Notion: Notes for the module and patch writer

The Notion Team, `http://notion.sourceforge.net`
Tuomo Valkonen, tuomov at iki.fi

March 9, 2016

The latest version of this document can be found at `http://notion.sourceforge.net` in HTML and PDF formats. Updates are welcome, preferably in the form of patches against the LaTeX sources which can be found in the git repository at `git://notion.git.sourceforge.net/gitroot/notion/notion-doc` (`http://notion.git.sourceforge.net/git/gitweb.cgi?p=notion/notion-doc;a=summary`)

Abstract

This document is an unorganized collection of notes for those who want to write modules or patches to Notion.

Contents

1 Class and object hierarchies

While Notion does not have a truly object-oriented design[1], things that appear on the computer screen are, however, quite naturally expressed as such "objects". Therefore Notion implements a rather primitive OO system for these screen objects and some other things.

It is essential for the module writer to learn this object system, but also people who write their own binding configuration files necessarily come into contact with the class and object hierarchies – you need to know which binding setup routines apply where, and what functions can be used as handlers in which bindings. It is the purpose of this section to attempt to explain these hierarchies. If you do not wish the read the full section, at least read the summary at the end of it, so that you understand the very basic relations.

For simplicity we consider only the essential-for-basic-configuration Notioncore, *mod_tiling* and *mod_query* classes. See Appendix B for the full class hierarchy visible to Lua side.

1. the author doesn't like such artificial designs

1.1 Class hierarchy

One of the most important principles of object-oriented design methodology is inheritance; roughly how classes (objects are instances of classes) extend on others' features. Inheritance gives rise to class hierarchy. In the case of single-inheritance this hierarchy can be expressed as a tree where the class at the root is inherited by all others below it and so on. The tree below lists out the Notion class hierarchy and below we explain what features of Notion the classes implement.

1.1.1 Partial Notioncore, *mod_tiling* and *mod_query* class hierarchy

```
Obj
 |-->WRegion
 |     |-->WClientWin
 |     |-->WWindow
 |     |     |-->WRootWin
 |     |     |-->WMPlex
 |     |     |     |-->WFrame
 |     |     |     `-->WScreen
 |     |     `-->WInput (mod_query)
 |     |           |-->WEdln (mod_query)
 |     |           `-->WMessage (mod_query)
 |     |-->WGroup
 |     |     |-->WGroupWS
 |     |     `-->WGroupCW
 |     `-->WTiling (mod_tiling)
 `-->WSplit (mod_tiling)
```

1.1.2 The core classes

Obj Is the base of Notion's object system.

WRegion is the base class for everything corresponding to something on the screen. Each object of type WRegion has a size and position relative to the parent WRegion. While a big part of Notion operates on these instead of more specialised classes, WRegion is a "virtual" base class in that there are no objects of "pure" type WRegion; all concrete regions are objects of some class that inherits WRegion.

WClientWin is a class for client window objects, the objects that window managers are supposed to manage.

WWindow is the base class for all internal objects having an X window associated to them (WClientWins also have X windows associated to them).

WMPlex is a base class for all regions that "multiplex" other regions. This means that of the regions managed by the multiplexer, only one can be displayed at a time.

WScreen is an instance of WMPlex for screens.

WRootWin is the class for root windows of X screens. Note that an "X screen" or root window is not necessarily a single physical screen as a root window may be split over multiple screens when techniques such as Xinerama are used. (Actually there can be only one root window when Xinerama is used.)

WFrame is the class for frames. While most Notion's objects have no graphical presentation, frames basically add to WMPlexes the decorations around client windows (borders, tabs).

WGroup is the base class for groups. Particular types of groups are workspaces (WGroupWS) and groups of client windows (WGroupCW).

Classes implemented by the *mod_tiling* module:

WTiling is the class for tilings of frames.

WSplit (or, more specifically, classes that inherit it) encode the WTiling tree structure.

Classes implemented by the *mod_query* module:

WInput is a virtual base class for the two classes below.

WEdln is the class for the "queries", the text inputs that usually appear at bottoms of frames and sometimes screens. Queries are the functional equivalent of "mini buffers" in many text editors.

WMessage implements the boxes for warning and other messages that Notion may wish to display to the user. These also usually appear at bottoms of frames.

