OpenHoldem 2.0 Project
Manual and Documentation

Last updated:

Table of Contents

Table of Figures

What is OpenHoldem?

OpenHoldem, and its supporting applications, OpenScrape, ManualMode and OHReplay, is an open source screen scraping framework and programmable logic engine for the online Texas Hold'em style poker game. This framework provides the capabilities to allow you to build your own Texas Hold'em robot (bot).

There are two major parts to any poker bot, the game state engine, and the action engine.

There are a number of approaches for getting the game state from a poker client, including directly reading the memory of the poker, injecting code into the client's address space, parsing the client's chat box, and interpreting the pixels presented on the screen by the client. OpenHoldem's game state engine uses the last approach - it provides a parameter driven method of interpreting the pixels ("screen scraping") presented by the poker client to determine the state of the game at any given time. Screen scraping is not a new approach to this kind of problem, poker botting or otherwise. Screen scraping has been a recognized approach for solving a specific set of computer problems almost since computers were invented. Screen scraping has been around for a longer period of time than most of you reading this have been alive! (http://en.wikipedia.org/wiki/Screen_scraping)

The action engine in OpenHoldem is called the Autoplayer. The Autoplayer will use logic that is provided by you to decide what poker action to take based on the current game state. The Autoplayer will then click the buttons and enter the text on the screen to make the poker client execute this action.

OpenHoldem is a framework. What that means is that you need to provide it with a couple of things to enable the game state engine and the action engine to operate correctly. The first is the parameters that instruct OpenHoldem how to interpret the pixels presented by the casino that you play at. Each casino presents these pixels in a slightly different manner, and thus these parameters need to be specific for each casino. Second is the logic to tell the action engine what to do - just like every person plays live/human poker in a different style, this logic will instruct OpenHoldem how to play the style that you want it to play.

OpenHoldem supports every casino in existence today, as far as we know. If you find a casino that does not work with OpenHoldem's game state or action model, please hit the forums and let us know. The developer team is very responsive to modifying OpenHoldem to work with the continually changing landscape of Texas Hold'em casino client software.

OpenHoldem also supports every sub-flavor of Texas Hold'em poker known to exist today. No-limit, fixed-limit, pot-limit, full ring, heads-up, 6-max, MTT, SNG, double or nothing, turbo, normal speed, whatever. Someone else has already made it work - you can make it work too.

1.1 What OpenHoldem is not

OpenHoldem is not a complete poker bot. There is much work you need to do to utilize the framework effectively, however this manual and the community on the forums can help you to do so. If you truly desire a poker bot that includes logic written by someone else, a quick Google search will turn up a number of likely candidates. Be aware, however, that a pre-packaged, complete, click-Setup.exe-and-

go, poker bot is not a good way to be successful. Consistent and trusted positive results from these pre-packaged bots are impossible to find. If pre-packaged bots really worked, if we could have spent $20 to make $1000's, we certainly would have bought one ourselves, rather than spending the thousands of hours to create this framework!

OpenHoldem does not allow for automated collusion. Poker botting currently is in a gray area of legality and morality, with many strong perspectives on both sides of the argument. The OpenHoldem development team strongly believes that automated collusion is clearly illegal and immoral, and will not provide the facility to help you achieve these ends.

OpenHoldem is not a general-purpose poker botting engine for all styles of poker. It is intentionally suited only to the "Texas Hold'em" flavor of poker favored my the majority of online players today. Don't ask if it supports Omaha - it doesn't.

1.2 WinHoldem heritage

Reproduced from the original post by Matrix here:
http://www.maxinmontreal.com/forums/viewtopic.php?p=28468#p28468

“For the first couple of years of its existence WH (WinHoldEm) was actively developed and developed an enthusiastic following. Although it was a commercial product Ray Bornert put in an astonishing amount of time answering queries on the WH BBS, and seemed to respond to opinions and requests there pretty quickly. One of the features he introduced were private forums (fora) which groups could request and then run their own discussions and developments within them. These must have been problem children for him. because there was a measure of falling-out, squabbling and bad blood within them. The biggest and most active of these was the infamous Lab1.

The first sign of Ray's movement away from the total nurture and support role was the introduction of Winscrape, a utility to enable the creation of profiles (Table maps). In principle WH users were then not completely dependent on him to generate table maps. This was probably motivated by the increasing tendency of pokersites to change their client display slightly at frequent intervals, and maintaining table maps must have been an increasing burden for Ray. After a period he then stopped creating table maps altogether, and threw the users onto their own abilities with Winscrape. A market economy around them was supposed to develop.

There were external WH groups by this time. Ray had closed down the labs, saying that he did not want to be in the position of one who could be accused of spying on their work, and forced them to find alternative homes. The only one which did this permanently and successfully was Lab1; ending up on a host run by a mysterious presence called MaxinMontreal. The external market economy for table maps did not work very well, because Ray had put no mechanism in WH to enable maps to be given an equivalent of DRM, and piracy and strife soon occurred.

Whilst all this was happening there was a third significant development – PBWC. [Ed: PBWC = PokerBot World Championship] Ray had always been more evangelical than commercial with regard to WH, and he shifted his energies to organising a pokerbot/human championship. Support and development of WH stagnated for a year, at least by comparison with the situation before.

These three threads combined to generate a lot of dissatisfaction amongst the dedicated WH users. The focus of this was in Lab1. The practical result was that a certain element produced a WH functional clone to seed an open source project called OpenHoldem. An open source equivalent of Winscrape followed.

This gamble seems to have paid off. There has always been a dichotomy in botters between their interests in collaborating to develop techniques, and their interest in protecting the final commercial results - the bots. Lab1 helped enormously by making their table maps non-commercial. This established the precedent that the infrastructure for a bot was a community responsibility and development. People seem to have accepted this as a good working model.

As to the hatred - the major OH protagonists acknowledge their debt to Ray for WH, and the only comment that Ray has made about OH is that you cannot compete with open source."

1.3 References

Binary package download site: http://code.google.com/p/openholdembot/downloads

Source code download: http://code.google.com/p/openholdembot/source
- Source instructions: http://www.maxinmontreal.com/forums/viewtopic.php?f=113&t=3542)
- Source license, GPL v3: http://www.gnu.org/copyleft/gpl.html

Discussion forums: http://www.maxinmontreal.com/forums

Issue tracker: http://code.google.com/p/openholdembot/issues

Wiki: http://www.maxinmontreal.com/wiki/index.php5

Other open source projects used in the OpenHoldem project:
- Spirit parser library; http://www.boost.org
- HScrollListBox; Nebula Technologies, Inc.; http://www.nebutech.com
- Dynamic window resizng class; Paul DiLascia; MSDN Magazine -- July 2001
- PokerEval library; https://gna.org/projects/pokersource/
- Hashing algorithm from Bob Jenkins; http://burtleburtle.net/bob/hash/doobs.html
- Visual Leak Detector; http://dmoulding.googlepages.com/vld
- Normalized random distribution Java class; Indiana; http://pokerai.org/pj2/index.php
- Scintilla source code editing component; http://www.scintilla.org
- Scintilla MFC C++ wrapper class; Horst Brückner; http://www.codeproject.com/editctrl/scintillawnd.asp?select=1475770
- Progress Indicator Dialog; P. GopalaKrishna; http://www.codeproject.com/win32/UPDialog.asp
- Netscape-style preferences dialog; Chris Losinger; http://www.codeproject.com/KB/tabs/saprefs.aspx

A (few) word(s) of caution

2.1 Effort

This is a lengthy manual, and for good reason. This is a complicated hobby with a steep learning curve. If you are reading this and coming into this hobby for the first time, and you have the expectation of running a profitable poker bot tomorrow, then we kindly suggest you close this manual and look elsewhere. This hobby is very rewarding, intellectually, academically and monetarily, but that reward does not come without effort. Maybe one of the $20 pre-packaged bots will meet your needs, if truly all you want to do is to MakeMoneyFast.

2.2 Scams

There are people who will try to sell you partial or complete OpenHoldem-based poker bot solutions. The same warning applies to these offers as to the $20 click-Setup.exe-and-go poker bots. Ask yourself how much you would sell a complete poker bot for if it was profitable and made you money every day, even while you were sleeping? A simple calculation should clarify this point:

- Say you have a pokerbot that makes $2 per table per hour. Most sites allow you to run 4 tables simultaneously without issue. This equates to $8 per hour.
- Further assume you run your bot 6 hours per day to better model human play. That gives you $48 per casino per day.
- If you the run your bot 5 days per week, you stand to make $12,480 per year.

Multiply that by the number of casinos you play at, and you have a reasonable target selling price. Would you be willing to pay that amount of money for an unproven, "complete", "winning" poker bot? How much do you think a $100 or $500 "complete" poker bot is really worth?

There is a long history of people being scammed out of hard earned money for "winning" poker bot solutions. Sometimes lots of money. Don't be stupid...the age-old maxim applies here: if it sounds too good to be true, it probably is.

2.3 Gray area

The hobby of poker botting is arguably a legal and moral gray area. While other online real money activities actively allow and even encourage the use of bots (stock trading, currency speculation, sports betting, etc), online poker has taken a somewhat unusual stance on this topic. Some casinos will attempt to detect your bots with various means, including spyware-like actions of scanning your hard drives and taking screen shots, statistical analyses of mouse clicks, post-analyzing hand actions for consistent behaviors and so on.

What this means, however, is that one must take caution when poker botting to ensure that you protect yourself, your money balances, and your private information on your computer. All of these anti-bot detection attempts can be defeated, and this will be discussed further in the stealth section.

On the legality question, the legality of poker botting has not been court-tested to date. Even if it was, some of these casinos believe they operate above the law and can do whatever they want to your account at any time.

Elaboration on the fine points of these arguments is beyond the scope of this document, but a vigorous discussion can be found on forums online, including PokerAI, here: http://pokerai.org/pf3/viewforum.php?f=80

The truth is that as long as casinos need to display the game state on your computer screen, there will be a way to use a bot to play the game. Simple fact.

How to begin

3.1 Choosing an operating system

OpenHoldem has been designed to work best on the 32-bit versions of Microsoft's Windows 2000 or Windows XP operating systems. Any service pack level on these two choices seems to be fine.

We recommend you avoid Vista or its dependents (Windows 7, etc). These operating systems add a lot of useless cruft and bloatware that is not useful in this hobby. Most of the poker botters you will interact with in the community use these systems as well, so as you learn and exchange ideas, it is useful to be on the same platform. This is especially true of Table Maps. A TableMap designed for Windows XP is not very likely to work on Vista.

Finally, as you scale your botting operation up, you want to keep things as lightweight as possible to best utilize the limited computing resources at your disposal. For this reason, Windows 2000 is often the choice of poker botters due to its very lean profile.

3.2 Configuring the operating system

OpenHoldem requires specific system settings in order to correctly intuit the game state from the pixels on-screen, and to correctly take poker actions.

Color depth

The display color depth must be set to 24-bit or 32-bit. Most Table Maps are designed for 32-bit color depth, so your display should be set to that for maximum compatibility with others' work.

Display resolution

You need enough display resolution to contain both the poker client window and OpenHoldem at it smallest size (i.e. table view off). In general, the minimum acceptable display resolution is 1024x768. The smaller resolutions 800x600 and 640x480 do not have enough real estate to show both the poker client and OpenHoldem without overlapping windows.

Font smoothing

Font smoothing must be turned off in order to interpret fort pixels correctly. Our objective is to interpret the pixels as presented by the poker client, and not post-mangled by the operating system. To disable font smoothing, right click the desktop, and choose Properties-->Appearance-->Effects. Disable smoothed fonts in this dialog.

Window style (themes)

Windows 2000 only has one window style. There are no themes, there is no candy blue coloring around windows, so you are OK if you use Windows 2000. With Windows XP, it is suggested to use the "Classic" theme, which will present windows in the same style as Windows 2000. You are of course free to use whatever theme you want, but be aware that public Table Maps usually do not work unless they are used with the same theme that they were designed for. Most OpenHoldem botters use Windows 2000 or the Windows XP Classic theme.

Windows XP and ClearType

If you use Windows XP, it is recommended that you disable ClearType fonts, for the same reason that we suggest you disable font smoothing. Here is the link to the Microsoft web page on how to do this:
http://www.microsoft.com/typography/cleartype/tuner/Step1.aspx

3.3 Installing the software

OpenHoldem and its supporting applications do not come with an installer. This is by design – installers tend to leave traces on your system, and in this hobby, the less traces, the better. Stealth will be discussed in greater depth in a later section, but this installer-less install is an important part of the stealth strategy.

Since you do not have an installer to do the work for you, there are several steps you need to take to correctly configure the software on your machine:

1. Create your "OpenHoldem" base directory. For purposes of this walkthrough, we will create this directory as "c:\mpb" ("My Poker Bot" – clever, eh?). You can make this directory anything you want, however.
2. Under c:\mpb, create another directory "c:\mpb\scraper"; this directory is used by the "Green Circle Button" to automagically connect OpenHoldem to a poker table. More on this later. There is a non-automagic way to connect to tables, but at this point, you probably want this directory.
3. Download the compiled binary packages. They can be found at the links in the {References} section.
4. Extract all the files from the compiled binary packages into the c:\mpb directory. When done, your directory should look something like this, depending or course, on which programs you chose to extract.

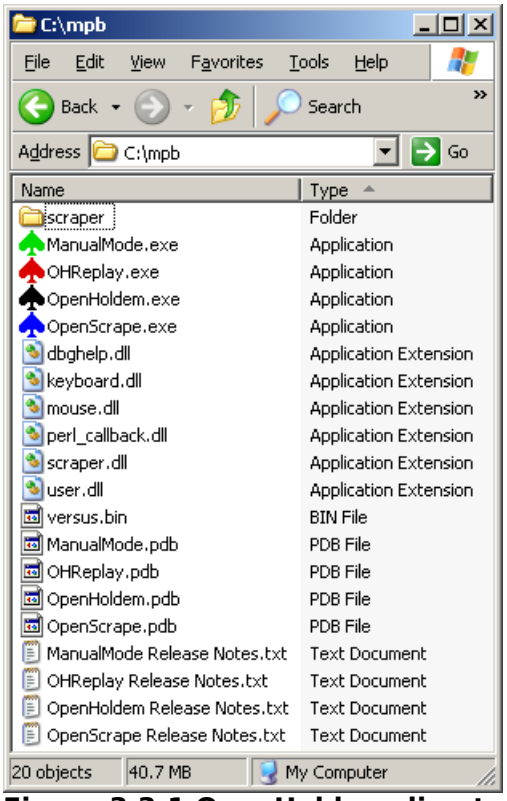


Figure 3.3-1 OpenHoldem directory

In this directory are the following files:

- ManualMode.exe, OHReplay.exe, OpenHoldem.exe, OpenScrape.exe – these are the main files for each of the programs in the OpenHoldem project. Details on how to use each of these are throughout this manual.
- dbghelp.dll – this is a Microsoft file that is used to produce memory dumps on critical errors. These memory dumps are useful for the developers to diagnose issues.
- keyboard.dll, mouse.dll, perl_callback.dll, scraper.dll, user.dll – these files are all extensions to OpenHoldem that you can use to customize the software. More detail on all of these are provided in this document.
- versus.bin – this file is the large lookup table that is used by the versus symbols that are described in this document. If you want to use versus symbols, you need this file.
- Files with .pdb extensions – these files contain the debug symbols that are required if you want to diagnose crashes or simply trace execution in a debugger. If you need these, you know how to use these. If you don't know what these are, you can safely ignore or delete them.
- Files with .txt extensions – these are the release notes for each of the programs in the project. Feel free to peruse these if you have problems falling asleep some night.

3.4 Finding a Table Map

The “scraper” directory should be where your Table Map files are stored. Table maps provide the OpenHoldem game state engine the instructions for deriving the state of the game from the pixels on the screen. These files have the “.tm” extension on them. The challenge with table maps is that every casino needs its own map. In some cases, you might need different maps for different game flavors. NL vs. FL, SNG vs. Ring, 10-seat vs. Heads-Up, for example.

The OpenHoldem project team does not provide official Table Maps for any casino or any Texas Hold’em game flavor. The reason for this is that there are thousands of casinos out there, and tens- or hundreds- of thousands of map combinations when you consider the different game flavors. Adding to that is the fact that casinos regularly update their client software and make minor or comprehensive changes to the pixels as they are displayed on the screen. These changes most often require changes to the table map as well. The OpenHoldem Project development team is a group of volunteers – we do not get paid for this, we have real lives and real families, and do this as a hobby. Maintaining working Table Maps would be a more than full time job, and thus instead, we put the power in your hands; we provide Table Map creation tools and testing tools to allow you to be self-sufficient. An explanation of the “OpenScrape” program, which is key to this self-sufficiency, will be provided in more detail later in this document.

That said, there are a number of places you can find pre-created maps for use with OpenHoldem. There is a section on the Wiki (http://www.maxinmontreal.com/wiki/index.php5?title=Table_Maps) that contains user-contributed maps. These maps are likely to be more or less out of date, and would require some validation to ensure they are 100% correct. Consider these maps to be a starting point for learning how maps are constructed and how they work. The forums have a lot of discussion on maps, and you can likely find maps attached to posts; same caveats apply as to the Wiki maps. You can try buying or trading maps with other community members – be cautious with this, however, and make sure you protect your interests; the OpenHoldem development team cannot be held accountable if you do something foolish on your own. Caveat emptor.

Creating a Table Map

Creating a Table Map is not a difficult process, once you get the hang of it, but it does have a steep learning curve. The benefit is that once you learn how to create maps, you can quite quickly take your bot to any casino on the Internet.

OpenScrape is the program that is used to create Table Maps. It provides a graphical interface to provide a set of parameters that are stored in human-readable text files with the .tm extension. Try it – open any .tm file in Notepad and you can quite clearly see the way the file is laid out. Each Table Map has a set of parameters, grouped by record type. The following sections will discuss how to use each of these record types to create your map.

4.1 First thing's first: connect to the poker window

Start a casino, and open up a play money table. You can even use "observer" mode for this activity if you like. Alternatively, once you learn how to use saved frames and OHReplay, you can also connect OpenScrape to these saved frames while offline from the casino.

For now, let's keep it simple. Once you have a poker table setup, startup the OpenScrape program from the "c:\mpb" directory that you set up above. Arrange the casino table window and the OpenScrape windows on your screen in such a way that there is no overlap – OpenScrape needs a clear view of the poker table window to do its job.

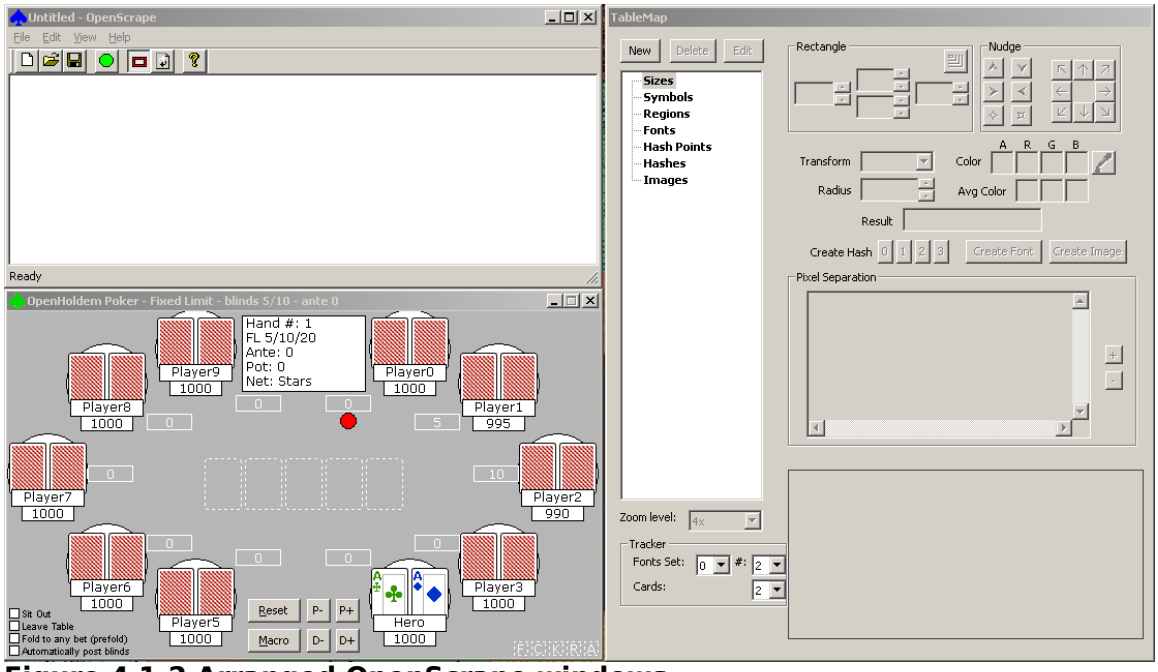


Figure 4.1-2 Arranged OpenScrape windows

Connect OpenScrape to the casino table by clicking on the Green Circle Button on the OpenScrape toolbar. A dialog with a list of top-level windows that are currently visible on your screen will appear. Select the one from the list that matches your casino table, and select OK. (Note, in this screenshot, we are connecting to a "ManualMode" window. You will learn more about ManualMode later in this

document, but for now, just consider ManualMode to be a simulated, offline poker table).

