a set of Start files.

## List of Fields

zc\_createdDate

**Field Type:** date

**Options:** auto-enter Created Date, prohibit modification

**Value:** (auto-entered)

zc\_createdName

**Field Type:** text

**Options:** auto-enter Creator Name, prohibit modification

**Value:** (auto-entered)

*n.b.* if available, a user name from a login system may be used here instead (*i.e.* set via a script or lookup).

zc\_createdTime

**Field Type:** time

**Options:** auto-enter Created Time, prohibit modification

**Value:** (auto-entered)

zc\_currentRecNo\_gn

**Field Type:** global [number]

**Options:** [n/a]

**Use:** used in the save found set process to store the record number of the current record while the values in zc\_recordSerial are copied to the clipboard.

zc\_currentRecNoMark\_gn

**Field Type:** global [number]

**Options:** [n/a]

**Use:** used in the mark records process to store the record number of the current record while the zc\_recordSerial fields are copied to the clipboard.

## Standard Fields

---

### List of Fields (Cont'd)

zc\_mark\_gt

**Field Type:** global [text]

**Options:** [n/a]

**Use:** used in the mark records process to store marked records' zc\_recordSerial separated by returns.

zc\_markRecord\_u

**Field Type:** calculation [text]

**Options:** Case(PatternCount(zc.mark.g, zc.recordSerial), "Y")

**Use:** used in the mark records process to display to the user that the record has been marked.

zc\_modifiedDate

**Field Type:** date

**Options:** auto-enter Modification Date, prohibit modification

**Value:** (auto-entered)

zc\_modifiedName

**Field Type:** text

**Options:** auto-enter Modifier Name, prohibit modification

**Value:** (auto-entered)

*n.b.* if available, a user name from a login system may be used here instead (*i.e.* set via a script or lookup).

zc\_modifiedTime

**Field Type:** time

**Options:** auto-enter Modification Time, prohibit modification

**Value:** (auto-entered)

zc\_recCount\_u

**Field Type:** calculation [number]

**Options:** unstored

**Value:** Status(CurrentRecordCount)

## Standard Fields

---

### List of Fields (Cont'd)

zc\_recFound\_u

**Field Type:** calculation [number]

**Options:** unstored

**Value:** Status(CurrentFoundCount)

zc\_recNumber\_u

**Field Type:** calculation [number]

**Options:** unstored

**Value:** Status(CurrentRecordNumber)

zc\_recordSerial

**Field Type:** Text

**Options:** Auto Enter Serial Number, Prohibit modification of value

zc\_userCurrentName\_u

**Field Type:** calculation [text]

**Options:** unstored

**Value:** Status(CurrentUserName)

*n.b.* if available, a user name from a login system may be used here instead.

zi\_recInfoShow\_u

**Field Type:** calculation [text]

**Options:** unstored

**Calculation (example):**

"Record " & z\_recNumber\_u & " of " & zc\_recFound\_u & " Found ... "  
& zc\_recCount\_u & " Record" & Case(zc\_recCount\_u > 1, "s") & "  
Total"

**Display (example):**

Record 4 of 25 Found ... 193 Records Total

## Standard Fields

---

### List of Fields (Cont'd)

zi\_userMsg\_gt

**Field Type:** global [text]

**Options:** [n/a]

**Use:** store message text (e.g. for use in dialogs)

zk\_constant\_a

**Field Type:** calculation [text]

**Options:** indexed

**Value:** for security reasons, even though every file in each solution will share the same “solution key”, this value should be different for each solution – any reasonably long random string of alphanumeric digits (e.g. V2WD117K) is sufficient for the purpose

zv\_counter\_gnr

**Field Type:** global [number]

**Options:** 5 repetitions standard

**Use:** Incremental counters in looping scripts, with repetitions used for nested loops.

zv\_error\_gn

**Field Type:** global [number]

**Options:** [n/a]

**Use:** store Status(CurrentError) value

zv\_tempNum\_gnr

**Field Type:** global [number]

**Options:** 5 repetitions standard

**Use:** store temporary number values

zv\_tempText\_gtr

**Field Type:** global [text]

**Options:** 5 repetitions standard

**Use:** store temporary text values

# Layouts

---

## General

1. Where appropriate, the status area should be locked.
2. Every solution should contain at least one detail (form) layout, one list layout and one Table layout (if using FileMaker Pro 5).