There are also some other "proxy" classes that do not refer to objects on the screen. The only important one of these for basic configuration is WMoveresMode that is used for binding callbacks in the move and resize mode.

1.1.3 Run-time access to types

Even though it often indicates a design mistake, sometimes it can be useful to have run-time access to the types of objects.

For example, to check wether a given object is of type WMPlex, the following C code can be used:

```
if(obj_is((Obj*)cwin, &CLASSDESCR(WMPlex)))
    ....
```

Its lua counterpart is:

```
if(obj_is(cwin, "WMPlex"))
    ....
```

While there's also an 'obj_cast' method available, C structs can be freely cast to their 'superclass' using a regular C cast, for example:

```
bool input_fitrep(WInput *input, WWindow *par, const WFitParams *fp)
{
    if(par!=NULL && !region_same_rootwin((WRegion*)input, (WRegion*)par))
        return FALSE;
    ...
```

1.2 Object hierarchies: WRegion parents and managers

1.2.1 Parent–child relations

Each object of type WRegion has a parent and possibly a manager associated to it. The parent for an object is always a WWindow and for WRegion with an X window (WClientWin, WWindow) the parent WWindow is given by the same relation of the X windows. For other WRegions the relation is not as clear. There is generally very few restrictions other than the above on the parent—child relation but the most common is as described in paragraph 1.2.1.

Most common parent–child relations

```
WRootWins
 |-->WGroupWSs
 |-->WTilings
 |-->WClientWins in full screen mode
 `-->WFrames
      |-->WGroupCWs
      |-->WClientWins
      |-->WFrames for transients
      `-->a possible WEdln or WMessage
```

1.2.2 Manager–managed relations

WRegions have very little control over their children as a parent. The manager WRegion has much more control over its managed WRegions. Managers, for example, handle resize requests, focusing and displaying of the managed regions. Indeed the manager—managed relationship gives a better picture of the logical ordering of objects on the screen. Again, there are generally few limits, but the most common hierarchy is given in Figure **??**. Note that sometimes the parent and manager are the same object and not all regions may have a manager, but all non-screen regions have a parent—a screen if not anything else.

Most common manager–managed relations

```
WRootWins
 |-->WGroupCWs for full screen WClientWins
 |    |-->WClientWins
 |    `-->WFrames for transients (dialogs)
 |         `--> WClientWin
 |-->WGroupWSs for workspaces
 |    |-->WTiling
 |    |    |-->WFrames
 |    |    |    `-->WGroupCWs (with contents as above)
 |    |    `-->possibly a WStatusBar or WDock
 |    |-->WFrames for floating content
 |    |-->possibly a WEdln, WMessage or WMenu
 |    `-->possibly a WStatusBar or WDock (if no tiling)
 `-->WFrames for sticky stuff, such as the scratchpad
```

Note that a workspace can manage another workspace. This can be achieved with the `WGroup.attach_new` function, and allows you to nest workspaces as deep as you want.

In the presense of Xinerama there may be WScreen objects at the top of the manager–managed tree instead of WRootWin.

1.3 Summary

In the standard setup, keeping queries, messages and menus out of consideration:

- The top-level objects that matter are screens and they correspond to physical screens. The class for screens is WScreen.

- Screens contain (multiplex) groups (WGroup) and other objects, such as WFrames. Some of these are mutually exclusive to be viewed at a time.
- Groups of the specific kind WGroupWS often contain a WTiling tiling for tiling frames (WFrame), but groups may also directly contain floating frames.
- Frames are the objects with decorations such as tabs and borders. Frames contain (multiplex) among others (groups of) client windows, to each of which corresponds a tab in the frame's decoration. Only one client window (or other object) can be shown at a time in each frame. The class for client windows is WClientWin.

2 Object system implementation

First, to get things clear, what are considered objects here are C structures containing a properly initialised structure defined in *ioncore/obj.h* as the first element (or the first element of the structure which is the first element and so on which gives rise to inheritance). The WObj structure contains a pointer to a WObjDescr class type info structure and a list of so called "watches". The WObjDescr structure simply lists the class name, a table of dynamic functions and a pointer to deinitialisation function (or "destructor").

2.1 Object creation

Object instances are created using the `CREATEOBJ_IMPL` macro, but by convention for each class `WFoo` a construction method `create_foo` is provided.