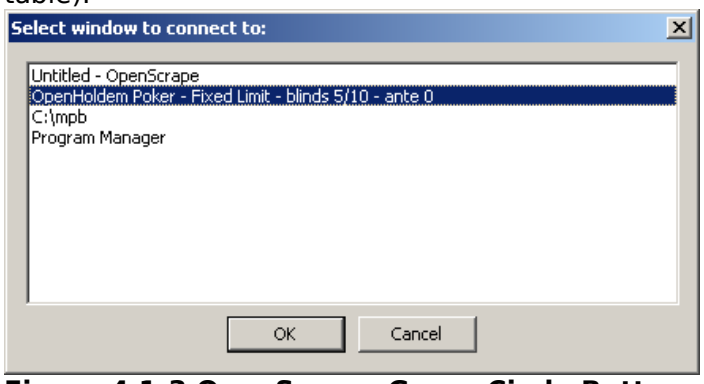


Figure 4.1-3 OpenScrape Green Circle Button selection window

A static image of the poker table window will be displayed in OpenScrape's main window. Once OpenScrape has the poker table's image displayed, you can minimize the poker table if you want, to give yourself more room to work. As the casino table changes state, and if you want to update the image that is displayed in the OpenScrape main window, you can do so at any time by clicking on the refresh button on the toolbar (Black and White arrows), or by choosing View/Refresh from the menu. When a refresh is requested, OpenScrape will bring the connected window to the front, grab the image, and then place it to the back again. Additionally, if you have OpenScrape connected to an OHReplay window, it automatically moves OHReplay to the next saved frame prior to grabbing the image.

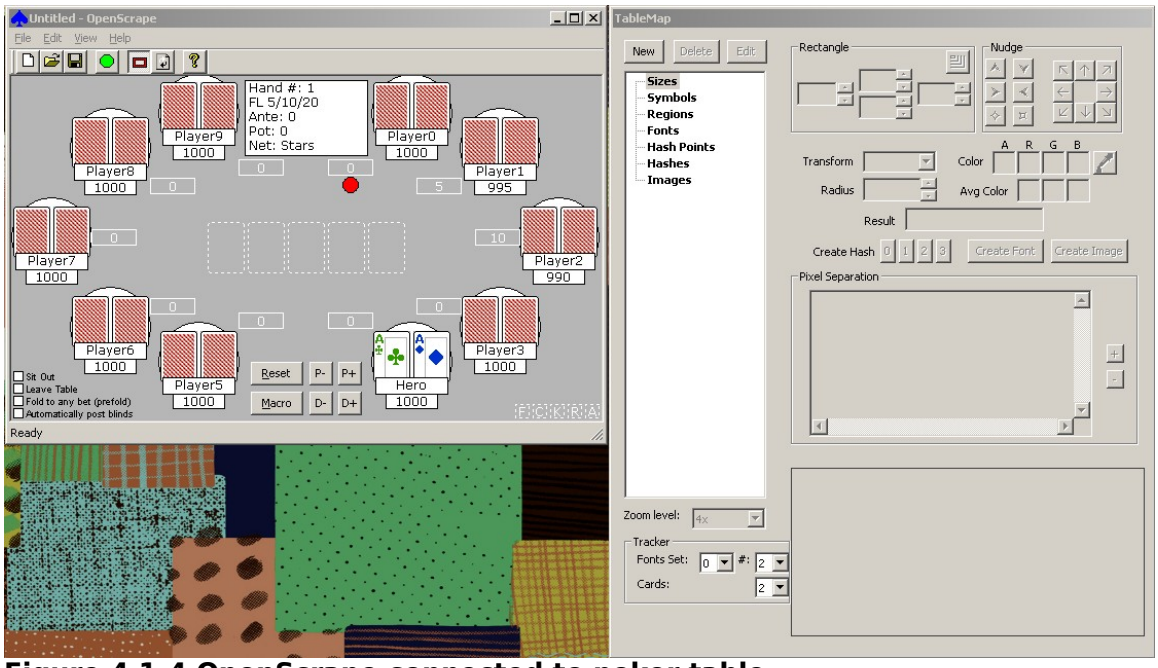


Figure 4.1-4 OpenScrape connected to poker table

Once we have the poker table image in OpenScrape, it is time to start building our Table Map, and providing the parameters that the OpenHoldem game state engine will use to "read" the poker table.

4.2 Familiarizing yourself with OpenScrape

When you started the OpenScrape program, two windows appeared. The “Main” window has the menu bar, the toolbar, and a display of the poker table image. The other window that appeared is the “TableMap Editor” window, and is where you will be doing most of your work with OpenScrape. The “TableMap Editor” window does not have a menu or toolbar, and has “TableMap” as its title.

Main: Menu Options

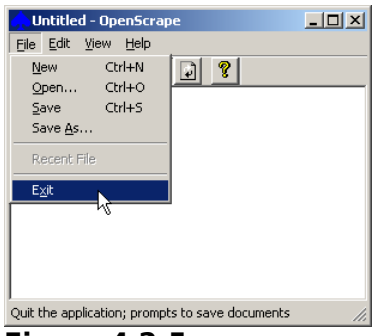


Figure 4.2-5 OpenScrape File menu

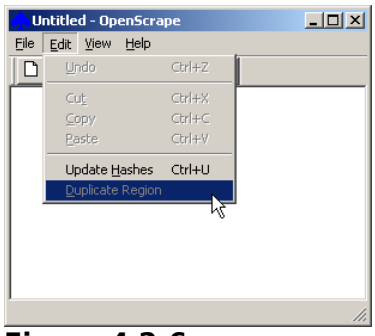


Figure 4.2-6 OpenScrape Edit menu

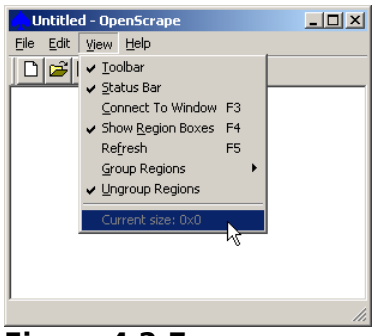


Figure 4.2-7 OpenScrape View menu

The File menu contains the standard Windows options to open and save Table Map files.

The Edit menu contains the standard Windows options, plus two options specific to OpenScrape:

- Update Hashes: This option will recalculate all the Hash records in the current map – useful if you have made changes to the underlying images, for example. More on Hashes later.
- Duplicate Region: This option will allow you to easily duplicate Regions. For example, once you have the player 1 card locations defined, you might want to copy those to players 2-10 to make things easier on yourself. More on Regions later.

The View menu has the standard Windows Toolbar/Status Bar options, plus the following specific OpenScrape options:

- Connect To Window: This option duplicates the “Green Circle Button” on the toolbar, and will bring up a dialog to allow you to select which window OpenScrape will copy an image from.
- Show Region Boxes: Allows you to toggle whether or not Region boxes are displayed on top of the poker table image in the main window.
- Refresh: This option duplicates the Black and White Arrow button on the toolbar, and will copy to current casino table image to the OpenScrape main window.
- Group Regions/Ungroup Regions: These options allow for customization of how Region records are displayed in the Table Map Editor window.
- Current size: Displays the client size of the window that OpenScrape is currently attached to. This is very useful when creating certain String records.

Main: Toolbar

Figure 4.2-8 OpenScrape Main window toolbar

The toolbar on the main window performs the following actions, from left to right:

- New Table Map
- Open Table Map
- Save Table Map
- Connect to window
- Toggle Region box display on or off
- Refresh the poker table image
- Display Help/About

> *Note: Keyboard shortcuts exist for many of these menu options and toolbar buttons, in addition to the standard Windows keyboard shortcuts:*
> *F1 – Help/About*
> *F3 – Connect to window*
> *F4 – Toggle Region box display on or off*
> *F5 – Refresh the poker table image*
> *Control + U – Update Hashes*

Table Map: Buttons/fields

The Table Map Editor window has a plethora of information on it, and various fields and controls will become enabled or disabled, depending on the type of map record selected.

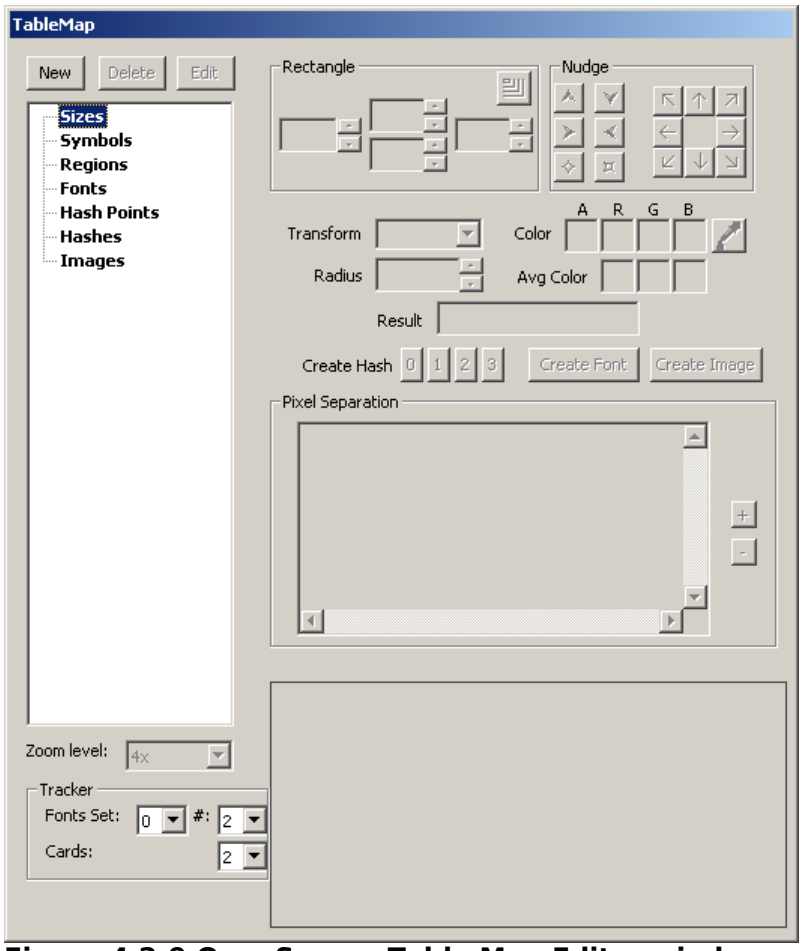


Figure 4.2-9 OpenScrape Table Map Editor window

On the top left of the window are three buttons, New/Delete/Edit, that allow you to add new records to the map, remove records from the map, or edit records in the map. The “New” button will prompt you to add a record of the type that is selected in the tree display directly below. For example, if you have a Region record selected, clicking New will prompt you to add a new region record to the map. The “Delete” and “Edit” buttons operate on the currently selected record.

Directly below the New/Edit/Delete buttons is the record tree display. This box displays all records that are currently present in the map, and groups them by type. The Region records within this tree can be further sub-grouped using the View menu options on the main window. There are a number of choices for sub-grouping Region records, and are intended to improve the productivity of Table Map creation. Experimentation of these sub-groupings is necessary to see if any meet your personal needs.

Below the tree display is a drop down to select the zoom level of the image that is displayed in the big box on the bottom right portion of the window. The image displayed is dependent on the record type selected.

Below the Zoom Level are various drop downs that are used to serve as checklists for capturing fonts and card images. Capturing fonts and card images will be discussed later.

On the top right of this window are a large group of controls, there is a “Rectangle” group, a “Nudge” group, and “Transform”, “Radius”, “Color”, “Avg Color” , and “Result” fields. These controls are used mainly for setting the parameters of Region records, and are only enabled when a Region record is selected in the record tree display on the left side of the window. The meaning of these fields will be described in the “Region” record section below. Some of these fields may display other relevant information, depending on the record type selected in the display tree.

Below this large group of controls are buttons to create Hash records, Font records and Image records. These buttons will be enabled/disabled appropriately depending on the record type that is selected in the display tree. For example, creating an Image record only makes sense if a Region record is selected in the display tree. Creating a Font record only make sense if a Region record with a text transform type is selected in the display tree.

The box below the Hash, Font and Image buttons is used to display “pixel separation”. The “+” and “-“ buttons to the right of this box control the size of the font used in this box. Pixel separation is used for regions that use font transforms. A good separation between foreground and background colors is imperative for good font recognition, and this box lets you preview that separation.

4.3 Size Records

Description

Size records are used for one purpose only, to allow OpenHoldem’s “Green Circle Button” to find windows of the correct size. More specifically, they provide the ability to specify the accepted sizes for the casino table that the scraper engine will recognize. Upon a click of the “Green Circle Button”, OpenHoldem will first search for poker table windows that match the exact size (*clientsize*) and if no match is found, will then look for poker table windows that are between the *clientsizemin* and *clientsizemax* sizes. In the majority of Table Maps, the only size record that is needed is the *clientsize* record. Use of *clientsizemin* and *clientsizemax* records should only be considered in unusual and exceptional cases.

Manipulating size records using OpenScrape

- To create a new size record, select the “Sizes” category in the record tree display on the Table Map Editor window, and click the “New” button. A window will be displayed that allows you to enter the width and height parameters for this new record.
- To edit an existing size record, select the record in the record tree display on the Table Map Editor window, and click the “Edit” button. A window will be displayed that allows you to adjust the width and height parameters for this record, or change the record name itself.
- To delete an existing size record, select the record in the record tree display on the Table Map Editor window, and click the “Delete” button. A window will be displayed asking for verification of this action.

A note about window client sizes: Microsoft Windows provides several different measures of a windows’ size, such as the complete window including frame and title bar, and an altogether different measure, the so-called “client area”, that lies inside the window frame and excludes frames and title bars. OpenHoldem references the Microsoft “client area” measurement for these “Size” records. To help you understand the size of the poker window that you have attached to (with the Green

Circle Button), OpenScrape always displays the client size of the attached poker table in the “View” menu on the main OpenScrape window. The menu option is at the bottom of this menu and is preceeded by “Current Size: “. The width and height displayed in this menu can be entered directly into a *clientsize* record.

Technical reference

In the Table Map (.tm) file, these records will be preceded with the characters "z$”. Size records have the following format:

```
z$<name> <width> <height>
```

<name> is the name of the record (see table below)
<width> and <height> are integer values.

Valid size records and their descriptions:

Record	**Description**
clientsize	Specifies the exact client size of the poker table window.
clientsizemin	Specifies the minimum allowed client size of the poker table window.
clientsizemax	Specifies the maximum allowed client size of the poker table window.

4.4 Symbol Records

Description

Symbol records are a general purpose record type that serves many different purposes within OpenHoldem and OpenScrape. The common characteristic of these records is that they all contain a name and a free-form text field. The interpretation of that text field is different, depending on the name of the symbol record. The table in the technical reference section below describes how the free-form text is interpreted for each different Symbol record type.

Manipulating symbol records using OpenScrape

- To create a new symbol record, select the “Symbols” category in the record tree display on the Table Map Editor window, and click the “New” button. A window will be displayed that allows you to enter the free-form text for this new record.
- To edit an existing symbol record, select the record in the record tree display on the Table Map Editor window, and click the “Edit” button. A window will be displayed that allows you to adjust the free-form text for this record, or change the record name itself.
- To delete an existing symbol record, select the record in the record tree display on the Table Map Editor window, and click the “Delete” button. A window will be displayed asking for verification of this action.

> *Note: The Insert and Delete keys can also be used as a shortcut for adding and deleting Symbol records.*

In the case of the “New” and “Edit” actions, the window that pops up also contains a field for “Titlebar text”, a button to parse the title bar text, and a field to display the parse results. These three controls are only relevant if the record name is one of the

ttlimits types. If a *ttlimits* record type is selected, these fields will be enabled, otherwise they will be grayed out.

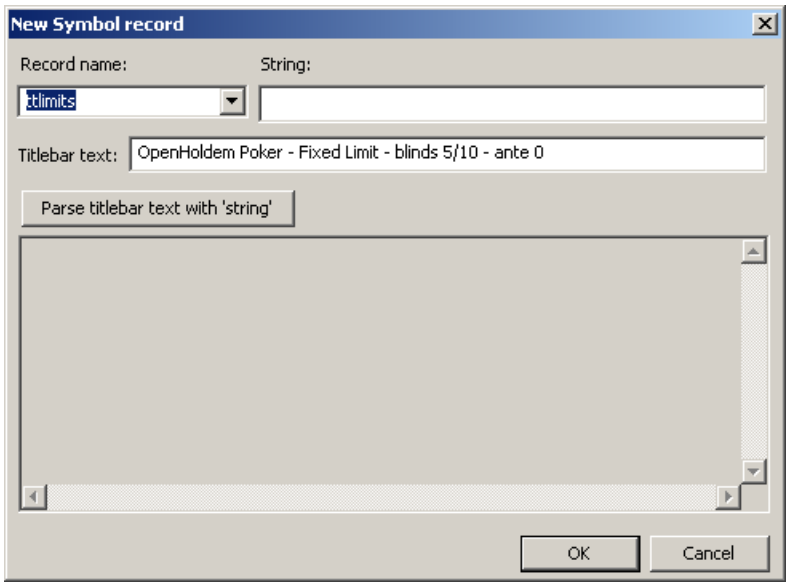


Figure 4.4-10 OpenScrape New/Edit Symbol window

These three fields allow you to test the parsing of the window's title bar text with the parse string that is specified in the "String" field, and see the results of this parse in the results field at the bottom of the window. The "Titlebar text" field is initially populated with the titlebar text of the window that OpenScrape is attached to (via the "Green Circle Button"), but this text can be overridden with anything you want to test parse.

Technical Reference

In the Table Map (.tm) file, these records will be preceded with the characters "s$". Symbol records have the following format:

```
s$<name> <text>
```

<name> is the name of the record (see table below)
<text> is the content of the record

Valid symbol records and their descriptions:

Record	**Description**
titletext, titletext0 - titletext9	When the "Green Circle Button" is pressed, OpenHoldem will look through each of these symbols and do a partial match of <text> to each window's title bar text. If any substring provided in any of these symbols matches, then the window is considered a match.
!titletext, !titletext0 -!titletext9	When the "Green Circle Button" is pressed, OpenHoldem will look through each of these symbols and do a partial match of <text> to each window's title bar text. If all substrings provided in all of these symbols do not match, then the window is considered a match.
ttlimits, ttlimits0 -	These symbols are intended to provide a format string that is

ttlimits9	used to parse the window's titlebar text, and extract sblind, bblind, bigbet, ante and game type information. The syntax usage of the format string is very much like the standard C-language format strings for scanf and printf. The placeholders use the caret character as the indicator and have the following meanings: ^h hand number ^d dummy integer (read chars 0-9 only) ^f dummy float (read chars 0-9 including decimal '.') ^s sblind ^b bblind ^B bigbet ^a ante ^v sb or bb depending on limit type ^V bb or BB depending on limit type ^* skip all text up to the literal character following the * ^R roman numeral as in "Level VIII" as typically seen in SNGs ^L limit indicator For the integer and float numerics, the scanner will automatically accept and skip a leading dollar sign “$” or leading/trailing cent sign “¢” (ascii 162 hex a2). In the case of the cent sign the target number will be divided by 100. The scanner will also recognize the standard trailing metric unit multipliers of “k” and “m” (1000 and 1000000 respectively). The following case-insensitive strings are recognized for the ^L limit indicator: no limit, no-limit, nolimit, nl pot limit, pot-limit, potlimit, pl fixed limit, fixed-limit, fixedlimit, fl, limit As an example, this title bar text "NL SNG Level VI 25/50 (2)" could be parsed with this format string "^L SNG Level ^R ^s/^b (^a)" to result in the sblind, bblind and game type being recognized. If multiple *ttlimits* symbols are provided, then the engine will scan each in order, and extract matching information from the first time it is found. That means that if you specify ^s in both *ttlimits* and *ttlimits3*, for example, it will be extracted from *ttlimits* and ignored in *ttlimits3*. OpenHoldem allows you to specify up to 11 *ttlimits* records in any single Table Map file. Specifying multiple *ttlimits* records might be needed if a given Table Map file is intended to describe multiple Texas Hold’em game flavors. However, it is a best practice to define different game flavors in different Table Maps, and thus most Table Maps contain a single *ttlimits* symbol record.
c0limits, c0limits0-c0limits9	These are text format strings with the same structure as the *ttlimits* records described above. They are intended to be used to extract blind/limit information from the *c0limits* regions using text transforms. As a poker site could certainly

	paint the various blind/limit information in different locations in the window, it is very likely that multiple *c0limits* records are required, if any are needed at all.
sitename	This record describes the name of the poker site that this Table Map is for. It is good practice to include the number of chairs and game flavor that this Table Map is intended to be used for. You might have a "fulltilt9sng", "fulltilt6ring" and "fulltilt2ring" for different Table Map files for Full Tilt's various table layouts and flavors. This record is what is queried via the OpenHoldem "sitename$" symbol.
network	The name of the poker network that this Table Map is for. This is used to determine the appropriate network to query Poker Tracker data for when s$sitename is not a Poker Tracker-supported site. OpenHoldem will first check s$sitename for a supported network name, then s$network. Supported names for both s$sitename and s$network must be lowercase and are the following: stars, party, ultimate, absolute, microgaming, ongame, cryptologic, pacific, fulltilt, b2b, tribeca, worldpex, ipoker, tain, bodog, everest, boss, and betfair.
nchairs	The number of chairs that this Table Map supports. The value of this record is what is returned via the OpenHoldem "nchairs" symbol.
reseller	Legacy support of WinScrape's symbol. Not particularly relevant in the open source OpenScrape world, but there if you need it. This is not used in the OpenHoldem engine anywhere, and is just metadata for the Table Map file.
mechanic	Legacy support of WinScrape's symbol. Not particularly relevant in the open source OpenScrape world, but there if you need it. This is not used in the OpenHoldem engine anywhere, and is just metadata for the Table Map file.
chairconfig	Legacy support of WinScrape's symbol. OpenHoldem does not adjust its display layout based on the value of this symbol like WinHoldem, and thus this is not used in the OpenHoldem engine anywhere.
swagtextmethod	Used by the Autoplayer to determine how to enter the f$swag amount into the swag entry box on the casino window.
potmethod	This is used to determine the appropriate site interpretation for the contents of c0pot0: 1: Common pot 2: Total pot, includes current bets and side bets
activemethod	This is used to determine how OH treats information from the pXactive and uXactive regions: 1: Inactive unless pXactive/uXactive returns true/active 2: Active unless pXactive/uXactive returns false/inactive/out/away
t0type - t3type	These symbols control which OCR algorithm will get used for the specific text transform 0-3. For example if the symbol h3type is defined as fuzzy then all regions with transform Text3 will use the fuzzy algorithm. The default value is "plain" if no symbol exists for a given group.