## Organization

### Standards

1. [No standards have been set for this item.]

### Guidelines

1. Layouts should be arranged into groups according to an apparent and logical organization scheme.
2. Layout groups and individual layouts within those groups should be arranged according to an apparent and logical ordering scheme.
3. Layout groups should be separated with a blank layout named "-".  
(*n.b.* this will place a divider between the layout groups in menus).

## Consistency

### Standards

1. [No standards have been set for this item.]

### Guidelines

1. Layouts should be consistent across a solution.
2. Buttons should be placed in a consistent manner on similar type layouts (e.g. buttons on list layouts would always appear in the same location).
3. Portals should be consistent in appearance and behaviour, and should exhibit consistent properties.
4. Reports should appear consistent in design (*q.v.* Reports section for more information).

# Layouts

---

## Cross Platform Considerations

### Standards

1. [No standards have been set for this item.]

### Guidelines

1. Layouts should be designed so that all elements (*e.g.* fonts, colours, screen sizes, field widths, *etc.*) appear in an acceptable fashion on all machines in a cross-platform environment. The system specifications should also be followed when implementing this guideline.

## Navigation

### Standards

1. [No standards have been set for this item.]

### Guidelines

1. File, layout, and record navigation should be consistent in all files.
2. Navigation should leave the user on an appropriate layout, according to the following guidelines:
  - a. if the file is being opened for the first time in a particular (login) session, navigation should leave the user on the primary or most commonly accessed layout — quite often this would be an overview (list) layout;
  - b. if the file is being re-entered as a result of round-trip navigation (*e.g.* having navigated previously to a distinct location), navigation should leave the user on the layout from which the navigation originated.
3. Whenever possible, round-trip navigation should leave the user within the same found set, and viewing the same record from which the navigation originated.

# Layouts

---

## Fonts

### Standards

1. [No standards have been set for this item.]

### Guidelines

1. No more than three (3) fonts should appear in any one layout.
2. Unless there is a compelling reason not to, we recommend that Arial 11pt or Verdana 11pt be used as a standard font for fields. If the solution is to be used in a cross platform environment this will produce the most consistent results on both platforms.

## Buttons

### Standards

1. [No standards have been set for this item.]

### Guidelines

1. Buttons should be named appropriately and consistently.
2. Buttons should be placed in a consistent manner on similar type layouts (e.g. buttons should always appear in the same location on list layouts).
3. Whenever appropriate, buttons should be available to display the child or parent record(s) of the current record; whenever possible, the entire portal row should act as the button to display the related record.
4. Buttons should be available to perform standard database operations (e.g. Find, Find All, New, Delete, Sort, Print).

# Layouts

---

## Portals

### Standards

1. No portal shall be formatted to allow the manual creation of related records — *i.e.* all related record creation shall be done via scripts which are called by a button on or near the portal.

### Guidelines

1. Portals should be consistent in appearance and behaviour, and should exhibit consistent properties.

## Data Entry Layouts

### Standards

1. [No standards have been set for this item.]

### Guidelines

1. Current record information (e.g. total records in the database, number of records in found set, and the record number in the found set) should always be displayed.
2. Buttons should be used to navigate to list layouts.

# Layouts

---

## List Layouts

### Standards

1. [No standards have been set for this item.]

### Guidelines

1. No fields in a list view should allow entry or editing unless otherwise specified in the system specifications.
2. The number of records in the current found set should always be displayed.
3. Column labels should contain sort buttons where appropriate.
4. When appropriate, buttons which perform the following functions should be displayed:
  - a. mark records;
  - b. unmark records;
  - c. find records;
  - d. find all records;
  - e. view record detail (*i.e.* on a data entry layout);

## Table View

### Standards

1. [No standards have been set for this item.]

### Guidelines

1. Table View should only be available for layouts designed specifically to be viewed as a table.
2. Table View Properties should be set to always include Headers and Footers and columns should always be resizable and reorderable.

# Layouts

---

## Reports

### Standards

1. [No standards have been set for this item.]