2.2 Safe references

NOTE: This subsection appears out-of-date, it seems the WWatch infrastructure was removed from Ion3.

Notion does not do any reference counting, garbage collecting or other fancy things related to automatic safe freeing of objects with its simplistic object system. Instead special watches (the WWatch structure) may be used to create safe references to objects that might be destroyed during the time the specific pointer is needed. When an object is destroyed, its list of watches is processed, setting the pointers in the watches to NULL and the watch handlers for each watch are called.

2.3 Dynamic dispatch

To override a method means to reimplement it specifically for a given subtype. This is used in Notion to allow subclasses to override of their superclasses' methods.

Overridable methods are specified as `DYNFUN`, for example `region_fitrep` in *ioncore/region.h*:

```
DYNFUN bool region_fitrep(WRegion *reg, WWindow *par, const WFitParams *fp);
```

Dynamic functions can be called with `CALL_DYN` or `CALL_DYN_RET`. When a subclass desires to override a dynamic function, it does so by specifying a DynFunTab. For example, the function table for WGroup overrides `region_fitrep` with `group_fitrep`:

```
static DynFunTab group_dynfuntab[]={
    {(DynFun*)region_fitrep,
     (DynFun*)group_fitrep},
```

```
    ...
    END_DYNFUNTAB
};
EXTL_EXPORT
IMPLCLASS(WGroup, WRegion, group_deinit, group_dynfuntab);
```

Whenever `region_fitrep` is called on a region that is actually a WGroup, `group_fitrep` is called instead.

3 The Lua interface

This section finally describes the implementation details and how modules should us the Lua interface. First, in section 3.1 we look at types supported by the interface, how objects are passed to Lua code and how Lua tables should be accessed from Notion and modules. In section 3.2 the methods for exporting functions and how they are called from Lua are explained and in section 3.3 the method for calling Lua functions is explained.

3.1 Supported types

The following types are supported in passing parameters between the C side of Notion and Lua:

Identifier character	C type	Description
i	int	Integer
s	char*	String
S	const char*	Constant string
d	double	
b	bool	
t	ExtlTab	Reference to Lua table
f	ExtlFn	Reference to Lua function.
o	Any WObj*	

The difference between identifiers 's' and 'S' is that constant strings as return values are not free'd by the level 1 call handler (see below) after passing to Lua (`lua_pushstring` always makes a copy) unlike normal strings. String parameters are always assumed to be the property of the caller and thus implicitly const.

Likewise, if a reference to 't' or 'f' is wished to be stored beyond the lifetime of a function receiving such as an argument, a new reference should be created with `extl_ref_table/fn`. References can be free'd with `extl_unref_table/fn`. References gotten as return values with the `extl_table_get` (how these work should be self-explanatory!) functions are property of the caller and should be unreferenced with the above-mentioned functions when no longer needed. The functions `extl_fn/table_none()` return the equivalent of NULL.

WObjs are passed to Lua code with WWatch userdatas pointing to them so the objects can be safely deleted although Lua code might still be referencing them. (This is why SWIG or tolua would not have helped in creating the interface: extra wrappers for each function would still have been needed to nicely integrate into Notion's object system. Even in the case that Notion was written in C++ this would be so unless extra bloat adding pointer-like objects were used everywhere instead of pointers.) It may be sometimes necessary check in Lua code that a value known to be an Notion WObj is of certain type. This can

be accomplished with `obj_is(obj, "typename")`. To find the type name for a WObj, use `obj_typename(obj)`.

3.2 Exporting functions

Exported functions (those available to the extension language) are defined by placing `EXTL_EXPORT` before the function implementation in the C source. The script mkexports.pl is then used to automatically generate *exports.c* from the source files if `MAKE_EXPORTS=modulename` is specified in the Makefile. All pointers with type beginning with a 'W' are assumed to be pointers to something inheriting WObj. In addition to a table of exported functions and second level call handlers for these, *exports.c* will contain two functions `module_register_exports()` and `module_unregister_exports()` that should then be called in module initialisation and deinitialisation code.