	The possible values for these symbols are "fuzzy", "plain", or a specific tolerance value. The default fuzzy text tolerance is 25%, but can be overridden by stating the value in this symbol, such as "0.355", which would cause the scraper engine to use the fuzzy text algorithm with a 35.5% tolerance value. Values of 0 are the same as "plain".
swagselectionmethod	* New Symbol in version 2.0; previously, this was a global preference set from the OpenHoldem preferences GUI. This symbol instructs the Autoplayer how to take the first action in a SWAG sequence. A SWAG sequence always consists of: 1. select the text in the SWAG box 2. delete the text in the SWAG box 3. enter the new bet in the SWAG box 4. confirm the SWAG bet amount. Valid values for this Symbol are available in a drop down control and are: ▪ "Sgl Click" – Single left click on the SWAG box to select the existing text ▪ "Dbl Click" – Double left click on the SWAG box to select the existing text ▪ "Click Drag" – Left click on the leftmost pixel of the SWAG box, drag to the rightmost pixel of the SWAG box, and then release the left mouse button to select the existing text ▪ "Nothing" – Skip the text selection step of the SWAG action
swagdeletionmethod	*** New Symbol in version 2.0; previously, this was a global preference set from the OpenHoldem preferences GUI.** This symbol instructs the Autoplayer how to take the second action in a SWAG sequence. A SWAG sequence always consists of: 1. select the text in the SWAG box 2. delete the text in the SWAG box 3. enter the new bet in the SWAG box 4. confirm the SWAG bet amount. Valid values for this Symbol are available in a drop down control and are: ▪ "Delete" – Press the “Delete” key to delete the selected text in the SWAG box ▪ "Backspace" – Press the “Backspace” key to delete the selected text in the SWAG box ▪ "Nothing" – Skip the text deletion step of the SWAG action
Swagconfirmation method	*** New Symbol in version 2.0; previously, this was a global preference set from the OpenHoldem preferences GUI.**

	This symbol instructs the Autoplayer how to take the fourth action in a SWAG sequence. A SWAG sequence always consists of: 1. select the text in the SWAG box 2. delete the text in the SWAG box 3. enter the new bet in the SWAG box 4. confirm the SWAG bet amount. Valid values for this Symbol are available in a drop down control and are: ▪ "Enter" – Press the “Enter” key to confirm the SWAG bet entry ▪ "Click Bet" – Click the “Bet” button to confirm the SWAG bet entry ▪ "Nothing" – Skip the bet confirmation step of the SWAG action
buttonclickmethod	*** New Symbol in version 2.0; previously, this was a global preference set from the OpenHoldem preferences GUI.** This symbol instructs the Autoplayer how to click any button defined by the iXbutton region records. Valid values for this Symbol are available in a drop down control and are: ▪ "Single" – Single left click on buttons ▪ "Double" – Double left click on buttons
handresetmethod	*** New Symbol in version 2.0** This symbol identifies which scrape changes should be used as a trigger to identify a "hand reset" event. This is a bitmapped integer; only one of the selected scrape changes needs to occur in order for the scrape to be considered a "hand reset". The values below are in hexadecimal format for explanatory purposes, but the value assigned to the *handresetmethod* symbol must be specified in decimal (base-10) format. As an example, if you want to instruct OpenHoldem to treat a change in dealer button OR a change in player cards as a "hand reset", then *handresetmethod* should be set to 5. If you want to instruct OpenHoldem to treat only a change in dealer button as a “hand reset", then *handresetmethod* should be set to 1. Bits and meanings: ▪ 0x00000001 (bit 1) = dealer button ▪ 0x00000010 (bit 2) = handnumber ▪ 0x00000100 (bit 3) = player cards (not triggered by change to "no cards" or to "card backs")
balancenumbersonly	*** New Symbol in version 2.0** This Symbol instructs the screen scraper that all characters besides numbers (0-9 and dot/decimal separator) are

	stripped from the text transform result for balance regions. This is useful for those casinos that include spurious/changing alphanumeric information in their balance boxes. Setting this string to "True" or "Yes" will enable balance cleansing, any other value or the absence of this Symbol will result in no balance cleansing.
chipscrapemethod	*** New Symbol in version 2.0** This Symbol instructs the “chip stack counter” within the screen scraper engine to act in one of three distinct modes of operation: ▪ If this Symbol is not present, or does not match one of the other formats, the “chip stack counter” begins counting chips at the 00 Region record, stops counting a given stack when a chip match is not found, and stops counting all stacks when the first chip in a given stack is not matched. This is the default, pre-2.0 behavior. ▪ If this symbol is set to "All”, then the scraper engine will check all 200 possible chip locations (10 stacks of 20 chips) every time a chip count is required. ▪ If this symbol is set to the format "XxY", where X is an integer between 1 and 10 representing the number of stacks, and Y is an integer between 1 and 20 representing the number of chips in each stack, then the scraper engine will check X times Y chip locations every time a chip count is required.
Scraperdll	*** New Symbol in version 2.0** This Symbol instructs OpenHoldem to load a user-supplied external scraper override DLL when this Table Map is used. This external DLL can be used to override the results of the internal scraper engine. This DLL is called immediately after the internal scraper engine finishes its pass every heartbeat, at which point the current state of the internal screen scraper is passed to the DLL for inspection and modification. A reference scraper DLL is provided in the source code as a starting point. An obvious use of this would be to hook or inject the target casino client to retrieve better state information than that which could be collected using the built-in pixel-based screen scraper.

4.5 Region Records

Description

The purpose of the Region records is to describe named rectangular areas on a poker table, each of which contains an element of the game state. These regions will “return” values to the OpenHoldem game state engine, depending on the type of transform selected.

An example will make this clearer. Each seat at the table will have two cards in front it, and we want our screen scraper to tell the OpenHoldem game state engine what those cards are. If the casino client displays a group of pixels for our first hole card,

that to a human being looks like an ace of hearts, we need somehow to tell the screen scraper to “return” “Ah” to the game state engine each time that group of pixels is seen in one of the card locations. To do this we collect an image for that ace of hearts, then setup a region record that returns “Ah” to OpenHoldem when that ace of hearts is seen during real live game play.

Manipulating region records using OpenScrape

- To create a new region record, select the “Regions” category in the record tree display on the Table Map Editor window, and click the “New” button. A window will be displayed that allows you to select the name of the region to create. After creating the record, you can use the controls on the right side of the Table Map Editor window, as described below, to change the settings for the new record.
- To edit an existing region record, select the record in the record tree display on the Table Map Editor window. Immediately after selecting the record, the controls on the right side of the Table Map Editor window are updated with the settings for the selected region record. These can then be changed, as described below.
- To delete an existing region record, select the record in the record tree display on the Table Map Editor window, and click the “Delete” button. A window will be displayed asking for verification of this action.

> *Note: The Insert and Delete keys can also be used as a shortcut for adding and deleting Region records.*

> *Note: Regions can also be selected by left clicking within the region’s flashing red rectangle that is layered on top of the poker table image on the Main window. If multiple regions are layered on top of each other, clicking again on the same area will select the next region below the selected region. This only works for two regions. If three or more regions are layered on top of each other, the selection tree will need to be used.*

The majority of the right side of the Table Map Editor window is used to manipulate the parameters of the region record.

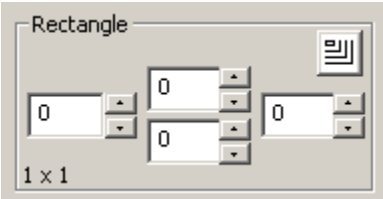


Figure 4.5-11 OpenScrape Rectangle Group Box

Starting at the top, there is a rectangle group box. This group box allows you to define the extents (left, right, top, bottom) of the region record’s bounding rectangle. The bounds are inclusive – in other words, a rectangle defined as left=0, right=5, top=0 and bottom=6 will be 6 pixels wide and 7 pixels high. The rectangle settings can be direct entered into the controls, you can use the spinners next to the controls, or you can use the rectangle drawing tool to quickly draw the rectangle on the poker table.

To use the rectangle drawing tool, click once on this control:

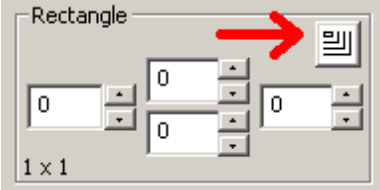


Figure 4.5-12 OpenScrape Rectangle drawing tool

...then move your mouse cursor to the Main OpenScrape window where the image of the poker table is displayed. The mouse cursor will change to look like crosshairs with an arrow pointing right and down. Left click and hold the mouse button anywhere on the poker table image to define the left-top corner of the rectangle. Drag to the right and down, and when you reach the right-bottom corner of the rectangle, release the left mouse button.

A zoomed view of the region is displayed in the box at the bottom right of the Table Map Editor window. The Table Map Editor window can be enlarged, if needed, to show the entire region. Use the Zoom Level control to adjust the zoom level in this display.

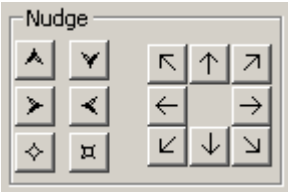


Figure 4.5-13 OpenScrape Nudge group box

To the right of the rectangle group box is the nudge group box. These controls allow you to adjust the size and position of the region record's rectangle in a variety of ways. The group of six controls on the left, starting from top left and moving clockwise, allow you to make the rectangle taller, shorter, thinner, smaller, bigger and wider. The group of 8 controls on the right, starting from top left and moving clockwise, allow you to move the rectangle up-left, up, up-right, right, down-right, down, down-left, and left.

> *Note: Regions can also be moved by first selecting the region by left clicking within the region's flashing red rectangle on the poker table image on the Main window, then by holding the Shift key and dragging and dropping the region to its new location.*

> *Note: Keyboard shortcuts also exist to move and resize regions:*
> *Arrow keys - Move the region 1 pixel*
> *Numpad keys 1, 3, 7, and 9 – Move the region diagonally 1 pixel*
> *Shift + Arrow keys - Move the region 5 pixels*
> *Control + Arrow keys - Resize the region by 1 pixel*
> *Control + Shift + Arrow keys – Resize the region by 5 pixels*

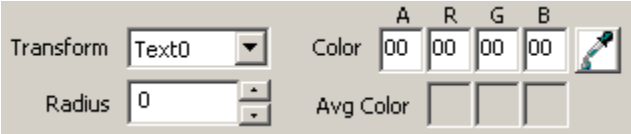


Figure 4.5-14 OpenScrape Region record parameters

The next set of controls allows you to set the remainder of the parameters on a region record. Here you can set the type of transform, the radius of the RGB color cube, and the center color of the RGB color cube. The use of RGB color cubes for transforms of type Color and Text is described in the technical reference below. Also in this group of controls is an eyedropper button that is used to populate the color parameters, as opposed to directly entering those values. To use the eyedropper, click once on the eyedropper button, then hover over the region's image display at the bottom of the Table Map Editor window. When the color you want is present in the four color fields, click the left mouse button to lock them in.

As a helpful reference, the "Avg Color" fields are populated with the calculated average color of all pixels in the region. This is useful for finding RGB color cube centers and for text transform separations.

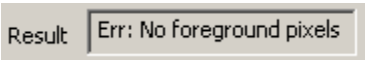


Figure 4.5-15 OpenScrape Transform result preview

This field provides an instant preview of the results of combining the current pixels in the region with the current transform for that region. If this is a Text transform, the actual transformed ASCII text is displayed. If this is a Color transform, True or False is displayed, depending if the region's average color is inside or outside of the defined RGB color cube. If this is a Hash transform, the value of the matching Hash record is displayed. If this is an Image transform, the value of the matching Image record is displayed.

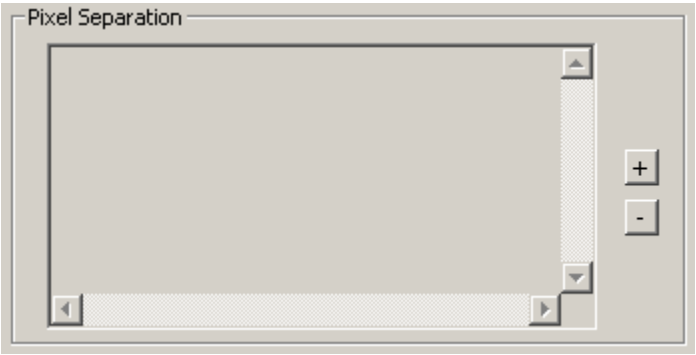


Figure 4.5-16 OpenScrape Pixel Separation display

If your region is defined to have a Text transform, then the Pixel Separation window will be populated with a preview of this separation. The key to good Text transformation is the separation of foreground pixel colors from background pixel colors. In this display, foreground pixels will be shown as an "X", and background pixels as a blank space. The size of the font used in this display can be adjusted with the "+" and "-" buttons to the right.

Note: Remember that the whole purpose of a region definition, with a given type of transformation, is to tell the screen scraper what the pixels mean that are displayed on the screen within that region. The return value of the transformation is provided to the OpenHoldem game state engine which then uses those values to establish the game state and to provide various information that

your bot logic needs to make action decisions. A human can easily look at the balance for seat #2 and see that it is $950.12. For a computer it is harder – we need to specify the rectangle that bounds this balance region, define it as a text transform, choose a color that lets the screen scraper identify foreground from background pixels, and then collect each of the individual font characters (more on this below) so that the screen scraper engine can look up the actual ASCII text for these pixels.

The Edit menu of on the Main window contains a "Duplicate Region" entry, which is intended to allow the quick creation of related region records. First, select a region record from the tree display on the Table Map Editor window, then select Edit/Duplicate Region from the Main window. This action will display this window:

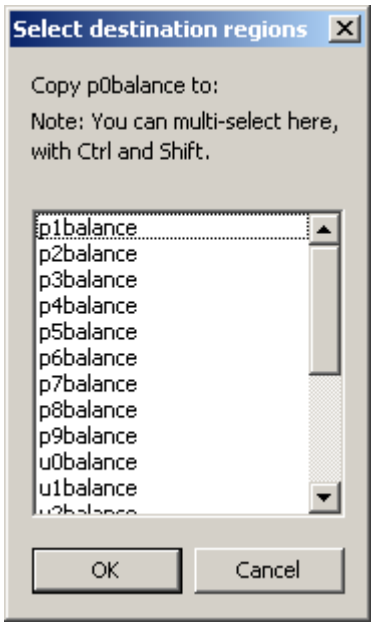


Figure 4.5-17 OpenScrape Duplicate Region window

In this case, the *p0balance* region was selected. You are given the option to duplicate the selected region record to other region records of the same type. In this case, all of the unpopulated *pXbalance* and *uXbalance* region records are candidates. You can select one or more than one destination record here (using the standard Windows Ctrl and Shift multi-select capability) to copy the selected region to. When you click the OK button, the new region records will be created for you, and you can then move them where needed and adjust them appropriately.

Technical Reference

In the Table Map (.tm) file, these records will be preceded with the characters "r$". Region records have the following format:

```
r$<name> <left> <top> <right> <bottom> <color> <radius> <transform>
```

<name> is the name of the record (see table below)
<left>, <top>, <right> and <bottom> are integer values within the casino table, where the upper left corner of the client area is considered to be 0, 0

<color> is a 1-8 character hexadecimal number in the form AABBGGRR (Microsoft Window's COLORREF format). The usage of the color value is dependent on the transform selected.
<radius> is an RGB color distance.
<transform> describes the engine algorithm that should be used to interpret the pixels in the region.

The available transforms are:

Type	Description
N	None - This is used for those regions that do not need a transform as their only purpose is to describe a rectangle for use by the scraper engine. An example is button location (i0button - i9button) which informs the Autoplayer engine of the allowed rectangle for mouse clicks.
C	Color - This is a calculation of the average RGB color for the entire region. Used in conjunction with the radius, the engine will determine if the contents of the region, as seen during game play, match the color/radius as specified in the Table Map. The color value in the region record defines the center of a cube, and the radius defines the dimensions of that cube. This cube is then mapped to an RGB color space, and if the average color of the region falls within this cube, then a match is reported. A negative radius can also be specified, in which case if the average color of the region falls outside of the cube, then a match is reported.
I	Image - This transform does two things. It first searches the Image records in the Table Map for a pixel-by-pixel match, and returns the value of that Image record if a match is found. If no pixel-by-pixel match is found, the transform then searches for the closest match, based on this perceptual difference algorithm: http://pdiff.sourceforge.net/ There is an internal hard-coded limit, which prevents Image records from matching the region if 65% or more of available pixels of that region do not match. If this limit did not exist, then clearly non-matching regions could return all sorts of junk. This exact-match-to-pdiff failover is automatic, and requires no additional work from the Table Map creator other than to set the transform to "I" and to collect the images. This is a powerful transform, but be aware that this power comes at the cost of CPU cycles.
T	Text - This is an OCR (http://en.wikipedia.org/wiki/Optical_character_recognition) algorithm for the purpose of converting screen pixels to ASCII text. The color cube specifications (as in the C transform) are used to separate the foreground from the background pixels. A lookup is then performed against the Font records in the Table Map on a character by character basis to produce a resultant text string.
Hn	Hash n - This transform reduces the contents of the rectangle to a 32-bit value. OpenHoldem and OpenScrape use only a single, well respected hash algorithm that is open for public review. Because of the selection of this algorithm, we have not yet seen a hash collision (that did not have an unrelated root cause, like record duplication). That is not to say that a hash collision is not impossible, however the mapping of 52 card images (for example) into a 32-bit address space makes this highly unlikely. If a collision

	ever occurs, there is the option to use hash types other than Type 0. Types 0-3 all use different hash seeds with the same algorithm, and thus if a collision is seen, one could simply change the hash type for that region. The Type 0 hash algorithm uses all pixels in a region to calculate a hash. To use specific pixels in the region, use Types 1-3. Types 1-3 default to using all pixels in a region as well, but if Hash Point records are present in the Table Map and of a matching type, the engine will use those specific points for calculation of the hash. Once a region is hashed, the resultant 32-bit hash value is then looked up in the h$ records in the Table Map. If a match is found, the name of the h$ record is returned and acted on appropriately (i.e. common card 1 is Queen-Hearts, Fold button is present, Player 2 is seated, etc.)