### Guidelines

1. Reports should appear consistent in design.
2. Layouts which have printable reports should be formatted (in the Layout Setup dialog) with one-half inch (0.5") fixed margins.
3. Reports may adhere to some or all of the following guidelines:
  - a. the name and address of the client should appear in bold at the top left of the header section;
  - b. the title of the report should appear in bold in the center of the header section
  - c. the date of the report should appear in bold at the top right of the header section;
  - d. column labels should appear in bold in the appropriate place in the header section;
  - e. a horizontal line should appear just below the column labels in the header section;
  - f. a horizontal line should appear at the top of the footer section;
  - g. the page number of the report should appear at the center of the footer section.
  - h. within budget and interface constraints, reports should be previewed using a floating 'palette' file.

## Find Layouts

### Standards

1. [No standards have been set for this item.]

### Guidelines

1. Within budget and interface constraints, the system should contain layouts specifically designed for performing finds.

# Layouts

---

## Developer Layouts

### Standards

1. Every database shall have a layout named 'Dev\_Fields' that contains all fields referenced by scripts which incorporate any of the following ScriptMaker™ steps: Copy; Paste; Replace.

### Guidelines

1. Any layouts used by the developer for internal processes should be prefixed with "Dev\_" and should be formatted so that they are not visible in the Layouts menu while in Browse mode.
2. All layouts that are referenced by the Copy Record and Copy All Records script steps should be clearly marked. (e.g. 'Dev\_Copy\_CustID' ).

## Dialogs

### Standards

1. The Troi Dialog External Plug-In shall be used for all dialog needs that cannot be fulfilled by the native FileMaker Pro Show Message script step (*i.e.* layouts that mimic dialogs shall not be used).

### Guidelines

1. The Troi Dialog External Plug-In should be used for all dialog needs, including those that can be fulfilled by the native FileMaker Pro Show Message script step.
2. Unnecessarily negative or restrictive wording should be avoided (e.g. "abort", "crash", "fatal", "beyond repair", etc.).
3. Dialog text should be as detailed and descriptive as possible (e.g. "Need more info." is not as helpful as "Please ensure that the full name has been entered."; "General Protection Fault FOOBAR269" is not as helpful as "Because of a lack of physical RAM, the computer ran out of memory, causing a stack heap error.").

# Scripting

---

## Organization

### Standards

1. [No standards have been set for this item.]

### Guidelines

2. Scripts should be arranged into groups and ordered according to an apparent and logical scheme.

## Opening Scripts

### Standards

1. [No standards have been set for this item.]

### Guidelines

1. All databases should perform a script upon opening which brings the user to the appropriate (*e.g.* primary) layout. The first script step in this opening script should always call the opening script in the “Main” or “Primary” file of the system. This will ensure system housekeeping steps in the “Main” opening script will always execute regardless of what file in the system is opened first.

## Browse Mode

### Standards

1. Every script that is not to be used in Find Mode shall begin with an explicit “Enter Browse Mode” step.

If it is necessary to reuse a script that is to be used in Find Mode, a duplicate script shall be created to which an explicit “Enter Browse Mode” step shall be added, and appropriate indicators shall be appended to the names of both scripts.

### Guidelines

1. [No guidelines have been set for this item.]

# Scripting

---

## Find Mode

### Standards

1. [No standards have been set for this item.]

### Guidelines

1. When defining an “omit” request in Find Mode, the Status Bar must be unlocked.

## Restoring Options

### Standards

1. Script steps that require the restoring of options (*i.e.* Find, Sort, Page Setup, Import, Export) shall be placed into separate, unambiguously named scripts, and the options being restored shall be commented thoroughly in the body of the script. The only exception to this is when an “Enter Find Mode [Restore]” step is required and the currently running script needs to be paused. Comments are still required in the body of the script.

### Guidelines

1. [No guidelines have been set for this item.]

## Dependencies

### Standards

1. [No standards have been set for this item.]

### Guidelines

1. Scripts that rely on a specific layout should be used with caution, and the name and description of the target layout should be placed in a comment step directly above or below the Go To Layout step.
2. When using the Copy, Paste and Replace steps, the script should navigate to a developer layout which contains all the appropriate fields (*e.g.* Dev\_Fields).