You've seen the terms level 1 and 2 call handler mentioned above. The Lua support code uses two so called call handlers to convert and check the types of parameters passed from Lua to C and back to Lua. The first one of these call handlers is the same for all exported functions and indeed lua sees all exported as the same C function (the L1 call handler) but with different upvalues passing a structure describing the actual function and the second level call handler. The L1 call handler checks that the parameters received from Lua match a template given as a string of the identifier characters defined above. If everything checks out ok, the parameters are then put in an array of C unions that can contain anyof these known types and the L2 call handler is called.

The L2 call handler (which is automatically generated by the mkexports.pl script) for each exported function checks that the passed WObjs are of the more refined type required by the function and then calls the actual function. While the WObj checking could be done in the L1 handler too, the L2 call handlers are needed because we may not know how the target platform passes each parameter type to the called function. Thefore we must let the C compiler generate the code to convert from a simple and known enough parameter passing method (the unions) to the actual parameter passing method. When the called function returns everything is done in reverse order for return values (only one return value is supported by the generated L2 call handlers).

3.3 Calling Lua functions and code

The functions `extl_call`, `extl_call_named`, `extl_dofile` and `extl_dostring` call a referenced function (ExtlFn), named function, execute a string and a file, respectively. The rest of the parameters for all these functions are similar. The 'spec' argument is a string of identifier characters (see above) describing the parameters to be passed. These parameters follow after 'rspec'. For dofile and dostring these parameters are passed in the global table arg (same as used for program command lien parameters) and for functions as you might expect. The parameter 'rspec' is a similar description of return values. Pointers to variables that should be set to the return values follow after the input values. The return value of all these functions tells if the call and parameter passing succeeded or not.

Sometimes it is necessary to block calls to all but a limited set of Notion functions. This can be accomplished with `extl_set_safelist`. The parameter to this function is a NULL-terminated array of strings and the return value is a similar old safelist. The call `extl_set_safelist(NULL)` removes any safelist and allows calls to all exported functions.

3.4 Miscellaneous notes

Configuration files should be read as before with the function `read_config_for` except that the list of known options is no longer present.

Winprops are now stored in Lua tables and can contain arbitrary properties. The 'proptab' entry in each WClientWin is a reference to a winprop table or `extl_table_none()` if such does not exist and properties may be read with the `extl_table_gets` functions. (It is perfectly legal to pass `extl_table_none()` references to `extl_table_get*`.)

4 Modules

4.1 Anatomy

A Notion module usually consists of a 'mod_mymodule' directory with a 'mymodule.c' and a 'mod_mymodule.lua'. It is common for configuration to go into a 'cfg_modulename.lua' file in the configuration directory.

4.1.1 The lua part

The lua part of the module is defined in the Makefile of the module as the MODULE_STUB. When no MODULE_STUB is defined, an automatically generated stub is created:

```
ioncore.load_module('mod_mymodule')

-- Mark ourselves loaded.
package.loaded['mod_mymodule']=true
```

5 Miscellaneous design notes

5.1 Destroying WObj's

To keep Notion's code as simple as possible yet safe, there are restrictions when the WObj `destroy_obj` function that calls watches, the deinit routine and frees memory may be called directly. In all other cases the `mainloop_defer_destroy` function should be used to defer the call of `destroy_obj` until Notioncore returns to its main event loop.

Calling the `destroy_obj` function directly is allowed in the following cases:

- In the deinit handler for another object. Usually managed objects are destroyed this way.
- The object was created during the current call to the function that wants to get rid of the object. This is the case, for example, when the function created a frame to manage some other object but for some reason failed to reparent the object to this frame.
- In a deferred action handler set with `mainloop_defer_action`. Like deferred destroys, other deferred actions are called when Notioncore has returned to the main loop.
- You are absolute sure that C code outside your code has no references to the object.

If there are no serious side effects from deferring destroying the object or you're unsure whether it is safe to destroy the object immediately, use `mainloop_defer_destroy`.

5.2 Method signatures

5.2.1 The types `char*` and `const char*` as function parameters and return values

The following rules should apply to using strings as return values and parameters to functions.

Type	Return value	Parameter
`const char*`	The string is owned by the called function and the caller is only guaranteed short-term read access to the string.	The called function may only read the string during its execution. For further reference a copy must be made.
`char*`	The string is the caller's responsibility and it *must* free it when no longer needed.	The called function may modify the string but the "owner" of the string is case-dependent.