Valid region records and their descriptions:

Record	**Description**
c0cardface0 - c0cardface4	Common cards 0-4; represents the entire card, the best transformation is probably hash. Typically either *c0cardfaceX* is used, or *c0cardfaceXrank* / *c0cardfaceXsuit*, but not both. Valid results are a case-insensitive, two character string. The first character is the rank and should be one of: “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”, “t”, “j”, “q”, “k”, or “a”. The second character is the suit and should be one of: “c”, “d”, “h” or “s”.
c0cardface0rank - c0cardface4rank	Rank of common cards 0-4; the best transformation is probably text. Typically either *c0cardfaceX* is used, or *c0cardfaceXrank* / *c0cardfaceXsuit*, but not both. Valid results are case-insensitive, and are one of “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”, “10”, “t”, “j”, “q”, “k”, or “a”.
c0cardface0suit - c0cardface4suit	Suit of common cards 0-4; the best transformation is probably text. Typically either *c0cardfaceX* is used, or *c0cardfaceXrank* / *c0cardfaceXsuit*, but not both. Valid results are case-insensitive, and are one of “c”, “d”, “h” or “s”.
c0handnumber, c0handnumber1 - c0handnumber9	Ten regions to specify where the game state engine should find the current handnumber. The first region found with valid numeric information will be used and the remainder skipped. *c0handnumber* is searched first, then *c0handnumber1* to *c0handnumber9* in that order.
c0istournament	Used to identify if this table is a tournament game. Any transform can be used equally well with this region. If the result from this region’s transform is anything besides blank (empty string) then this region is considered to represent a tournament game.
c0smallblind	Identifies the region on the poker table where the small blind can be found. Title text parsing is much easier to implement if that option is available, otherwise a text transformation can be used with this region to identify the small blind.
c0bigblind	Identifies the region on the poker table where the big blind can be found. Title text parsing is much easier to implement if that option is available, otherwise a text transformation can be used with this

	region to identify the big blind.
c0bigbet	Identifies the region on the poker table where the big bet can be found. Title text parsing is much easier to implement if that option is available, otherwise a text transformation can be used with this region to identify the big bet.
c0ante	Identifies the region on the poker table where the ante can be found. Title text parsing is much easier to implement if that option is available, otherwise a text transformation can be used with this region to identify the ante.
c0pot0 - c0pot4	The value of the main pot should be identified by *c0pot0* using a text transform. *c0pot1* to *c0pot4* can be used to identify side pots if required.
c0potNchipXY	See chip scraping instructions for *p0chipXY - p9chipXY* below. The difference is that the "N" refers to the different pots - N=0 is the main pot, N=1-4 are side pots.
c0limits, c0limits0 - c0limits9	Identifies the region on the casino table where the limit information can be found; this region is used in conjunction with the Symbol records *c0limits* and *c0limits0-c0limit9* text parse strings. This region should really only be used if title text parsing is not an option.
i0button - i9button	Specifies the rectangular area in which the Autoplayer can click the mouse for each button 0-9.
i0label - i9label	Specifies the action that should be associated with buttons 0-9. By default, and if not overridden here, button 0 will be interpreted as the fold button, 1 as the call button, 2 as the raise button and 3 as the allin button. The return value from the transformation of this region (hash or text) will determine how the scraper engine interprets the button. All values are changed to lowercase and have spaces and '-' removed. Only the leftmost characters will be tested "allin", "a11in", "allln", "a111n", "aiiin" : button is seen as the allin button "raise", "ra1se", "ralse", "bet" : button is seen as the raise button "call", "caii", "ca11" : button is seen as the call button "check" : button is seen as the check button "fold", "fo1d", "foid" : button is seen as the fold button "autopost", "aut0p0st" : button is seen as the autopost button "sitin", "s1t1n" : button is seen as the sitin button "sitout", "s1tout", "sit0ut", "s1t0ut" : button is seen as the sitout button "leave" : button is seen as the leave button "prefold" : button is seen as the prefold button Examples: "All-In" will be recognized as allin; "Auto-post blinds" will be recognized as autopost
i0labelY - i9labelY	Where Y = 0 - 9. As an adjunct to the *iXlabel* regions, the *iXlabelY* region definitions allow the specification of an alternative location with alternate properties, such as color, for any button. The specification of these regions is exactly as that of the *iXlabel* regions with an addition of a numeric (0 - 9) suffix. The *iXlabel* regions are searched first, then the *iXlabelY* regions. The first

	region that resolves to a text value will stop any subsequent evaluations.
i0state - i9state	Specifies whether the *button0 - button9* is live and available to be clicked. The best transformation is hash or color. If the leftmost characters of the result from this region's transform are either "true", "on", "yes", "checked" or "lit" then the button will be seen as available to be clicked. If any other result is returned, such as "false", then the button will be seen as not available to be clicked. Results are case-insensitive.
i3edit	Specifies the rectangular region that can be used for entering a SWAG bet.
i86button, i860button - i869button	Similar to the *i0button - i9button* regions, however these regions are intended to be used to combat spam/popups that could occlude the poker window. If the corresponding state region returns true, the Autoplayer will try to click in this region to dismiss the spam/popup.
i86state, i860state - i869state	The corresponding state regions for the *i86button*, *i860button - i869button* regions above. The best transformation is hash or color. If the leftmost characters of the result from this region's transform are either "true", "on", "yes", "checked" or "lit" then the button will be seen as available to be clicked. If any other result is returned, such as "false", then the button will be seen as not available to be clicked. Results are case-insensitive.
p0active - p9active	Used to determine if a player is active (sitting in) or not (sitting out). Any transform can be used equally well with this region. If the leftmost characters of the result from this region's transform are either "true" or "active", then the region is considered to be active. If any other result is returned, such as "false", "inactive", "out" or "away", then the chair is not considered to be active. Results are case-insensitive.
p0balance - p9balance	Used to identify a player's balance. It only makes sense to use a text transform for these regions.
p0bet - p9bet	Used to identify a player's bet. It only makes sense to use a text transform for these regions.
p0cardback - p9cardback	Used to identify when a seat 0-9 displays a card back. The best transformation is hash or color. If the leftmost characters of the result from this region's transform are either "true" or "cardback", then the region is considered to be showing a cardback. If any other result is returned, such as "false", then the chair is not considered to have be showing a cardback. Results are case-insensitive.
p0cardface0 - p9cardface0	The first card for seats 0-9; represents the entire card. The best transformation is probably hash. Typically either *pXcardfaceY* is used, or *pXcardfaceYrank* / *pXcardfaceYsuit*, but not both. Valid results are a case-insensitive, two character string. The first character is the rank and should be one of: "2", "3", "4", "5", "6", "7", "8", "9", "t", "j", "q", "k", or "a". The second character is the suit and should be one of: "c", "d", "h" or "s".
p0cardface1 - p9cardface1	The second card for seats 0-9; represents the entire card. The best transformation is probably hash. Typically either *pXcardfaceY* is used, or *pXcardfaceYrank* / *pXcardfaceYsuit*, but not both. Valid results are a case-insensitive, two character string. The first

	character is the rank and should be one of: “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”, “t”, “j”, “q”, “k”, or “a”. The second character is the suit and should be one of: “c”, “d”, “h” or “s”.
p0cardface0rank - p9cardface0rank	The rank of the first card for seats 0-9. The best transformation is probably text. Typically either *pXcardfaceY* is used, or *pXcardfaceYrank* / *pXcardfaceYsuit*, but not both. Valid results are case-insensitive, and are one of “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”, “10”, “t”, “j”, “q”, “k”, or “a”.
p0cardface0suit - p9cardface0suit	The suit of the first card for seats 0-9. The best transformation is probably text. Typically either *pXcardfaceY* is used, or *pXcardfaceYrank* / *pXcardfaceYsuit*, but not both. Valid results are case-insensitive, and are one of “c”, “d”, “h” or “s”.
p0cardface1rank - p9cardface1rank	The rank of the second card for seats 0-9. The best transformation is probably text. Typically either *pXcardfaceY* is used, or *pXcardfaceYrank* / *pXcardfaceYsuit*, but not both. Valid results are case-insensitive, and are one of “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”, “10”, “t”, “j”, “q”, “k”, or “a”.
p0cardface1suit - p9cardface1suit	The suit of the second card for seats 0-9. The best transformation is probably text. Typically either *pXcardfaceY* is used, or *pXcardfaceYrank* / *pXcardfaceYsuit*, but not both. Valid results are case-insensitive, and are one of “c”, “d”, “h” or “s”.
p0dealer - p9dealer	Identifies if the dealer button is found for seat 0-9, or not. Color transforms are usually used for these regions, but hash, text or image transforms can be used too. If the leftmost characters of the result from this region’s transform are either “true” or “dealer”, then the chair is considered to have the dealer button. If any other result is returned, such as "false", then the chair is not considered to have the dealer button. Results are case-insensitive.
p0name - p9name	Identifies the name of the player in seat 0-9. It only makes sense to use a text transform for these regions.
p0seated - p9seated	Identifies whether seat 0-9 is occupied (seated) or not. Color transforms are usually used for these regions, but hash, text or image transforms can be used too. If the leftmost characters of the result from this region’s transform are either “true” or “seated”, then the chair is considered to be seated. If any other result is returned, such as "false" or "unseated", then the chair is not considered to be seated. Results are case-insensitive.
p0chipXY - p9chipXY	Note: Chip scraping should *only* be used if the actual numeric bet text cannot be found elsewhere on the poker table. Text transformation is magnitudes of effort easier to accomplish. Step 1 : Ensure that the poker site uses static landings for all of the chip stacks. Step 2 : Determine the size of an individual chip, and should include any portion of the chip that is painted anywhere within the window regardless of how faded or transparent. Step 3 : Create regions for the initial base of each player's chip stacks with the size determined in step 2; *p0chip00*, *p1chip00*, ..., *p9chip00*.

	Step 4 : Create vertical stride chip regions for each player's chip stack; *p0chip01*, *p1chip01*, ..., *p9chip01*. These regions should perfectly surround the 2nd chip that rests above the base pXchip00. Step 5 : Create horizontal stride chip regions for each player's chip stack; *p0chip10*, *p1chip10*, ..., *p9chip10*. These regions should perfectly surround the base chip in the second stack. Step 6 : Optional and rare: If the vertical stride between chips in a stack is not uniform, then you will need to define *pXchip02-pXchip09* for each chip in the stack. For those sites with non-uniform vertical chip strides, this means that there is a limit of 10 chips per stack that will be recognized by the engine. Step 7 : Optional and rare: If the horizontal stride between chip stacks is not uniform, then you will need to define *pXchip10-pXchip90* for each stack. For those sites with non-uniform horizontal chip strides, this means that there is a limit of 10 stacks that will be recognized by the engine. In a worst case situation, there may be 100 chip regions * 10 players = 1000 region definitions required to scrape chips! (So now go look at scraping the bet text instead)
u0active - u9active	These are alternatives to the *p0active-p9active* regions, and are used for the player's seat. They are typically used when the player's seat changes appearance when it is the player's action, to avoid misscrapes due to change in color/size, whatever. Any transform can be used equally well with this region. If the leftmost characters of the result from this region's transform are either "true" or "active", then the region is considered to be active. If any other result is returned, such as "false", "inactive", "out" or "away", then the chair is not considered to be active. Results are case-insensitive.
ubalance, u0balance - u9balance	These are alternatives to the *p0balance-p9balance* regions, and are used for the player's seat. They are typically used when the player's seat changes appearance when it is the player's action, to avoid misscrapes due to change in color/size, whatever. It only makes sense to use a text transform for these regions.
u0bet – u9bet	*** New Symbol in version 2.0** These are alternatives to the *p0bet-p9bet* regions, and are used for the player's seat. They are typically used when the player's seat changes appearance when it is the player's action, to avoid misscrapes due to change in color/size, whatever. It only makes sense to use a text transform for these regions.
u0cardface0 - u9cardface0	These are alternatives to the *p0cardface0-p9cardface0* regions, and are used for the player's seat. They are typically used when the player's seat changes appearance when it is the player's action, to avoid misscrapes due to change in color/size, whatever. Valid results are a case-insensitive, two character string. The first character is the rank and should be one of: "2", "3", "4", "5", "6", "7", "8", "9", "t", "j", "q", "k", or "a". The second character is the suit and should be one of: "c", "d", "h" or "s".
u0cardface1 - u9cardface1	These are alternatives to the *p0cardface1-p9cardface1* regions, and are used for the player's seat. They are typically used when the player's seat changes appearance when it is the player's action, to avoid misscrapes due to change in color/size, whatever.

	Valid results are a case-insensitive, two character string. The first character is the rank and should be one of: “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”, “t”, “j”, “q”, “k”, or “a”. The second character is the suit and should be one of: “c”, “d”, “h” or “s”.
u0dealer – u9dealer	* **New Symbol in version 2.0** These are alternatives to the *p0dealer-p9dealer* regions, and are used for the player's seat. They are typically used when the player's seat changes appearance when it is the player's action, to avoid misscrapes due to change in color/size, whatever. Color transforms are usually used for these regions, but hash, text or image transforms can be used too. If the leftmost characters of the result from this region’s transform are either “true” or “dealer”, then the chair is considered to have the dealer button. If any other result is returned, such as "false", then the chair is not considered to have the dealer button. Results are case-insensitive.
uname, u0name - u9name	These are alternatives to the *p0name-p9name* regions, and are used for the player's seat. They are typically used when the player's seat changes appearance when it is the player's action, to avoid misscrapes due to change in color/size, whatever. It only makes sense to use a text transform for these regions.
u0seated - u9seated	These regions are alternatives to the *p0seated-p9seated* regions, and are used for the player's seat. They are typically used when the player's seat changes appearance when it is the player's action, to avoid misscrapes due to change in color/size, whatever. Color transforms are usually used for these regions, but hash, text or image transforms can be used too. If the leftmost characters of the result from this region’s transform are either “true” or “seated”, then the chair is considered to be seated. If any other result is returned, such as "false" or "unseated", then the chair is not considered to be seated. Results are case-insensitive.
tablepointNNN	These regions are used when the "Green Circle Button" is clicked on the OpenHoldem toolbar to "automagically" find a table to connect to. These regions must be one pixel wide by one pixel tall and must use a color transform. All tablepoint records must match the pixel colors on the casino window for it to be recognized on a connect. See {How the “Green Circle Button” finds tables} for more details.

4.6 Font Records

Description

Font records are used as lookup records for Regions that utilize Text transformations. When a Text transformation is required for a region, the foreground pixels in that region will be scanned left to right, and the widest font record that matches those pixels will be considered a match. The region will continue to be scanned left to right until all the pixels in the region are consumed. The return value of that region’s text transform is this entire sequence of ASCII characters.

Manipulating font records using OpenScrape

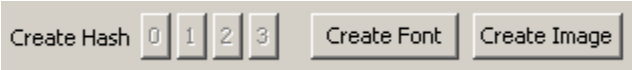


Figure 4.6-18 OpenScrape Create Font button

The "Create Font" button on the Table Map Editor window is used to create new font records. This button is only available if a Text transform is specified for the selected region record.

To create new font records:
1. Select a Region record that has a text transform
2. Ensure that the Region's color parameters provide good separation of foreground and background pixels (preview in the Pixel Separation field)
3. Click the "Create Font" button

> *Note: The number of the text transform in the region (Text0, Text1, Text2, or Text3) determines which Font record group (0 to 3) the new fonts will be added to.*

Assuming that some of the pixels in the selected region represent unknown font characters, then the following window will appear when the "Create Font" button is clicked:

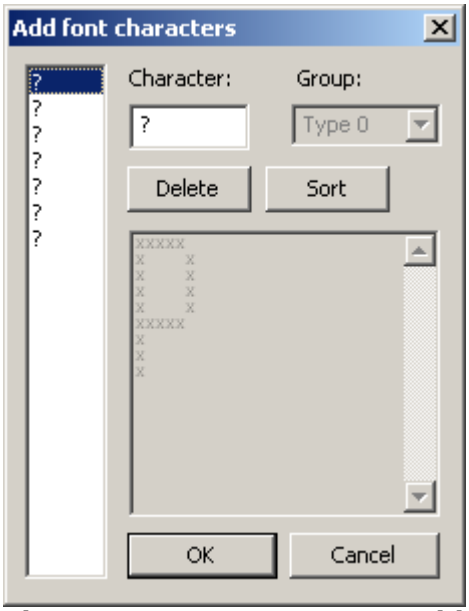


Figure 4.6-19 OpenScrape Add Font records window

On this window, a list of the individual font characters that OpenScrape thinks it has found is displayed in the list on the left side. As we have not told OpenScrape what character each of these pixel groups represent, they are displayed as question marks. (Note that the question mark can be considered a valid font character as well, so if you use this character in your Table Map, be aware of this fact).

For each unknown character in the list on the left, click to select it, then either:
- Enter which character is represents in the "Character" field (in this example, it is evident that the grouping of foreground pixels represents the capital letter P), or

- Click on the “Delete” button if this is not a character you want stored in the Table Map for some reason, such as it is a duplicate, or the automatic character separation didn’t work quite correctly

Clicking on the “Sort” button will arrange the characters alphabetically in the list, which might make them easier to process.

When you dismiss this window by clicking on the “OK” button, OpenScrape will add each of the characters in the list on the left, their ASCII representation, and their pixel layout to the list of Table Map records. This includes duplicates, and unknowns (question marks), so be sure to remove these prior to clicking “OK”, or your Table Map might get filled with superfluous Font records.

Fuzzy fonts

“Fuzzy fonts” are sometimes used by casinos when displaying text on their poker interfaces. You can identify if a casino uses fuzzy fonts by looking closely at the pixel patterns for individual characters as they are displayed at different times or in different locations on the screen. (Hint: use zoom) As an example, if when you look at the capital “A” character as it is displayed for seat 0’s name vs. seat 1’s name, you can see that those two letters use different pixels, of different colors, especially on the slanted arms of the “A”, you generally will need to use fuzzy font recognition.

The general procedure for defining fuzzy font records is as follows:

1. Decide which fonts will be fuzzy, and create those in a different Font group from your other fonts (we will use group 1 “Text1” in this example). Fonts can be created with the font color defined as the foreground pixel color, or the background color as the foreground pixel color. In the case of the latter, the so-called inverse fonting may sometimes give better results. Experiment! If using the font color for the foreground pixel color, it is probably best to make the fonts as thick as possible using a large RGB color cube radius. If using the background color as the foreground pixel color, a radius of zero probably works best.
2. Create a *t1type* Symbol record (for the Text1 group), and set the free-form text to "fuzzy" for default tolerance (25%), or to a specific tolerance value (0.50 for 50%, 0.33 for 33%, for example). Experiment to find the best tolerance – all casinos are different in how they display fuzzy fonts. If this symbol is set to anything besides "fuzzy" or a floating point value, then the engine will used non-fuzzy font recognition engine.
3. You can also create multiple samples for various characters, and it will help. The numbers “6”, “8”, and “3”, for example, are similar enough that distinguishing fuzzy versions of these is difficult. Creating more font records (samples) for the various 6's, 8s and 3's will help with the recognition.

Technical Reference

In the Table Map (.tm) file, these records will be preceded with the characters "t$". Font records have the following format:

```
T<n>$<a> <x0> <x1> <x2> ... <x30>
```

<n> is the font group (0-3) that this font record belongs to
<a> is an actual printable (non-whitespace) case-sensitive ASCII character (note that the angle brackets <> are not used)

<x0> to <x30> are 32bit hexadecimal values that describe the foreground pixels for the character. The maximum individual character width is 31 pixels. These values describe the on-off state of the pixels for the character, with 1 meaning on (foreground) and 0 meaning off (background).

4.7 Hash Point Records

Description

The purpose of Hash Point records is to define the pixels within a region that are used in a hash calculation. The “Hash0” transform always uses all pixels in a region to calculate its hash, so these records are only relevant for regions that use “Hash1”, “Hash2”, or “Hash3” transforms.

There is a one to one correlation between the “Hash1”, “Hash2”, or “Hash3” transforms and the group 1, 2 and 3 Hash Point records. For example, a region that calls for a “Hash2” transform will use the pixels specified by the group 2 Hash Point records in the hash calculation.

The points defined for a Hash Point group must fall within the boundaries of the regions that you are calculating a hash for. Hash Points that fall outside of the region will be ignored, as they are invalid. For example, if you have card regions that are 10x20 pixels, then hash points (11,15) and (8,22) will be ignored, whereas hash points (5,5) and (9, 18) will be used in the hash calculation.