*(n.b.* Ensure that the field labels do not overwrite the fields themselves.)

# Scripting

---

## Previewing

### Standards

1. Unless explicitly demanded by the client, the Status Bar shall not be displayed (even during Preview operations).

### Guidelines

1. Alternatives Preview operation interfaces can be found in the startup file set.

## New Related Records

### Standards

1. No portal shall be formatted to allow the manual creation of related records — *i.e.* all related record creation shall be done via scripts which are called by a button on or near the portal.

### Guidelines

1. The most efficient method of related record creation is the following:
  - a. set a temp field in a central (*e.g.* Default) file equal to the appropriate key (*e.g.* zk\_companyID\_p);
  - b. run a script in the target file (*i.e.* an external script from the parent file) which creates a new record in the target file;
  - c. set the key in the new record (*e.g.* zk\_companyID\_f) to the value stored in the temp field in the central file.

## Script Steps To Avoid

### Standards

1. Unless there is no other option available avoid use of the following script steps; Copy, Cut, Paste and Clear.

# Preferences

---

## General

### Standards

1. [No standards have been set for this item.]

### Guidelines

1. The 'Enable Drag and Drop' option should always be on.
2. The 'Add newly defined fields to layout' option should always be unselected.
3. The 'Smart Quotes' option should always be off.
4. The 'Store Compatible Graphics' option should usually be on.

# Plug Ins

---

## Recommended Plug Ins

### Standards

1. [No standards have been set for this item.]

### Guidelines

1. The following plug-ins should be used in all development.

- a. Troi Dialog

This plug-in adds dynamic dialogs functions to FileMaker Pro. With it you can create dynamic dialog text (and buttons!) and progress bars on the fly by using a calculation.

- b. Troi File

Adds File Manipulation: including saving or reading a field from or to a file. Other file manipulations are also possible.

- c. SecureFM

This plug-in allows you to remove or disable FileMaker Pro menus and commands.

# Start Databases

---

## Introduction

### Purpose

1. To increase efficiency, ensure consistency, and reduce development time, a number of “starter” database sets have been created.
2. Each set includes examples of the main types of databases that a development team would need to generate a solution, including:
  - a. login and security files;
  - b. “menu” files;
  - c. utility files;
  - d. standard (general-use template) files; and,
  - e. Default and preferences files.
3. The Start files will be available on the corporate FTP site and on a Developer Toolkit CD.

## Start Databases

---

### Main.FP5

#### Standards

1. The Main file shall include a “Splash” layout (*i.e.* a graphic which displays at the time of initial application launch) showing at least the following information: our company logo; any application-specific graphics, logo, or title information; the current application version number; and, the date of the last install.
2. The Main file shall include an “About” layout (*i.e.* a layout containing a description of the solution and company information), showing at least the following information: company contact information; and, the name(s) of all developer(s) that were involved in the programming of the solution.

#### Guidelines

1. The Main file may contain a central menu system from where the user can navigate to other databases in the solution.

### Developer.FP5

#### Standards

1. The Developer file template is used as a basis for development of any file with which the typical user will interact.

#### Guidelines

1. [No guidelines have been set for this item.]

## Start Databases

---

### Utility.FP5

#### Standards

1. The Utility file template is used as a basis for development of any file with which the typical user will not interact (e.g. lookup tables, child table which is viewable only through portals or related fields in the parent table, join files, etc.).

#### Guidelines

1. All files developed from the Utility file template should be locked down to restrict access by the typical user (e.g. the status area should be locked and no layouts should appear in the layout menu).

### Default.FP5

#### Standards

1. Should be used to store information that can be accessible by every database in the solution being developed (e.g. common graphics, organization name and address, tax numbers, etc.).
2. Whenever possible, the Default file shall be named "Default".
3. The Default file shall have a constant relationship to all other files in the solution.

#### Guidelines

1. The Default file should have only one record.
2. The Default file should contain the standard fields, relationships, layouts, and scripts as displayed in the standard Default starter file.