5.2.2 Use of pointers

When a method has a pointer as a parameter, and there may be confusion over whether parameter may be `NULL`, it is encouraged to add `http://www.splint.org`-style annotations such as `/*@notnull@*/` and `/*@null@*/`.

5.3 Encoding

Unless otherwise specified, character strings are encoded in the current locale's encoding. Convenience booleans `ioncore_g.enc_sb`, `ioncore_g.use_mb` and `ioncore_g.enc_utf8` may be used to determine whether the current locale's encoding is 8-bit, whether to use multi-byte routines and whether the multi-byte encoding is UTF-8.

6 C coding style

If you want to submit patches to Notion, you **must** follow my coding style, even if you think it is the root of all evil. We don't want the code to be an incomprehensible mess of styles and I have better things to do than fix other people's style to match mine. The style should be obvious by studying the source, but here's a list of some things to take note of.

6.1 Whitespace

- Indentations of 4 with spaces.
- No extra spaces between operators, delimiters etc. except
  - around logical and, or (`&&`, `||`)
  - around the conditional `a ? b : c`
  - after commas and semicolons

  In my opinion this helps pointing out arithmetic or other expressions within logical expressions or parameter lists.
- All kinds of labels are out-tended to the level of the higher level block. For example:

```
void foo()
{
again:
    switch(asdf){
```

```
    case 1:
        ...
        break;
    default:
        ...
        break;
    }
}
```

6.2 Braces

- Opening brace is at the end of the line, except in function bodies, where it is at the beginning of the line following the definition.
- Never put the body of a control statement on the same line with the statement (e.g. `if(foo){ bar() }`).
  For example, the block
  ```
  void foo(int a, int b)
  {
      if(a==b && c+d==e){
          ...
      }
  }
  ```
  has correct style while the block
  ```
  void foo(int a,int b) {
      if (a == b && c + d == e) {
          ...
      }
  }
  ```
  does not.
- The `else` keyword follows immediately after the closing brace of previous `if`, if any. (This might change so I don't care if you put it on the next line.)
- I have used the convention that control statement bodies containing a single statement do not need braces around the block if, in case of the `if` all the blocks in `if ... else if ... else` contain just one statement. If you want to, just use braces in every case.

6.3 Names

- Function and variable names only have lower case letters. Type names are in mixed case while constants and macros (`#define`s) are in upper case letters.

6.4 Miscellaneous

- In the definition of a pointer variable, the asterisk is attached to the variable name: `char *s;`. (One could claim this an exception to the second rule.)
- You might optionally want to use Jed's foldings to group blocks of related code in a file to keep it organized:
  ```
  /*{{{ Many related functions */

  void code()
  ```

```
{
    ...
}

...

/*}}}*/
```

I think that's mostly it. Study the source when in doubt.

A GNU Free Documentation License

Version 1.2, November 2002

51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "**Document**", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "**Invariant Sections**" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "**Cover Texts**" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "**Transparent**" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "**Opaque**".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "**Title Page**" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that

this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or non-commercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer

review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `http://www.gnu.org/copyleft/`.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with ... Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

B Full class hierarchy visible to Lua-side

```
Obj
 |-->WHook
 |-->WTimer
 |-->WMoveresMode
 |-->WRegion
 |     |-->WClientWin
 |     |-->WWindow
 |     |     |-->WRootWin
 |     |     |-->WMPlex
 |     |     |     |-->WFrame
 |     |     |     `-->WScreen
 |     |     |-->WInfoWin
 |     |     |     `-->WStatusBar (mod_statusbar)
 |     |     |-->WMenu (mod_menu)
 |     |     `-->WInput (mod_query)
 |     |           |-->WEdln (mod_query)
 |     |           `-->WMessage (mod_query)
 |     |-->WGroup
 |     |     |-->WGroupWS
 |     |     `-->WGroupCW
 |     `-->WTiling (mod_tiling)
 `-->WSplit (mod_tiling)
       |-->WSplitInner (mod_tiling)
       |     `-->WSplitSplit (mod_tiling)
       |           `-->WSplitFloat (mod_tiling)
       `-->WSplitRegion (mod_tiling)
             `-->WSplitST (mod_tiling)
```

Index