Manipulating hash point records using OpenScrape

- There are two ways to create new Hash Point records:
  - Select the “Hash Points” category in the record tree display on the Table Map Editor window, and click the “New” button. A window will be displayed that allows you to enter the hash point group and the X and Y parameters:

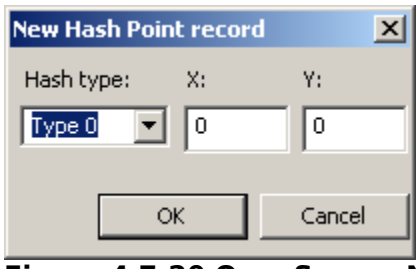


Figure 4.7-20 OpenScrape New Hash Point record – textual

  - Somewhat less intuitively, but ultimately easier to use, is to select the “Hash Points” category in the record tree display on the Table Map Editor window, and click the “Edit” button. A window will be displayed that allows you to select the hash point group, a sample image, and then interactively define the Hash Points for that group:

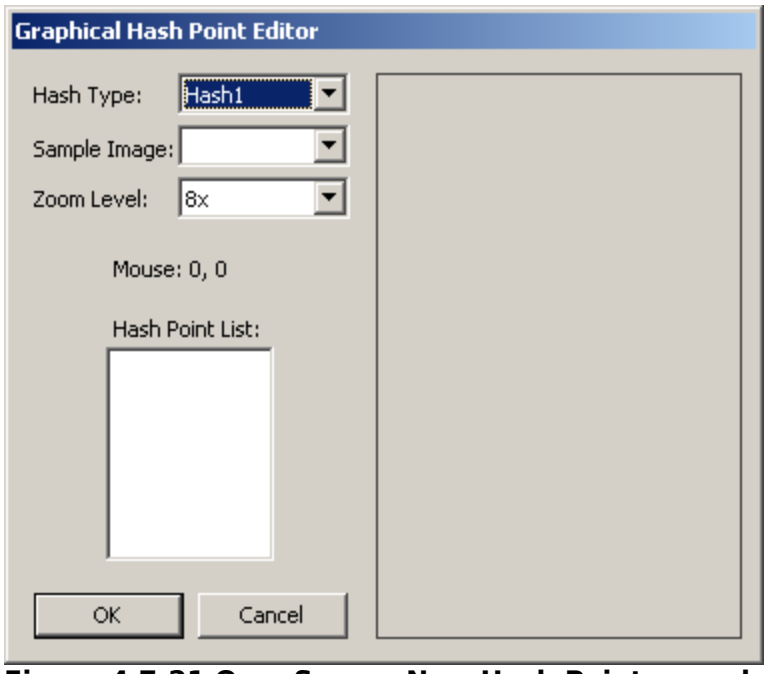


Figure 4.7-21 OpenScrape New Hash Point record – graphical

When you select a Hash Point group from the “Hash Type” control, the “Hash Point List” box will populate with all the Hash Points currently defined for that group. In this case, the box is empty, as we have not yet defined any Hash Point records for group 1.

After selecting an Image record in the “Sample Image” control (note you need to have Image records created first for this to work), the right side of the window will display that image and overlay the Hash Points for the selected group, like so:

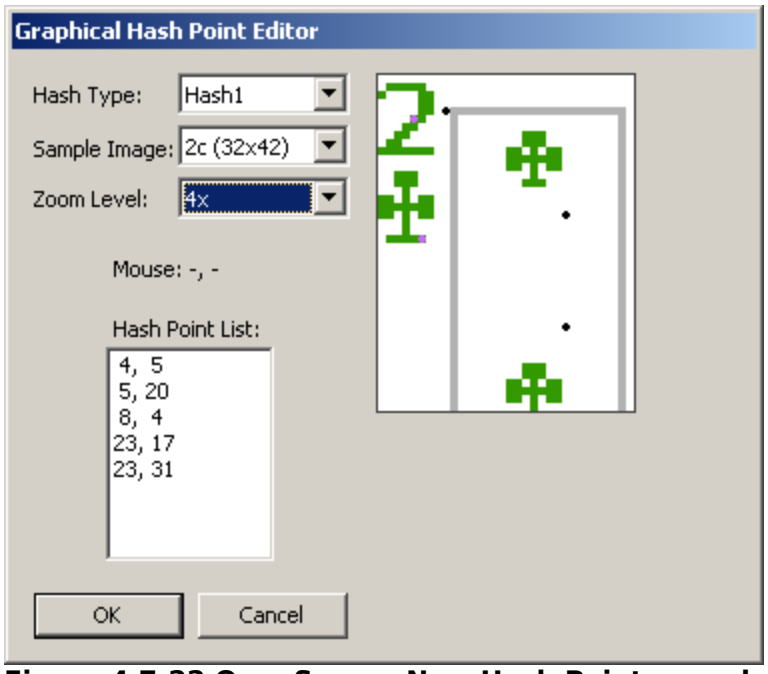


Figure 4.7-22 OpenScrape New Hash Point record – graphical with sample image

The zoom level of the sample image can be adjusted with the “Zoom Level” control. Adding Hash Points can be done interactively by left-clicking on the sample image where the point is needed. Right clicking on an existing point in the sample image will remove it from the group.

Clicking the “OK” button will then make the changes (adds and deletes) to the Table Map Hash Pointe record list.

- To edit an existing Hash Point record, select the record in the record tree display on the Table Map Editor window, and click the “Edit” button. A window will be displayed that allows you to edit the hash point group and the X and Y parameters.
- To delete an existing symbol record, select the record in the record tree display on the Table Map Editor window, and click the “Delete” button. A window will be displayed asking for verification of this action.
- An entire group of Hash Points can also be edited and/or deleted by selecting the “Hash Points” category in the record tree display on the Table Map Editor window, and clicking the “Edit” button. This will bring up the Graphical Hash Point Editor window, as described above.

Note: The Insert and Delete keys can also be used as a shortcut for adding and deleting individual Hash Point records.

Technical Reference

In the Table Map (.tm) file, these records will be preceded with the characters "p$". Hash point records have the following format:

p<n>$ <x> <y>

<n> is the group (1-3) that this hash point record belongs to

<x> and <y> are the coordinates of the point, relative to the region's rectangle, the top left corner of which is 0,0.

4.8 Hash Records

Description

The purpose of Hash Records is to allow for the rapid lookup of pixel patterns, without having to scan through each pixel in a region to find a match. For example, if we have a card region specified to use a “Hash0” transform, the screen scraper engine will first reduce the pixels in that region to a 32-bit hash value. (A link to Bob Jenkins’ hash algorithm that is used in this project can be found in the references section) Once the screen scraper engine has calculated the 32-bit hash value, it looks to see if there exists a matching 32-bit value in the associated group of Hash Records, in this example, group 0. If a matching 32-bit hash value is found, the screen scraper engine returns the <name> of the Hash Record to the game state engine.

Manipulating hash records using OpenScrape

Hash Records are created from specific Image records, so in order to create a Hash record, you must first have collected the associated Image record (see section {4.9}).

Figure 4.8-23 OpenScrape Create Hash Record buttons

The “Create Hash” buttons on the Table Map Editor window are used to create new Hash records. Button “0” creates Hash records in the Hash0 group, button “1” creates Hash records in the Hash1 group, and so on. These buttons are only available if an Image record has been selected in the tree display on the Table Map Editor window.

> *Note: Only one Hash record can exist in a Table Map in each hash group for any given Image record. Otherwise, there would be collisions, and the screen scraper engine would not know which Hash record was the correct match. In the “Create Hash Record buttons” screenshot above, Hash records for group 0 and 2 have already been created for the selected Image record, so those buttons are grayed out.*

To create a new Hash records:
1. Select an Image record
2. Click the “Create Hash” button for the hash group that you want to create this Hash record in.

The following window will then appear to confirm the addition of the new Hash record:

Figure 4.8-24 OpenScrape New Hash record confirmation

When you add a new Hash record with the “Create Hash” buttons, OpenScrape will calculate the hash value for you and add it to the record. (This is new behavior in 2.0; previous versions stored a zero hash value on new record creation) However, sometimes you may need to change that stored hash value, if for example, you replace the underlying Image record, or you change the number or location of the related Hash Points. If one or more hash values need to be updated, use the “Update Hashes” option on the Edit manu on the Main window, and the hash values on all Hash records will be re-calculated.

Technical Reference

In the Table Map (.tm) file, these records will be preceded with the characters "h$". Hash records have the following format:

```
h<n>$<name> <value>
```

<n> is the group (1-3) that this hash record belongs to
<name> is the string that is returned to the scraper engine upon a match (the “result”)
<value> is a 32bit hexadecimal value that contains the calculated hash

4.9 Image Records

Description

Image records serve multiple purposes. The first purpose is for regions that use the Image transform. When a region requires this type of lookup, the screen scraper engine will first look for an exact pixel-for-pixel image match in the Image records, and if no match is found, will then look for the closest match (with the 65% tolerance, as described in the Region Records {Technical Reference} section). The second purpose of the Image records is to serve as a reference from which Hash records are created (see the {Hash Records} section).

> *Note: Image transforms are very CPU intensive, as they require 2 passes through the Image records, the first to try to find an exact match, and the second to find a closest match. Each pass requires a comparison of every single pixel in the region. For large regions, this is a lot of comparisons. The time required to do Image transform lookups is directly proportional to the number of Image records present, and the size of the requesting Region record. It almost always makes more sense to use the Hash transform to do image matching, as the Hash lookup algorithm is orders of magnitude faster.*

Manipulating image records using OpenScrape

Figure 4.9-25 OpenScrape Create Image button

The “Create Image” button on the Table Map Editor window is used to create new Image records. This button is only available if a region record has previously been selected.

When the “Create Font” button is clicked, the following window will appear:

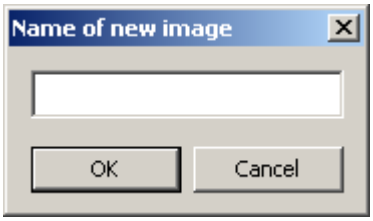


Figure 4.9-26 OpenScrape New Image Record window

The only required piece of information on this window is the name of the Image record, however, the name you choose is crucially important, as this name is what is returned to the game state engine directly for an Image transform, and is what is populated on the Hash record when you create a hash from this image.

The name is dependent on what the image is used for, but for example, if this is an image of the ace of clubs, you would want to name it “Ac”. If this is an image of a dealer button, you would want to name it “dealer”. (See the Region Records section, {Technical Reference}, for valid return values for each region)

Technical Reference

In the Table Map (.tm) file, these records will be preceded with the characters "i$". Image records have the following format:

```
i$<name> width height
scanline0
scanline1
...
scanlineN
```

<name> is the description of the image, and is the value returned to the screen scraper engine or is the name given to an associated Hash record
<width> and <height> are integer values that give the size of the image
<scanline0> - <scanlineN> contain the raw 32bit hexadecimal values for each pixel. Each pixel is in the form BBGGRRAA. There is no whitespace between pixel values. Each scanline is terminated with a standard Microsoft Windows line ending (carriage return/line feed). The number of scanline records is equal to the height of the image. The text length of a scanline record is width*8.

Creating the logic for your bot

5.1 Familiarizing yourself with OpenHoldem

So, here we are, 49 pages into the manual and we have not created a single snippet of bot logic yet. However, by this point you should have selected a target casino, and have acquired or created a Table Map for that casino so that the game state engine in OpenHoldem can understand what is happening on the poker table at any given time.

Before we get into the actual creation of bot logic, let's explore the OpenHoldem interface.

Menu Options

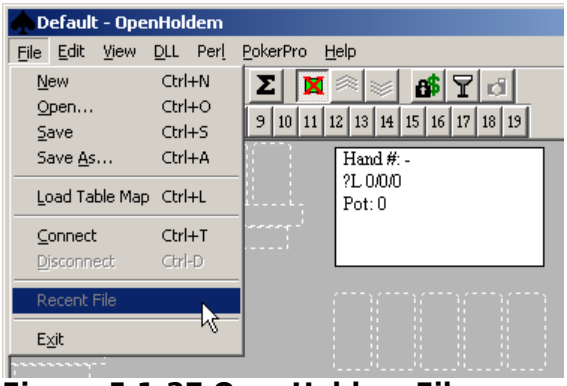


Figure 5.1-27 OpenHoldem File menu

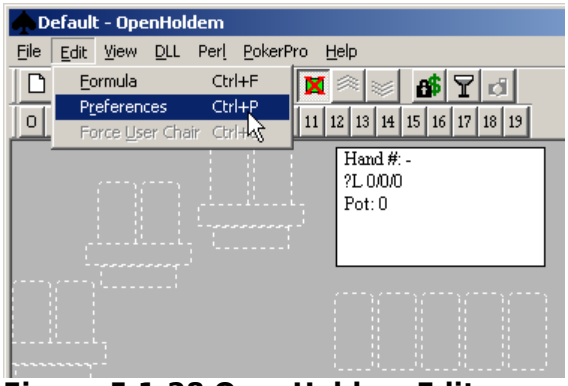


Figure 5.1-28 OpenHoldem Edit menu

The File menu contains the standard Windows options to open and save bot logic files. These logic files have the extension ".ohf", for "OpenHoldem Formula". There are also three OpenHoldem-specific options on this menu:

- Load Table Map: This option will force OpenHoldem to use a specific Table Map file and will override the automagic "Green Circle Button" method of connecting to poker tables.
- Connect and Disconnect: These options mirror the "Green Circle Button" and "Red Circle Button" on the main toolbar.

The Edit menu contains three options specific to OpenHoldem:

- Formula: This option will open the formula editor window – more information in the section {The Formula Editor}
- Preferences: This option will open a window to set various preferences for OpenHoldem's operation.
- Force User Chair: This option will open a window allowing you to specify which seat at the table that OpenHoldem considers the "User Chair". OpenHoldem usually autodetects the "User Chair" when card faces are seen in conjunction with action buttons, and this option will override that autodetection. This is probably only useful for testing purposes.

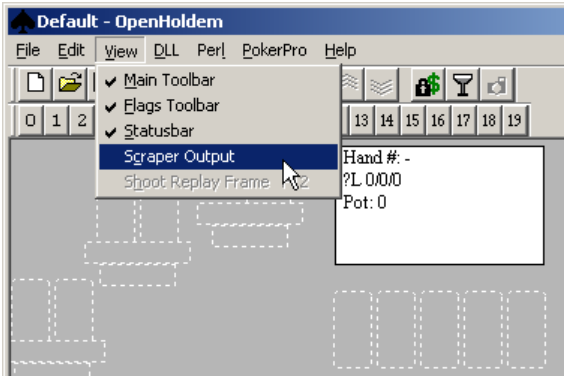


Figure 5.1-29 OpenHoldem View menu

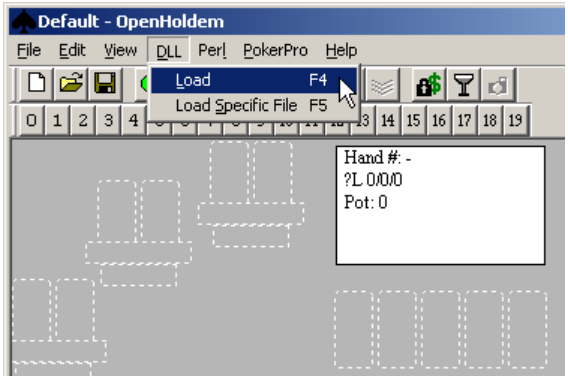


Figure 5.1-30 OpenHoldem DLL menu

The View menu has the standard Windows Toolbar/Status Bar options, plus two options specific to OpenHoldem:

- Scraper Output: This option opens a window that allows you to monitor the output from the screen scraper engine. This is useful for debugging purposes.
- Shoot Replay Frame: This option will save a "Replay Frame" in the "replay" folder in your OpenHoldem installation directory. See the {Replay Frames} section for more information.

The DLL menu is not a standard Windows menu item, but rather is used to load user-created DLL's containing bot logic for use by the OpenHoldem decision and action engine. See the {User DLL} section for more information.

- Load: This option will load the DLL specified in Preferences, or in the Formula file.
- Load Specific File: This option will present a standard File Open window to allow you to select a specific DLL file that you want to load.

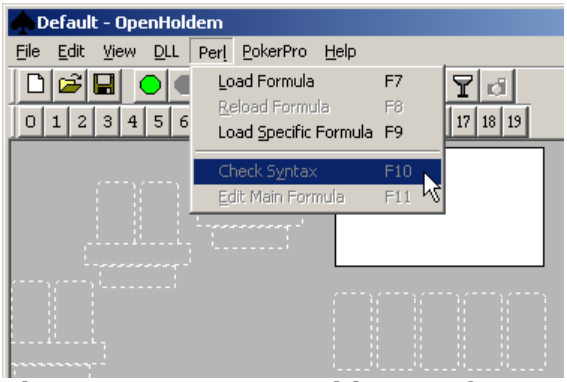


Figure 5.1-31 OpenHoldem Perl menu

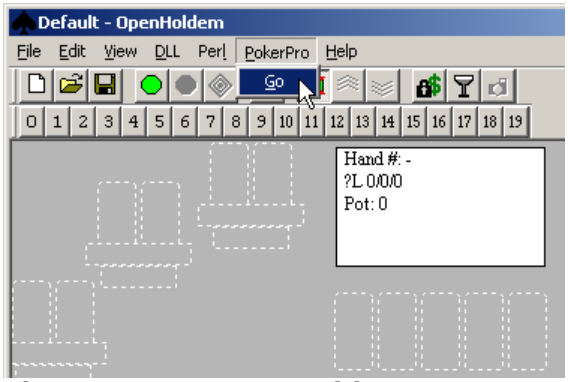


Figure 5.1-32 OpenHoldem PokerPro menu

The Perl menu is not a standard Windows menu item, but rather is used to load Perl files containing bot logic for use by the OpenHoldem decision and action engine. See the {Perl} section for more information.

- Load Formula: This option will load the Perl file specified in Preferences, or in the Formula file.
- Reload Formula: This option will re-load the Perl file specified in Preferences, or in the Formula file.

- Load Specific Formula: This option will present a standard File Open window to allow you to select a specific Perl file that you want to load.
- Check Syntax: This option will execute your Perl interpreters function to check the syntax of your Perl script.
- Edit Main Formula: This option will launch the editor specified in preferences and load it with your main Perl formula file.

The PokerPro menu is not a standard Windows menu item, but rather is used to connect OpenHoldem to a PokerPro server. PokerPro is an open server protocol designed by Ray Bornert. Full documentation on the PokerPro protocol can be found here: http://forum.winholdem.net/wbb/viewforum.php?f=22 More details can be found in the {PokerPro} section.

- Go: This option displays the PokerPro control window.

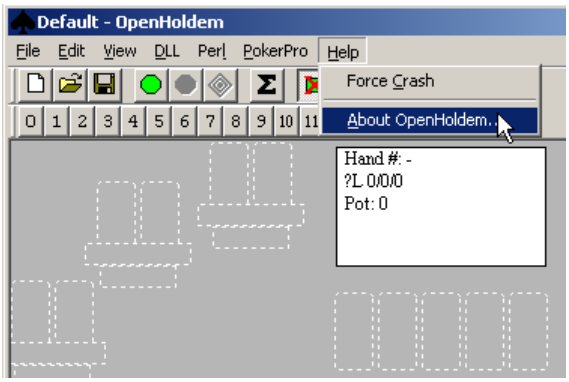


Figure 5.1-33 OpenHoldem Help menu

The Help menu has the standard Windows “About…” option, plus one option specific to OpenHoldem:

- Force Crash: This option will literally cause OpenHoldem to crash with an error. Specifically, the error is an assignment to a NULL pointer, like so:

  ```
  int *invalid_memory_access = NULL;
  *invalid_memory_access = 0;
  ```

  This option is present primarily for one reason, to allow you to test how your automation system will handle an OpenHoldem crash, if it occurs. OpenHoldem is very stable, but as with all software, the occasional bug creeps in. Also, as OpenHoldem allows the use of user-defined code (via DLLs, Perl scripts, etc), misbehaved user defined code can also cause crashes. See the chapter on {Automation} for more information.

Toolbars

Figure 5.1-34 OpenHoldem Main toolbar

From left to right, the buttons on the Main toolbar are:

- New formula file
- Open formula file
- Save formula file
- Connect to table (Green circle) – More information in the section {How the “Green Circle Button” finds tables}

- Disconnect from table (Red circle)
- Engage The Autoplayer (Nested diamonds)
- Formula editor (Sigma button) – More information in the section {The Formula Editor}
- Show/Hide table display (Green felt with red “X”)
- Attach to top of poker window (3 “Up” chevrons)
- Attach to bottom of poker window (3 “Down” chevrons)
- Lock blinds (Padlock and dollar symbol) – More information in the section {Locking Blinds}
- View scraper output (Paint scraper – this is NOT a wine glass!)
- Shoot replay frame (Camera) – More information in the section {Replay Frames}

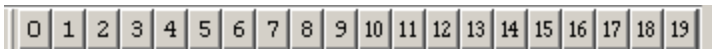


Figure 5.1-35 OpenHoldem Flags toolbar

The Flags toolbar is used to set internal OpenHoldem flag symbols to on or off. See the {OH-Script} section for discussion on the use of the flags symbols.