# Developer Toolkit

---

## Toolkit Contents

### Graphics Library

1. A variety of buttons and icons.
2. Solutions should use standard graphic elements from the Corepix Collection.

### User Interfaces

1. Includes a variety of user interface designs.
2. Screen Sizes. Standard Mac and Windows with and without the Status Bar.
3. Standard Fonts, e.g. Bullet fonts.

### User Feedback Database

1. Every solution should have a mechanism for managing client and developer feedback (*e.g.* feature requests, bug reports, *etc.*).

### Drop Rename (Mac Only)

1. This is a very useful application that allows you to quickly rename a file or group of files. For example, you could add an asterisk "\*" to the beginning of each file name ensuring an older file is not mistakenly opened by another file through a relationship or script.

# Developer Toolkit

---

## Toolkit Future Additions

The following resources will be added to the next release of the Developer Toolkit.

### Screen Sizes

Standard Macintosh and Windows screen sizes with and without the Status Bar and toolbar.

### Standard Fonts

Standard Macintosh and Windows fonts, including Bullet fonts.

### Scripts & Calculations

A database of scripts and special calculations, e.g. for licensed plug-ins.

### Login System

A secure log-in system may be made available to clients for an additional cost. A log-in system will be provided on the next release.

### States/Provinces

Contains the names of all states (United States) and provinces (Canada) and can be made accessible from a central lookup file (e.g. Default).

### Zip Codes/Postal Codes

Contains the United States Zip Codes and Canadian Postal Codes and can be made accessible from a central lookup file (e.g. Default).

# Developer Edition

---

## Runtime Solutions

### Standards

1. All runtime solutions shall include a button to recover (*q.v.* Guidelines §1, below).

### Guidelines

1. The Recover button and script (*q.v.* Standards §1, above) should be in the Default database, or another file that is both central and universally accessible to users (or solution administrators, if applicable).

## Dates/Y2K

---

### Standards

1. All date fields shall be formatted to validate as four-digit year.
2. All objects that reference dates (*e.g.* project folders) shall use four-digit years.

### Guidelines

1. Unless space is at a premium, all date fields should be formatted to display the year using four digits.

# Web Development Standards

---

## Web Development

### File Names

1. Only the lowercase and uppercase alphabet, the numeric digits (0...9), the underscore character (“\_”), and one period (“.”, to begin the extension) shall be used in file names.
2. The extension shall be limited to 3 if there is a choice. For example “.htm” instead of “.html”. In the case of Lasso files, this shall remain as “.lasso” because it is the prevailing standard.
3. The default index file in a folder shall be prefixed “default”. For example default.htm, default.php, default.lasso.
4. Title case shall be used to distinguish words as defined in *Naming Conventions / Files / Guidelines* with some differences: The first character shall be lower case and the length is not limited to 31 characters. Underscores should be avoided and only used in exceptional circumstances.

### Variable Names

1. Variable names shall be prefixed with three characters. The first character will define the scope and the next two shall define the type. These characters shall be lower case.
2. The scope of the variable can be local to the page or global to the application. Local variables are prefixed with “v”, global variables are prefixed with “g”.
3. The suffixes used for the type of the variable are:
  - tx text, character
  - in integer
  - dt date, time
  - dl decimal, double, currency
  - ob object
  - ar array
  - ss session

### Constant Names

1. Constant names shall use the same convention as variable names above, but the prefixes shall be different.
2. A local constant shall use “c”, and global constants shall use “a” (taken from ‘application’).

# Web Development Standards

---

## Function Names

1. [No guidelines have been set for this item.]

## Guidelines

	<b>Local Variable</b>	<b>Global Variable</b>	<b>Local Constant</b>	<b>Global Constant</b>
<b>Text</b>	vtxVariable	gtxVariable	ctxConstant	atxConstant
<b>Integer</b>	vinVariable	ginVariable	cinConstant	ainConstant
<b>Date, Time</b>	vdtVariable	gdtVariable	cdtConstant	adtConstant
<b>Decimal</b>	vdIVariable	gdIVariable	cdIConstant	adIConstant
<b>Object</b>	vobVariable	gobVariable	cobConstant	cssConstant
<b>Session</b>	vssVariable	gssVariable	cssConstant	assConstant

## Notes

---