Status Bar

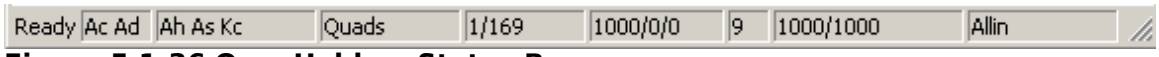


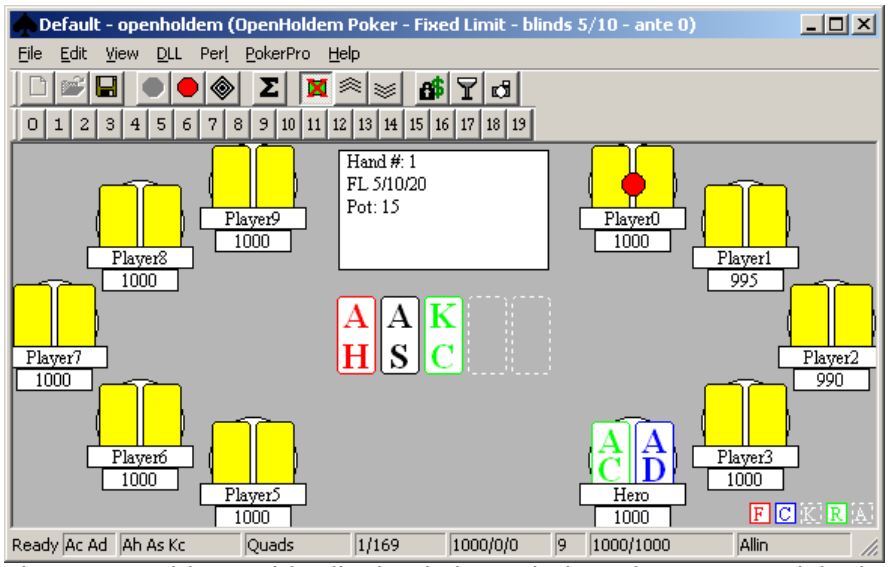
Figure 5.1-36 OpenHoldem Status Bar

From left to right, the fields on the Status Bar are:

- “Ready” – This is a standard Windows indicator to communicate application activity – for OpenHoldem it is always “Ready”.
- Your hole cards
- The common (board) cards
- Your poker hand
- The strength of your hole cards – the numerator of this field is set in preferences and can be one of 169, 1326 or 2652.
- Current values for prwin/prtie/prlos – More information in the section {The Iterator}
- Current value of f$P, one of the standard formula functions. This formula is intended to indicate number of opponents, however this number can be more or less than the number of opponents actually seen on the table at any given time. More information on f$P can be found in the {Primary Functions} section.
- Progress of the Monte Carlo iterator – the denominator of this field is drawn from your formula’s settings. See {The Formula Editor} section for more information on setting this number, and see the section {The Iterator} for more information on the Monte Carlo iterator.

The current action that the decision engine thinks the Autoplayer should take, based on the logic defined in your formula – see {Weighted prwin} section for more information on the Autoplayer.

Table Display



The OpenHoldem table display is intended to give you a quick visual reference as to what the screen scraper engine is actually seeing on the poker table at any given time. In this screen shot, OpenHoldem is connected to a ManualMode window. See the {ManualMode} section for more information on ManualMode.

The standard poker table information is presented in this display, including your hole cards, opponents' cardbacks, common cards, players' names and balances, the location of the dealer button, player bets and player seated/active status. If a player is seated, there will be a black circle displayed behind that players' card frames. If a player is also active, that black circle will be filled in with white.

The FCKRA boxes on the bottom right indicate the status of the Fold, Call, Check, Raise and Allin buttons; colored means those buttons are seen as ready to be clicked, and white dotted outlines means they are not seen as ready to be clicked.

The center information box contains various other information, including the hand number, limit type and limits, ante and current pot value.

5.2 The Formula Editor

When the "Edit/Formula" menu option is selected, or the "Sigma" button on the main toolbar is clicked, the following window is opened:

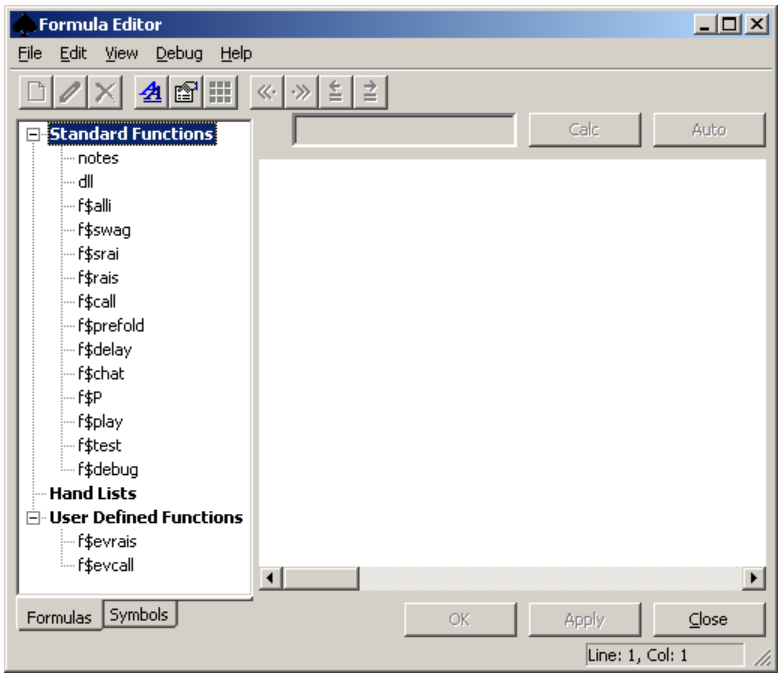


Figure 5.6-37 OpenHoldem Formula Editor

On the left side of this window is a tree (“Formulas”) that allows you to select the various functions that are present in your loaded formula file. Regardless of the language that you use for your bot logic, OH-script, C++ or Perl, the functions you see in the screenshot above will be present in every formula file, as these functions are central to how you instruct OpenHoldem to make poker decisions.

This tree also has a second tab (“Symbols”) that is built in documentation for all the OH-script language functions. See the {OH-Script} section for more information on this language.

The large control on the right side of the window displays the code for whatever function you have currently selected in the tree on the left.

> *Note: This window in the Formula Editor uses the open source “Scintilla Editor” component to display and allow editing of the selected funciton. More information on this component can be found here: http://www.scintilla.org*

Above this Scintilla editor control are a “results” field, and a “Calc” and an “Auto” button. The “Calc” button will calculate the currently selected function and display the results in the “results” field. The length of time in seconds that it takes to calculate this function is added to the titlebar text of the Formula Editor window in square brackets. The “Auto” button is only available when the “f$debug” function is selected, and will cause the information displayed in the Scintilla editor control to be updated once every screen scrape cycle.

Below the Scintilla editor control are “OK”, “Apply” and “Close” buttons that mirror the functionality of the File menu options, and the toolbar buttons as described below.

Primary Functions

The following table describes the primary functions. These functions are “primary”, because they directly instruct the Autoplayer to take poker actions, such as going all-in, raising, checking, calling or folding. For more information on the Autoplayer’s processing logic, please see this section {Weighted prwin}.

Function	Description
f$alli	This function instructs the Autoplayer to move all-in, either with the slider, or by entering the bet into the SWAG box, depending on the parameters in the Table Map. If this function evaluates to a non-zero result, and an appropriate control is present on the table (slider, swag field), then the Autoplayer will move all in.
f$swag	This function instructs the Autoplayer to place a stated wager (SWAG) by entering the bet into the SWAG box. If this function evaluates to a non-zero result, and the SWAG box is present on the table, then the Autoplayer will enter the results of f$swag into the SWAG box and confirm the bet.
f$rais	This function instructs the Autoplayer to raise. If this function evaluates to a non-zero result, and the raise button is present on the table, then the Autoplayer will click the raise button.
f$call	This function instructs the Autoplayer to call. If this function evaluates to a non-zero result, and the call button is present on the table, then the Autoplayer will click the call button.

Secondary Functions

The following table describes the secondary functions. These are secondary, because they do not directly instruct the Autoplayer to take action, but rather are used indirectly to determine the action, some of which are not the traditional poker actions of all-in, raising, checking, calling or folding.

Function	Description
f$P	This function is intended to evaluate to the number of opponents that should be simulated when OpenHoldem calculates various symbols. The symbols influenced by this function are those calculated by the Iterator (see {The Iterator} section}, prwin, prlos, and prtie, and also by the symbols prwinnow, prlosnow, and handrankp. The f$P value is important to these symbols because it determines how many opposing hands are simulated. The f$P value will radically affect the values of these symbols. For example, the higher the f$P value the more opponent hands will be simulated and your win probability will be lower. Similarly, the lower the f$P value the less opponent hands will be simulated and your win probability will be higher. For instance, AA against 9 opponents (f$P==9) has about a 30% chance of winning, assuming every hand stays for the showdown (no-foldem). AA against 1 opponent (P=1) has

	about an 80% chance of winning. The possible values for f$P are 0 to 22, inclusive. If f$P exceeds 22, it will be set to 22, as 22 is the maximum number of players our imaginary HoldEm table would be able to play (22 opponents * 2 cards = 44 cards, my two cards, the 5 common cards).

f$srai	This function is a support function for f$swag. The intent is that f$srai specifies the amount to raise above the current bet, and that f$swag then further adjusts that based on the style that the individual casino uses to post SWAG bets. Here is an example of a vanilla f$swag function that calls f$srai:
	`##f$swag##`
	`f$srai>0 ? ((((f$swag_adjust + f$srai)/sblind)`
	`+.5)^0)*sblind :`
	`0`
	
	`##f$swag_adjust##`
	`swagtextmethod == 3 ? (call + currentbet) :`
	`swagtextmethod == 2 ? (call) :`
	`0`
	
	Note that this f$swag also rounds your bet to the nearest sblind.

f$prefold	If this function evaluates to non-zero, and a pre-fold button is visible on the poker table window (as defined in the Table Map), then OpenHoldem will engage that pre-fold button as soon as it can (which may be before it is your turn to act).
	
	A good example of a f$prefold function might be the following:
	`br==1 && nplayersdealt>4 && handrank2652>2652*0.66`
	`&& !currentbet && !f$swag && && !f$rais && !f$call`

f$delay	The function controls how long to delay, once the specified number of stable frames have been seen (see {Preferences} section}, to take a poker action. The function evaluates to the delay time in milliseconds. An example of this might be: `f$monsterhand ? randomround * 3000 : 0)` This function says that if f$monsterhand returns true, to randomly delay the Autoplayer's action from zero to three seconds, otherwise it will not delay at all.
f$chat	This function directs the OpenHoldem Autoplayer when and what to enter into the casino client's chat box. See the {Before you try to configure OpenHoldem to work with a Poker Tracker database, you must have a properly working Poker Tracker configuration set up first. This is not the most trivial of activities, and does have a learning curve. The best information on how to set up Poker Tracker can be found here: http://www.pokertracker.com } section for more information.
f$play	This function controls how the Autoplayer interacts with the sitin, sitout, and leave buttons as defined in the Table Map.

	The possible values that this function can return are explicit and limited. Any other return values are ignored. The allow return values and their meaning are: -2, Leave table -1, No change (if you are sitting out, stay out, if sitting in stay that way) 0, Sit out 1, Sit in/Stay in
f$evrais	This function is a support function for f$rais. The intent is that f$evrais calculates the “Expected Value” of a raise action, and this value is then used in the f$rais decision function. This is generally considered a legacy function from WinHoldem, and is provided here for legacy compatibility reasons.
f$evcall	This function is a support function for f$call. The intent is that f$evcall calculates the “Expected Value” of a call action, and this value is then used in the f$call decision function. This is generally considered a legacy function from WinHoldem, and is provided here for legacy compatibility reasons.

Helper Functions

The following table describes the helper functions. These are not used by the Autoplayer.

Function	**Description**
notes	This is a free form function for you to use as you please. This is typically used for usage instructions and version notes.
dll	Contains a filename or path to a DLL to load by default. For example: `my_user_dll.dll`
f$test	This function is intended to be used to test expressions. Whatever is typed in here gets calculated as a whole when the "Calc" button is clicked. f$test can be referenced from other formulas, although that is not recommended, as f$test is really intended as a development/debugging aid.

f$debug	The debug function operates differently than the other function. Its purpose is to allow for arbitrary expressions to be evaluated and the results of those expressions to be displayed. To add a new expression add a new line with an equal sign, followed by the expression.
	For example:
	`= 2+2`
	`= f$UserDefinedFormula`
	`= nopponents`
	
	In order for the results to be displayed you must either press the 'Calc' or 'Auto' button. The 'Auto' button will recalculate the debug formula once each heartbeat.
	For example:
	`4 = 2+2`
	`1 = f$UserDefinedFormula`
	`6 = nopponents`

	If an expression is invalid it will be reported where the result would normally be found.

Menu Options

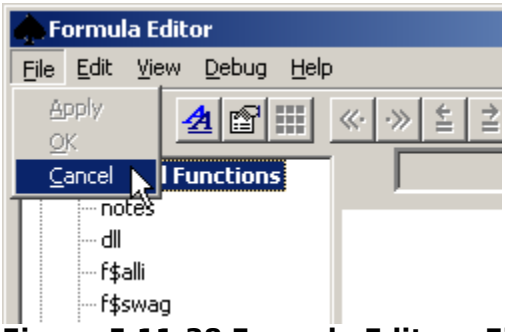


Figure 5.11-38 Formula Editor - File menu

The File menu has three options to control how changes made to your formula are applied back to the production formula:

- Apply: Copy changes made to the working formula to the production formula and keep the Formula Editor window open
- OK: Copy changes made to the working formula to the production formula and close the Formula Editor window
- Cancel: Discard changes made to the working formula and close the Formula Editor window

> *Note: When you open the Formula Editor, OpenHoldem makes a copy of the formula that you currently have loaded for use in the Formula Editor. Consider the formula that you loaded with File/Open, from the main OpenHoldem window, to be the "production" formula, and the copy that got made when you opened the Formula Editor to be the "working" formula. Keep in mind that the OpenHoldem Autoplayer ONLY operates on the production formula, and the Formula Editor ONLY operates on the working formula. The Apply/OK/Cancel options allow you to put changes back to the production formula that you made in the working formula. However, both the production and the working formulas draw from the same set of scraper engine output and calculated symbols.*

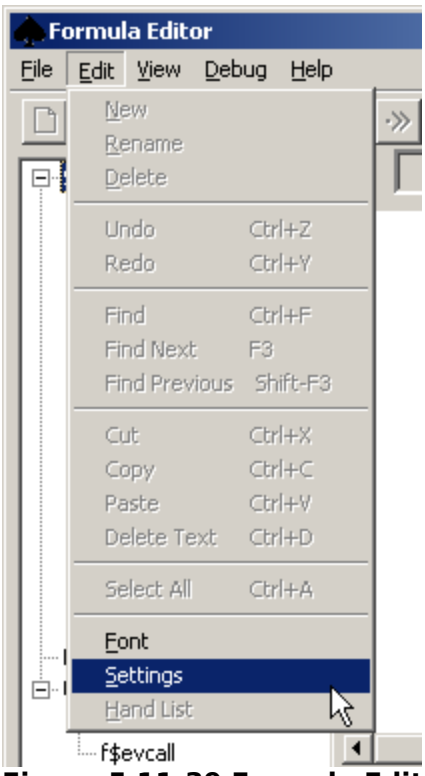


Figure 5.11-39 Formula Editor - Edit menu

The Edit menu has a number of options:

- New, Rename, Delete: These options are enabled/disabled depending on the function group or function that is selected. For example, User Defined Functions (UDFs) are only able to be created (with “New”) in the user Defined Functions group, and Standard functions cannot be deleted or renamed. If the Hand List group is selected, these options will all you to create, rename or delete Hand Lists as well. OpenHoldem supports 1000 different hand lists, numbered from list0 to list999.
- Undo, Redo, Find, Cut, Copy, Paste, Delete Text, Select All: These are all standard Windows menu items that operate in the standard way.
- Font: This menu option opens the standard Windows font selector dialog to allow you to select the font to display in the Scintilla editor window.
- Settings: This opens a window that allows you to set the Bankroll, Defcon, NIT, and Rake parameters of the formula file. These settings can be used in your formulas, and are referenced by some symbols.
- Hand List: When a Hand List is selected, this will open the Hand List Editor window so you can specify which starting hands belong in the selected Hand List. See the {Hand Lists} section for more information.

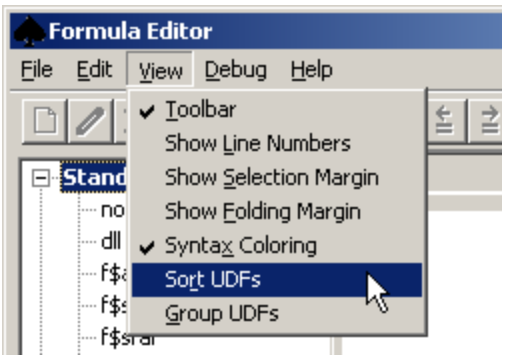


Figure 5.11-40 Formula Editor - View menu

The View menu has the standard Windows toolbar option, plus several OpenHoldem specific options:

- Show Line Numbers, Show Selection Margin, Show Folding Margin, and Syntax Coloring provide access to features of the Scintilla Control. See http://www.scintilla.org for more information.
- Sort UDFs and Group UDFs allow for grouping and sorting of User Defined Functions that you create. These options simply allow for ease of navigation, especially for formulas with large numbers of UDFs.

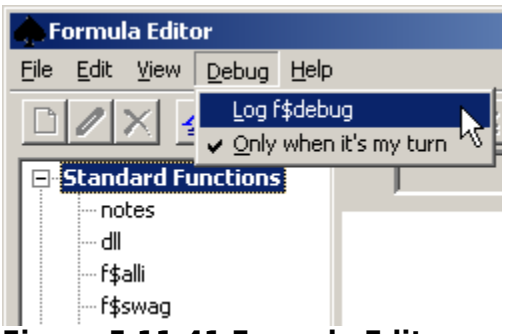


Figure 5.11-41 Formula Editor - Debug menu

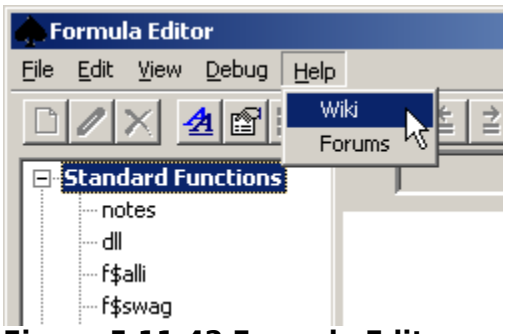


Figure 5.11-42 Formula Editor - Help menu

The Debug menu has options to control logging of the contents of the the f$debug function to a file. If the “Log f$debug” menu option is checked, and the "Auto" button is depressed, then OpenHoldem will write the entire contents of the f$debug function to a “f$debug*.log” file once per screen scrape cycle. The entire f$debug tab is written in comma-delimited format to a single line in the file. If the “Only when it’s my turn” option is checked, then the contents of f$debug are only written when the user chair has card faces and action buttons are present.

The Help menu provides links to the OpenHoldem Wiki and to the OpenHoldem discussion forums.

Toolbar

Figure 5.11-43 OpenHoldem Formula Editor toolbar

From left to right, the buttons on the Main toolbar are:

- New function/hand list (empty document)

- Rename function/hand list (pencil)
- Delete function/hand list (big X)
- Open the Windows font selection dialog (two blue letter As)
- Open the Formula Parameters window (hand pointing at a document)
- Open the Hand List Editor window (9 squares)
- Increase or decrease decimal precision (double arrow with a decimal point) – these two buttons are only valid when the f$debug function is selected, and control the number places that are displayed after the decimal point.
- Increase of decrease space for results (arrow with equal sign) – these two buttons are only valid when the f$debug function is selected, and control how much space is provided between the left hand side of the display and the equals sign, to allow more or less space to display results.

5.3 OH-Script

The Scripting Language

Introduction

OpenHoldem's scripting language is very similar to the C programming expression syntax. If you already know and understand C programming then you have an advantage in that the OH scripting language should be very familiar. This section is largely based off of Ray Bornert's original WinHoldem "C-tutorial" here: http://www.winholdem.net/help/help-c.html

Syntax

The general syntax for OpenHoldem formulas is same as that of the C programming language. However, the formulas are strict R-VALUES (right side of an assignment expression), meaning that there is no assignment operator. All values are stored as (double) floating point values on the expression stack. For bitwise operations the values are first converted to integer values and then the operator is applied.

Boolean Logic

A boolean expression is composed of logical operators (not, and, or, xor) and operands. All boolean operands have only two values - true and false. Each logical operand has a very well defined operation upon the operand(s) in the expression, with a very well defined result. When any numeric value is used in conjunction with a logical operator, any zero values are considered to be false (0) and any non-zero values are considered to be true (1). If the numeric value in question is not zero then it is considered to be true for boolean purposes.
Reams of material have been written on boolean logic. Example: http://en.wikipedia.org/wiki/Boolean_logic

Operators

These operators are listed in order of precedence

Category	**Operator(s)**	**Associativity**
Exponentiation	** ln (not standard ANSI C)	Right to Left
Unary	! ~ - `	Right to Left
Multiplicative	* / %	Left to Right
Additive	+ -	Left to Right
Bitwise Shift	<< >>	Left to Right
Relational	< > <= >=	Left to Right
Equality	== !=	Left to Right
Bitwise AND	&	Left to Right

Bitwise XOR	^	Left to Right
Bitwise OR	pipe (above forward slash on U.S. keyboard)	Left to Right
Logical AND	&&	Left to Right
Logical XOR	^^ (not standard ANSI C)	Left to Right
Logical OR	double pipe (above forward slash on U.S. keyboard)	Left to Right
Conditional	?:	Right to Left
Group	() [] {} (not standard ANSI C)	Left to Right

Exponentiation
*Power ** (not ANSI-C)*
a ** b
Standard algebraic exponentiation on a and b.
a is raised to the power of b.

Natural Log ln (not ANSI-C)
ln a
Standard algebraic natural log of a
a == e ** (ln a)

Natural Log Base e (not ANSI-C)
e == ln(1)
e == 2.71828182845905

Unary
A unary operator takes a single operand.

Logical NOT !
False when the operand is true. True when the operand is false.

a	**!a**
false	true
true	false
0	1

Bitwise NOT ~
Logical NOT operation on a bit by bit basis.

expression	**binary result**
a	11001001011101010110010101111010
~a	00110110100010101001101010000101

Negation -
A good example is the minus sign when it is used to alter the sign of a value.
Example: "a + b / -4" The minus sign in front of the 4 is a unary minus.

Bit Count ` (not ANSI-C)
Provides a count of the number of bits set in any integer value. Here are 3 examples of the number of bits that are set in some 32-bit integer numbers.

a	**`a**
00000000000000000000000000000000	0
00000000000001000000000000000000	1

00000100000001000000000000010000	3
11111111111111111111111111111111	32

Multiplicative
Multiply *
a * b
Standard algebraic multiplication on a and b.

Divide /
a / b
Standard algebraic division on a and b.

Modulo %
a % b
Standard algebraic modulo on a and b.

Additive
Add +
a + b
Standard algebraic addition on a and b.

Subtract -
a - b
Standard algebraic subtraction on a and b.

Bitwise Shift
Bitwise Shift Left <<
Slides the entire bit pattern to the left by N bits. Note that the leftmost bits are simply dropped and that the rightmost bits are filled with 0's. The shift magnitude is used as modulo 32, meaning that any shift N that is specified in excess of 32 bits has a N%32 operation performed prior to the shift.

expression	**binary result**
a	11001001011101010110010101111010
a<<1	10010010111010101100101011110100
a<<7	10111010101100101011110100000000
a<<31	00000000000000000000000000000000
a<<32	11001001011101010110010101111010

Bitwise Shift Right >>
Slide the entire bit pattern to the right by N bits. Note that the rightmost bits are dropped and that the leftmost bits are filled with the leftmost bit of the operand. The shift magnitude is used as modulo 32, meaning that any shift N that is specified in excess of 32 bits has a N%32 operation performed prior to the shift.

expression	**binary result**
a	11001001011101010110010101111010
a>>1	11100100101110101011001010111101
a>>7	11111111100100101110101011001010
a>>31	11111111111111111111111111111111
a>>32	11001001011101010110010101111010
a	01001001011101010110010101111010
a>>1	00100100101110101011001010111101

a>>7	00000000100100101110101011001010
a>>31	00000000000000000000000000000000
a>>32	01001001011101010110010101111010

Relational
Less Than <

a	**b**	**a < b**
-1	0	true
0	0	false
+1	0	false

Greater Than >

a	**b**	**a > b**
-1	0	false
0	0	false
+1	0	true

Less Than Or Equal <=

a	**b**	**a <= b**
-1	0	true
0	0	true
+1	0	false

Greater Than or Equal >=

a	**b**	**a >= b**
-1	0	false
0	0	true
+1	0	true

Equality
Equal ==

a	**b**	**a == b**
-1	0	false
0	0	true
+1	0	false

Not Equal !=

a	**b**	**a != b**
-1	0	true
0	0	false
+1	0	true

Bitwise AND &
Logical AND operation on a bit by bit basis.

expression	**binary**
a	11001001011101010110010101111010
b	01001010010100101110001010101111

a&b	01001000010100000110000000101010

Bitwise OR |
Logical OR operation on a bit by bit basis.

expression	**binary**
a	11001001011101010110010101111010
b	01001010010100101110001010101111
b	11001011011101111110011111111111

Bitwise XOR ^
Logical XOR operation on a bit by bit basis.

expression	**binary**
a	11001001011101010110010101111010
b	01001010010100101110001010101111
a^b	10000011001001111000011111010101

Logical AND &&
False when any operand is false. True when both operands are true.

a	b	a && b
false	false	false
false	true	false
true	false	false
true	true	true

Logical XOR ^^ (not ANSI-C)
False when operands are boolean equal. True when operands are not boolean equal. Note that "a^^b" is equivalent to "(a!=0)^(b!=0)".

a	b	a ^^ b
false	0	0
false	0	0
true	0	0
true	0	0

Logical OR ||
False when both operands are false. True when any operand is true.

a	**b**	**a || b**
false	false	false
false	true	true
true	false	true
true	true	true

Conditional ?:

Standard algorithmic if then else. "a ? b : c" means "If a then b else c"

a	**b**	**c**	**a ? b : c**
true	any	any	b
false	any	any	c

Grouping Operators ()[]{}
Note that [] and {} are not ANSI-C.

These grouping operators are used to either visibly separate sections of your code for readability and maintainability purposes, or to affect the precedence of logical operation. Note that unlike WinHoldem's grouping syntax, there is no limitation on how the various grouping operators can be nested.

Numeric Constants
Floating point constants

All numeric constants are treated internally as double floating point values in base 10.

Floating Point Numeric Constants
123.456
0.987
192837465
.5
17.
5.4321e-76

Integer constants
There are 4 integer options available as well that allow you to select the base of the constant. The 4 available bases are: 16, 8, 4, 2. Prefixing a numeric constant with a zero followed by a letter (see table below) will specify the numeric base of the constant.

Examples:

Decimal	**Hex**	**Octal**	**Quadal**	**Binary**
0	0x0	0o0	0q0	0b0
1	0x1	0o1	0q1	0b1
2	0x2	0o2	0q2	0b10
3	0x3	0o3	0q3	0b11
4	0x4	0o4	0q10	0b100
5	0x5	0o5	0q11	0b101
6	0x6	0o6	0q12	0b110
7	0x7	0o7	0q13	0b111
8	0x8	0o10	0q20	0b1000

9	0x9	0o11	0q21	0b1001
10	0xa	0o12	0q22	0b1010
11	0xb	0o13	0q23	0b1011
12	0xc	0o14	0q30	0b1100
13	0xd	0o15	0q31	0b1101
14	0xe	0o16	0q32	0b1110
15	0xf	0o17	0q33	0b1111
16	0x10	0o20	0q100	0b10000
31	0x1f	0o37	0q133	0b11111
63	0x3f	0o77	0q333	0b111111
127	0x7f	0o177	0q1333	0b1111111
255	0xff	0o377	0q3333	0b11111111

Calculated Symbols

This section is largely based on the information originally written by Ray Bornert and extended to include OpenHoldem-specific symbols. See here for the original WinHoldem symbols page: http://www.winholdem.net/help/help-formula.html#symbols

These symbols take output from the scraper engine, from Table Map parameters, and from your formula functions to derive various bits of data that are useful when instructing the OpenHoldem Autoplayer how to act.

General

Symbol	**Meaning**
ismanual	true if you're in manual mode, false otherwise
isppro	true if you're connected to a ppro server
site	0=user/ppro 1=scraped
nchairs	the integer value for the Table Map symbol s$nchairs
isbring	true if OpenHoldem is attached to a Bring client window
session	the current logging instance (0-9)
handnumber	the site hand number if available
version	returns the version number of OpenHoldem that is currently running

Table Map

Symbol	**Meaning**
sitename$abc	true if user defined string “abc” appears within the Table Map symbol s$sitename
network$def	true if user defined string “def” appears within the Table Map symbol s$network
swagdelay	Autoplayer delay in milliseconds between swag keystrokes and button click
allidelay	Autoplayer delay in milliseconds between alli slider jam and button click
swagtextmethod	the site interpretation for swag edit text (Table Map symbol) 1=f$srai 2=f$srai+call 3=f$srai+call+currentbet

Formula file

Symbol	Meaning
rake	percentage amount added/subtracted to/from the pot
nit	number of iterations used to calculate the prwin/prtie/prlos and associated symbols. For an analysis of how many iterations to use, see this section: {Iterations and Standard Deviation}
bankroll	the user defined, real world bankroll

Limits

Symbol	Meaning
bblind	the big blind amount
sblind	the small blind amount
ante	the current pre-deal ante requirement
lim	the current table limit 0=NL 1=PL 2=FL
isnl	(lim==0)
ispl	(lim==1)
isfl	(lim==2)
sraiprev	the difference between the two largest unique wagers
sraimin	Scraped - (currentbet+call); PokerPro - the greater of bet and the current raise
sraimax	balance-call
istournament	true if a tournament table is detected

Hand Rank

Symbol	Meaning
handrank	one of the following based on the option specified in preferences
handrank169	your pocket holdem hand rank 1-169
handrank2652	your pocket holdem hand rank 12-2652
handrank1326	your pocket holdem hand rank 6-1326 (handrank2652/2)
handrank1000	your pocket holdem hand rank 4-1000 (1000*handrank2652/2652)
handrankp	2652 / (1+nopponents)

Chairs

Symbol	Meaning
chair	your chair number 0-9 ... 0 is usually top right
userchair	user chair number (0-9)
dealerchair	dealer chair number (0-9)
raischair	raising chair number (0-9)
chair$abc	player “abc” chair number (0-9); -1 if not found
chairbit$abc	player “abc” chairbit (1 << chair$abc); 0 if not found

Rounds / Positions

Symbol	Meaning

betround	betting round (1-4) 1=preflop, 2=flop, 3=turn, 4=river
br	abbreviation for betround
betposition	your bet position (1=sblind,2=bblind,...,nplayersdealt=dealer); betposition will change as players fold in front of you.
dealposition	your deal position (1=sblind,2=bblind ... nplayersdealt=dealer); dealposition will not change as players fold
originaldealposition	a memory of deal position which retains its value through the hand, even if the user folds
callposition	your numbered offset from the raising player (who is 0)
seatposition	your seat position relative to the dealer
dealpositionrais	the deal position of the raising player (1-10)
betpositionrais	the bet position of the raising player (1-10)

Probabilities

Symbol	**Meaning**
prwin	the probability of winning this hand (0.000 - 1.000)
prlos	the probability of losing this hand (0.000 - 1.000)
prtie	the probability of pushing this hand (0.000 - 1.000)
prwinnow	probability that all opponents have a lower hand right now
prlosnow	probability that any opponents have a higher hand right now
random	random number between (0.000-1.000). Value is recalculated each time symbol appears in formula.
randomhand	random number between (0.000-1.000) for the hand. Value is calculated only once per hand.
randomround	random number between (0.000-1.000) for the current round. Value is calculated only once in current round.
randomround1 - randomround4	random number between (0.000-1.000) for round 1 - 4. Value is calculated only once in that round.

Statistics

The symbols to calculate risk of ruin have never been implemented due to lack of documentation and lack of practical relevance. **As of OpenHoldem 2.0.0 the risk-of-ruin-symbols are officially abolished and removed from the code-base.**
The original documentation for these symbols can be found here:
WinHoldEm Help - Formula Editor
WinHoldem-Forum: Risk Of Ruin

Formulas

Symbol	**Meaning**
f$name	reference the specified standard or user defined function

f$P Formula

Symbol	**Meaning**
defcon	defense level is typically used in the f$P formula and determines the number of analyzer opponents (0.000=maxoffense 1.000=maxdefense) (the Formula Editor parameters dialog uses values 0-10)
isdefmode	true when defcon is at max
isaggmode	true when defcon is at min

Chip Amounts

Symbol	Meaning
balance	your balance
balance0 - balance9	specific player/chair balance
stack0 - stack9	sorted playersplaying balances from 0=biggest to 9=smallest
currentbet	your current amount of chips in play
currentbet0 - currentbet9	specific player/chair currentbet
call	the amount you need to call
bet	the amount of a single initial bet or raise for current round
bet1 - bet4	the amount of a single initial bet or raise for round 1 - round 4
pot	the total amount of chips in play including player bets
potcommon	the total amount of chips in the middle
potplayer	the total amount of chips in front of all players
callshort	total amount that will be added to the pot if all players call
raisshort	callshort + bet * nplayersplaying

Number of Bets

Symbol	Meaning
nbetstocall	total number of additional bets required to call.
nbetstorais	total number of additional bets required to raise.
ncurrentbets	total number of bets currently in front of you.
ncallbets	total number of bets you would have on the table if you call
nraisbets	total number of bets you would have on the table if you raise

List Tests

Symbol	Meaning
islist0 - islist999	true if your hand is in the numbered (0-999) list
islistcall	true if your hand is in list 0
islistrais	true if your hand is in list 1
islistalli	true if your hand is in list 7
isemptylistcall	true if the call list is empty
isemptylistrais	true if the rais list is empty
isemptylistalli	true if the alli list is empty
nlistmax	highest list number in which your hand is listed
nlistmin	lowest list number in which your hand is listed

Poker Values

Symbol	Meaning
pokerval	absolute poker value for your 5 card hand
pokervalplayer	absolute poker value for your 2 card pocket hand only
pokervalcommon	absolute poker value for the common cards
pcbits	bit list of where your pocket cards are used in your 5 card hand
npcbits	number (0-2) of your pocket cards used in your 5 card hand

Poker Value Constants

Symbol	Meaning
hicard	1<< 0 (2 ** 0)
onepair	1<<24 (2 ** 24)
twopair	1<<25 (2 ** 25)
threeofakind	1<<26 (2 ** 26)
straight	1<<27 (2 ** 27)
flush	1<<28 (2 ** 28)
fullhouse	1<<29 (2 ** 29)
fourofakind	1<<30 (2 ** 30)
straightflush	1<<31 (2 ** 31)
royalflush	0x800edcba
fiveofakind	0xff000000

Hand Tests

Symbol	Meaning
$CCc	references a hand symbol - see the {Hand Symbols} section below for details
$$AB#	references a card symbol - see the {Card Symbols} section below for details
ishandup	true if your hand has gone up a level (i.e. from 1 pair to 2 pair)
ishandupcommon	true if common hand has gone up a level (i.e. from 1 pair to 2 pair)
ishicard	true when you have hicard hand
isonepair	true when you have one pair
istwopair	true when you have two pair
isthreeofakind	true when you have three of a kind
isstraight	true when you have a straight
isflush	true when you have a flush
isfullhouse	true when you have a full house
isfourofakind	true when you have four of a kind
isstraightflush	true when you have a straight flush
isroyalflush	true when you have a royal flush
isfiveofakind	true when you have a five of a kind

Pocket Tests

Symbol	Meaning
ispair	true when your two dealt pocket cards are rank equal (0-1)
issuited	true when your two dealt pocket cards are suit equal (0-1)
isconnector	true when your two dealt pocket cards are rank adjacent (0-1)

Pocket / Common Tests

Symbol	Meaning
ishipair	true when you have hi pair (0-1)
islopair	true when you have lo pair (0-1)
ismidpair	true when you have mid pair (0-1)

ishistraight	true when you have the highest straight possible
ishiflush	true when you have the highest flush possible

Players, Friends, Opponents

Note that the "friends" symbols are not interpreted the same way as they are in WinHoldem. OpenHoldem does not include any automated collusion capabilities, and thus "friends" has been redefined in OpenHoldem to mean “you” only. Thus if you are seated, nfriendsseated will resolve to 1. If you are not seated, nfriendsseated will resolve to zero.

Symbol	**Meaning**
nopponents	f$P function value for the Iterator
nopponentsmax	maximum allowable value for nopponents (1-22 default=9)
nplayersseated	number of players seated (including you) (0-10)
nplayersactive	number of players active (including you) (0-10)
nplayersdealt	number of players dealt (including you) (0-10)
nplayersplaying	number of players playing (including you) (0-10)
nplayersblind	number of players blind (including you) (0-10)
nfriendsseated	1 if you are seated, 0 otherwise (0-1)
nfriendsactive	1 if you are active, 0 otherwise (0-1)
nfriendsdealt	1 if you are dealt, 0 otherwise (0-1)
nfriendsplaying	1 if you are playing, 0 otherwise (0-1)
nfriendsblind	1 if you are in a blind, 0 otherwise (0-1)
nopponentsseated	number of opponents seated (not including you) (0-9)
nopponentsactive	number of opponents active (not including you) (0-9)
nopponentsdealt	number of opponents dealt (not including you) (0-9)
nopponentsplaying	number of opponents playing (not including you) (0-9)
nopponentsblind	number of opponents blind (not including you) (0-9)
nopponentschecking	number of opponents playing with a zero current bet equal to the previous bettor (0-9)
nopponentscalling	number of opponents playing with a non-zero current bet equal to the previous bettor (0-9)
nopponentsraising	number of opponents playing with a current bet greater than the previous bettor (0-9)
nopponentsbetting	number of opponents playing with a non zero current bet (0-9)
nopponentsfolded	number of opponents that have folded this hand (0-9)
nplayerscallshort	number of players that must call to stay in the hand
nchairsdealtright	number of chairs dealt before your chair
nchairsdealtleft	number of chairs dealt after your chair
playersseatedbits	bits 9-0: 1=seated 0=unseated
playersactivebits	bits 9-0: 1=active 0=inactive
playersdealtbits	bits 9-0: 1=dealt 0=notdealt
playersplayingbits	bits 9-0: 1=playing 0=notplaying
playersblindbits	bits 9-0: 1=blind 0=notblind
opponentsseatedbits	bits 9-0: 1=seated 0=unseated
opponentsactivebits	bits 9-0: 1=active 0=inactive
opponentsdealtbits	bits 9-0: 1=dealt 0=notdealt

opponentsplayingbits	bits 9-0: 1=playing 0=notplaying
opponentsblindbits	bits 9-0: 1=blind 0=notblind
friendsseatedbits	bits 9-0: 1=seated 0=unseated, you only
friendsactivebits	bits 9-0: 1=active 0=inactive, you only
friendsdealtbits	bits 9-0: 1=dealt 0=notdealt, you only
friendsplayingbits	bits 9-0: 1=playing 0=notplaying, you only
friendsblindbits	bits 9-0: 1=blind 0=notblind, you only

Flags

Symbol	**Meaning**
fmax	highest numbered flag button pressed
f0 - f19	true if flag 0 - flag 19 button is pressed, false otherwise
fbits	flag button bits 19-0 - 1=pressed 0=notpressed

Common Cards

Symbol	**Meaning**
ncommoncardspresent ncommoncardsknown	number of common cards present
nflopc	short for ncommoncardsknown

(Un)known Cards

Symbol	**Meaning**
nouts	the total number of unseen single cards that if dealt to the board might put your hand in the lead. to be counted as an out, the card must be able to bump your level and your new level must be higher than the resulting common level
ncardsknown	total number of cards you can see (yours and commons)
ncardsunknown	total number of cards you cannot see (deck and opponents)
ncardsbetter	total number of single unknown cards that can beat you, e.g. if the board is four suited in hearts, and you have two spades, then ncardsbetter will be at least 9, because of the possible flush

nhands

Symbol	**Meaning**
nhands	total possible number of two-card hands using the unseen cards (nhandshi+nhandslo+nhandsti)
nhandshi	number of hands that can beat you in a showdown right now
nhandslo	number of hands that you can beat in a showdown right now
nhandsti	number of hands that can tie you in a showdown right now

Flushes / Straights / Sets

Symbol	**Meaning**
nsuited	total number of same suited cards you have (1-7)
nsuitedcommon	total number of same suited cards in the middle (1-5)
tsuit	specific card suit for nsuited (1-4)
tsuitcommon	specific card suit for nsuitedcommon (1-4)
nranked	total number of same ranked cards you have (1-4)

nrankedcommon	total number of same ranked cards in the middle (1-4)
trank	specific card rank for nranked (2-14)
trankcommon	specific card rank for nrankedcommon (2-14)
nstraight	total number of connected cards you have (1-7)
nstraightcommon	total number of connected common cards (1-5)
nstraightfill	total number of cards needed to fill a straight (0-5)
nstraightfillcommon	total number of cards needed to fill a common straight (0-5)
nstraightflush	total number of suited connected cards you have (1-7)
nstraightflushcommon	total number of suited connected common cards (1-5)
nstraightflushfill	total number of cards needed to fill a straightflush (0-5)
nstraightflushfillcommon	total number of cards needed to fill a common straightflush (0-5)

Rank Bits (aces are hi and lo)

Symbol	**Meaning**
rankbits	bit list of card ranks (yours and commons)
rankbitscommon	bit list of card ranks (commons)
rankbitsplayer	bit list of card ranks (yours)
rankbitspoker	bit list of card ranks (pokerval)
srankbits	bit list of suited card ranks (yours and commons tsuit)
srankbitscommon	bit list of suited card ranks (commons tsuitcommon)
srankbitsplayer	bit list of suited card ranks (yours tsuit)
srankbitspoker	bit list of suited card ranks (pokerval tsuit)

Rank Hi (aces are hi)

Symbol	**Meaning**
rankhi	highest card rank (14-2) (yours and commons)
rankhicommon	highest card rank (14-2) (commons)
rankhiplayer	highest card rank (14-2) (yours)
rankhipoker	highest card rank (14-2) (pokerval)
srankhi	highest suited card rank (14-2) (yours and commons tsuit)
srankhicommon	highest suited card rank (14-2) (commons tsuitcommon)
srankhiplayer	highest suited card rank (14-2) (yours tsuit)
srankhipoker	highest suited card rank (14-2) (pokerval tsuit)

Rank Lo (aces are hi)

Symbol	**Meaning**
ranklo	lowest card rank (14-2) (yours and commons)
ranklocommon	lowest card rank (14-2) (commons)
rankloplayer	lowest card rank (14-2) (yours)
ranklopoker	lowest card rank (14-2) (pokerval)
sranklo	lowest suited card rank (14-2) (yours and commons tsuit)
sranklocommon	lowest suited card rank (14-2) (commons tsuitcommon)
srankloplayer	lowest suited card rank (14-2) (yours tsuit)

sranklopoker	lowest suited card rank (14-2) (pokerval tsuit)

Time

Symbol	**Meaning**
elapsed	time in seconds since sitting down
elapsedhand	time in seconds since end of previous hand
elapsedauto	time in seconds since autoplayer took action
elapsedtoday	time in seconds since midnight GMT
elapsed1970	time in seconds since 1970-01-01 00:00:00 GMT (Thursday)
clocks	number of CPU clocks since the last screen scrape
nclockspersecond	number of CPU clocks per second
ncps	number of CPU clocks per second

Autoplayer

Symbol	**Meaning**
myturnbits	bits 43210 correspond to buttons KARCF (check alli rais call fold). (bit 4 (check) was added in OpenHoldem 2.0)
ismyturn	(myturnbits & 7) (rais or call/chec or fold)
issittingin	true when you are not being dealt out
issittingout	true when you are being dealt out
isautopost	true when you are autoposting
isfinalanswer	true when autoplayer preparing to act; false any other time.

History

Symbol	**Meaning**
nplayersround1 - nplayersround4	number of players that began betting round 1 - round 4
nplayersround	number of players that began the current betting round
prevaction	record of previously attempted autoplayer action. (-1=fold 0=chec 1=call 2=rais 3=swag 4=alli)
didchec	the number of times the autoplayer has checked during the current round
didcall	the number of times the autoplayer has called during the current round
didrais	the number of times the autoplayer has raised during the current round
didswag	the number of times the autoplayer has swag'd during the current round
nbetsround1 - nbetsround4	the largest number of bets in front of any player during round 1-round 4
nbetsround	the largest number of bets in front of any player right now
didchecround1 - didchecround4	true if userchair checked during round 1 - round 4
didcallround1 - didcallround4	true if userchair called during round 1 - round 4
didraisround1 - didraisround4	true if userchair raised during round 1 - round 4

didswaground1 - didswaground4	true if userchair swag'd during round 1 - round 4

RON / RUN

These symbols report the total number of possible river endings for the opponent (ron$) and the user (run$). A value of zero means that type of poker hand is not possible. Any non-zero value means that type of poker hand will be seen that many times.

Symbol	**Meaning**
ron$royfl	river opponent number : possible royal flush
ron$strfl	river opponent number : possible straight flush
ron$4kind	river opponent number : possible four of a kind
ron$fullh	river opponent number : possible full house
ron$flush	river opponent number : possible flush
ron$strai	river opponent number : possible straight
ron$3kind	river opponent number : possible three of a kind
ron$2pair	river opponent number : possible two pair
ron$1pair	river opponent number : possible one pair
ron$hcard	river opponent number : possible high card
ron$total	river opponent number : sum of all possible river endings
ron$pokervalmax	the maximum possible pokerval for the opponent
ron$prnuts	opponent chances of hitting the nuts on or before the river
ron$prbest	opponent chances of hitting pokervalmax on or before the river
ron$clocks	total number of cpu clocks used to calculate the ron$ symbols
run$royfl	river user number : possible royal flush
run$strfl	river user number : possible straight flush
run$4kind	river user number : possible four of a kind
run$fullh	river user number : possible full house
run$flush	river user number : possible flush
run$strai	river user number : possible straight
run$3kind	river user number : possible three of a kind
run$2pair	river user number : possible two pair
run$1pair	river user number : possible one pair
run$hcard	river user number : possible high card
run$total	river user number : sum of all possible river endings
run$pokervalmax	the maximum possible pokerval for the user
run$prnuts	user chances of hitting the nuts on or before the river
run$prbest	user chances of hitting pokervalmax on or before the river
run$clocks	total number of cpu clocks used to calculate the run$ symbols

Versus

Symbol	Explanation
vs$nhands	Total possible number of opponent hands
vs$nhandshi	Number of opponent hands that have higher river chances
vs$nhandsti	Number of opponent hands that have equal river chances
vs$nhandslo	Number of opponent hands that have lower river chances
vs$prwin	Probability (0.000 - 1.000) of winning versus all possible opponent hands
vs$prtie	Probability (0.000 - 1.000) of chopping versus all possible opponent hands
vs$prlos	Probability (0.000 - 1.000) of losing versus all possible opponent hands
vs$prwinhi	Probability (0.000 - 1.000) of winning versus higher opponent hands
vs$prtiehi	Probability (0.000 - 1.000) of chopping versus higher opponent hands
vs$prloshi	Probability (0.000 - 1.000) of losing versus higher opponent hands
vs$prwinti	Probability (0.000 - 1.000) of winning versus equal opponent hands
vs$prtieti	Probability (0.000 - 1.000) of chopping versus equal opponent hands
vs$prlosti	Probability (0.000 - 1.000) of losing versus equal opponent hands
vs$prwinlo	Probability (0.000 - 1.000) of winning versus lower opponent hands
vs$prtielo	Probability (0.000 - 1.000) of chopping versus lower opponent hands
vs$prloslo	Probability (0.000 - 1.000) of losing versus lower opponent hands
vs$x$prwin	Probability (0.000 - 1.000) of winnning versus hand list x
vs$x$prlos	Probability (0.000 - 1.000) of losing versus hand list x
vs$x$prtie	Probability (0.000 - 1.000) of a tie versus hand list x

History

Symbol	Explanation
hi_<sym>x (x=1-4)	*the value of the symbol <sym> as of your last turn in betting round x. Example: hi_prwin1 would return prwin as of your last turn in br1.

Valid values for <sym> are:
PROBABILITIES: prwin, prlos, prtie
CHIP AMOUNTS: balance, balance0, balance1, balance2, balance3, balance4, balance5, balance6, balance7, balance8, balance9, stack0, stack1, stack2, stack3, stack4, stack5, stack6, stack7, stack8, stack9
POKER VALUES: pokerval, pokervalplayer, pokervalcommon, pcbits, npcbits
HAND TESTS: ishandup, ishandupcommon, ishicard, isonepair, istwopair, isthreeofakind, isstraight, isflush, isfullhouse, isfourofakind, isstraightflush, isroyalflush, isfiveofakind
POCKET/COMMON TESTS: ishipair, islopair, ismidpair, ishistraight, ishiflush
(UN)KNOWN CARDS: nouts, ncardsbetter
NHANDS: nhands, nhandshi, nhandslo, nhandsti, prwinnow, prlosnow
FLUSHES SETS STRAIGHTS: nsuited, nsuitedcommon, tsuit, tsuitcommon, nranked, nrankedcommon, trank, trankcommon, nstraight, nstraightcommon,

nstraightfill, nstraightfillcommon, nstraightflush, nstraightflushcommon, nstraightflushfill, nstraightflushfillcommon

RANK BITS: rankbits, rankbitscommon, rankbitsplayer, rankbitspoker, srankbits, srankbitscommon, srankbitsplayer, srankbitspoker

RANK HI: rankhi, rankhicommon, rankhiplayer, rankhipoker, srankhi, srankhicommon, srankhiplayer, srankhipoker

RANK LO: ranklo, ranklocommon, rankloplayer, ranklopoker, sranklo, sranklocommon, srankloplayer, sranklopoker

run$ ron$: ron$royfl, ron$strfl, ron$4kind, ron$fullh, ron$strai, ron$3kind, ron$2pair, ron$1pair, ron$hcard, ron$total, ron$pokervalmax, ron$prnuts, ron$prbest, run$royfl, run$strfl, run$4kind, run$fullh, run$strai, run$3kind, run$2pair, run$1pair, run$hcard, run$total, run$pokervalmax, run$prnuts, run$prbest

Action

Symbol	**Explanation**
lastraisedx (x=1-4)	which chair was the last to raise in round x
raisbitsx (x=1-4)	which chairs raised in round x
callbitsx (x=1-4)	which chairs called in round x
foldbitsx (x=1-4)	which chairs folded in round x
oppdealt	trailing indicator for nopponentsdealt
ac_aggressor	which chair was aggressor (might be from previous round)
ac_agchair_after	does the aggressor chair act after me?
ac_preflop_pos	preflop position of the userchair (SB=1 BB=2 Early=3 Middle=4 Late=5 Dealer=6)
ac_prefloprais_pos	preflop position of the raiser (SB=1 BB=2 Early=3 Middle=4 Late=5 Dealer=6)
ac_postflop_pos	postflop position of the userchair (first=1 early=2 middle=3 late=4 last=5)
ac_pf_bets	1 for no callers or blinds only 2 for Called Pot - 1 bet to call 3 for Raised Back - 1 more bet to call because someone behind you raised after you've already bet/called/raised 4 for Raised Pot - 2 bets to call 5 for Reraised Pot - 3+ bets to call NOTE: Only valid when br==1
ac_first_into_pot	returns true if you are first into the pot (first to act or checked to you)
ac_betposx (x=0-9)	returns bet position of specified chair
ac_dealposx (x=0-9)	returns deal position of specified chair

MyHand
Basically, the mh_str symbols (e.g. mh_str_twopair) return a relative ranking of your hand (hole and common) strength from 1 to 5. A Wiki article on this topic can be found here: http://www.maxinmontreal.com/wiki/index.php5?title=OpenHoldem:EndUserDocumentation:Symbols:mh_str_explained

Symbol	Explanation
mh_3straightxy	(x=1 for wheel, 0 not, y=1 for broadway, 0 not) - returns true if the board has a wheel straight draw or broadway straight draw, given the wheel/broadway parameters 00 - true if the board has neither a wheel straight draw nor a broadway straight draw 10 - true if the board has a wheel straight draw 01 - true if the board has a broadway straight draw 11 - true if the board has either a wheel straight draw or a broadway straight draw
mh_bottomsd	true if I have a bottom straight draw (if you are contributing a single card to an open-ended straight draw and that card is the smallest, this symbol is true e.g. hole: T2 common: 345K)
mh_nsuitedbetter	number of missing suited cards that are higher than my best suited card
mh_kickerbetter	number of cards that can beat your kicker
mh_kickerrank	rank of your kicker (returns 0 if kicker is shared [board] and thus useless)
mh_nouts	number of outs (HTC's formula)
mh_str_strflush	0-5 (5 best) of the relative strength of your straight flush
mh_str_quads	0-5 (5 best) of the relative strength of your four of a kind
mh_str_fullhouse	0-5 (5 best) of the relative strength of your full house
mh_str_flush	0-5 (5 best) of the relative strength of your flush
mh_str_straight	0-5 (5 best) of the relative strength of your straight
mh_str_trips	0-5 (5 best) of the relative strength of your three of a kind
mh_str_twopair	0-5 (5 best) of the relative strength of your two pair
mh_str_onepair	0-5 (5 best) of the relative strength of your one pair

Table statistics
The setting for "y minutes" is set Preferences, and defaults to 15 minutes.

Symbol	Explanation
floppct	percentage of players seeing the flop for the last y minutes
turnpct	percentage of players seeing the turn for the last y minutes
riverpct	percentage of players seeing the river for the last y minutes
avgbetspf	average number of bets preflop for the last y minutes
tablepfr	pfr percentage preflop for the last y minutes
maxbalance	my highest balance during the session
handsplayed	number of hands played this session
balance_rankx (x=0-9)	ranked list of player balances (includes players not currently in hand, and includes currentbet for each player as well). rank0 has highest balance.

Logging

Symbol	Explanation
log$YourTextGoesHere	When executed, adds your specified text to the log. For example if you have: *[br == 2 && (nsuited == 3 || nstraight == 3) && log$ItLives]* in your f$rais formula and you flop a backdoor flush draw, then OpenHoldem will raise and also add the text "ItLives" to the log file, just before the RAIS line. Everytime a log$ symbol is accessed it will be flagged for logging. You can use this to track which part of your formula is to blame for the action taken by OpenHoldem. Note: You want to use the log$ symbol at the **end** of a logic block, to make sure it only gets flagged when all the previous statements are true.

Note: You have to turn on Log Symbols in the preferences. You can also dictate the maximum number of log symbols that will be logged per action. When enabled, up to 4 log symbols will be displayed in OpenHoldem's main window.

ICM calculator

An overview of basic ICM push/fold decisions and how they work with OpenHoldem can be found here: http://www.maxinmontreal.com/wiki/index.php5?title=ICM

Symbol	Explanation
icm	my tournament equity before any action is considered (just balances)
icm_fold	my tournament equity if I fold
icm_callwin	my tournament equity if I call and win
icm_calllose	my tournament equity if I call and lose
icm_calltie	my tournament equity if I call and tie
icm_alliwin0 - icm_alliwin9	my tournament equity if I push all-in and win against 0 - 9 callers
icm_allilose1 - icm_allilose9	my tournament equity if I push all-in and lose against 0 - 9 callers

Hand multiplexor

The purpose of the hand multiplexor is to transfer control to a specific named formula based on your actual 2 card hand. Note that the X's and x's are not case sensitive.

Symbol	Explanation
f$$X	Evaluate the UDF that corresponds to my first card
f$$XX	Evaluate the UDF that corresponds to my first card and second card
f$$XXx	Evaluate the UDF that corresponds to my first card, second card and suited/unsuited state

The X's in the three symbols will be substituted with your actual hand values at time of evaluation. The first "X" will contain the the rank of your highest hole card, the second "X" will contain the rank of your lowest hole card, and the third "x" will contain either a "s" or "o" depending if your hole cards have the same suit or not.
The $$ will be replaced with a single dollar sign.
For example, if my hole cards are AhKh, then "f$$X" will result in "f$A" upon evaluation. Similarly, "f$$XX" would result in "f$AK" and "f$$XXx" would result in "f$AKs".

OpenHoldem will then evaluate the corresponding user defined function and return that value for the hand multiplexor symbol. If there is not a corresponding UDF defined for a given set of hole cards, then the return result is zero, not an error. The values returned by the hand specific UDFs are entirely defined by you; one simple suggestion is to have them return 1 for call and 2 for raise; if you do this then your raise formula can do this:

```
|| [ f$$XXx >= 2 ] //hand specific formula calc
```

Memory Symbols

The purpose of these symbols is to provide storage for values in order to reuse them at some later time. In reality, this is a kludge to close a hole in the WinHoldem (and now OpenHoldem) script language. A much better choice for LHS (left hand side) expressions is Perl or via the user-DLL extension.
Please see the {Memory Symbols} section for more information.

Symbol	**Meaning**
me_st_	Stores a value. Example: me_st_abc_123_45 - stores the value "123.45" in variable "abc". use "_" for the decimal in values a function name can also be passed in, instead of a number, for example: 'me_st_def_f$myfunc' would store the results of function f$myfunc in variable def
me_re_	Retrieves a previously stored value. Example: me_re_abc - retrieves the value from variable "abc".

Hand Symbols

You can reference your dealt hand (hole cards) directly by using the $ symbols. The general form of the $ hand symbols are: $RRs. The $ character is required, followed by 1 or 2 standard card rank characters. The hand symbols are not case sensitive. A,K,Q,J,T,9-2 or by X or x which is a wild card that will match any rank. This is followed by an optional suit indicator: s - suited, o - offsuit. If the suit indicator is omitted then suit does not matter.

Examples

Symbol	**Meaning**
$AA	true if you were dealt aces
$JT	true if you were dealt jack ten
$AKs	true if you were dealt ace king suited
$49o	true if you were dealt four nine offsuit
$QXs	true if you were dealt queen any suited
$J	true if you were dealt any jack
$XXs	true if you were dealt any suited hand
$XXo	true if you were dealt any offsuited hand
$q2	true if you were dealt queen two

Card Symbols

The actual card value or rank/suit value for both your hand and the board can be referenced using card symbols. The general form of the $$ card symbols are: $$AB#. The card symbols are not case sensitive.
$$ is required.
A - is p for player, c for common
B - is c for card, r for rank, s for suit
- is the card number (player 0-1, common 0-4)

Examples

Symbol	**Meaning**
$$pc0	the card value for player card zero (1st dealt card).
$$pr1	the rank value of player card one (2nd dealt card).
$$cr0	the rank value of common card zero (1st common card dealt).
$$cs4	the suit value of common card four (5th common card dealt - river).

Card Values

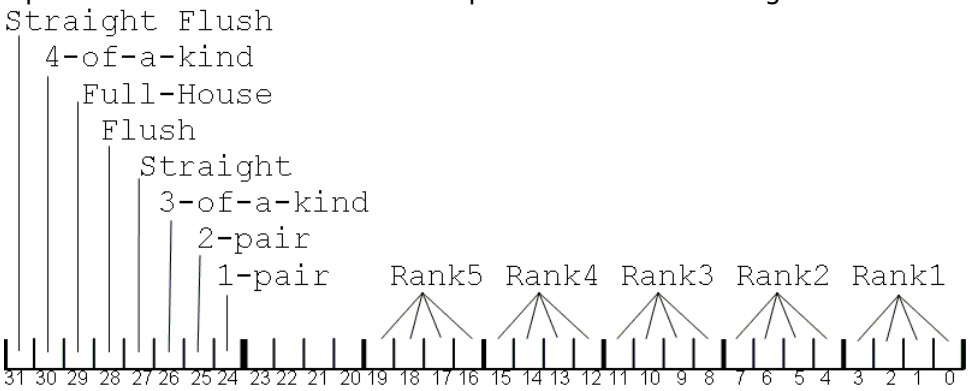
A single card value is stored in a single 8-bit byte. All bits (7-0) are used. The card rank is stored in the hi-order 4 bit nibble (bits 7-4). The card suit is stored in the lo-order 4 bit nibble (bits 3-0). The valid values for card rank are 1-14 as follows:

#	**CardRank**
14	Ace
13	King
12	Queen
11	Jack
10-2	Ten-Two
1	Ace (in lo straight)

#	**CardSuit**
4	Spade
3	Heart
2	Diamond
1	Club

Poker Value

OpenHoldem classifies and orders poker hands into a single 32bit value.



Bits 31-24 indicate the hand rank
- Straight flush = bit 31 set
- Four of a kind = bit 30 set
- Full house = bit 29 set
- Flush = bit 28 set
- Straight = bit 27 set
- Three of a kind = bit 26 set
- Two pair = bit 25 set
- One pair = bit 24 set
- 5-of-a-kind is indicated when bits 31-24 are set to 1
- High-card (no pair) is indicated when bits 31-24 are set to 0

Bits 23-20 are not used

Bits 19-16 indicate the rank of the top card in the hand

Bits 15-12 indicate the rank of the second card in the hand

Bits 11-8 indicate the rank of the third card in the hand

Bits 7-4 indicate the rank of the fourth card in the hand

Bits 3-0 indicate the rank of the fifth card in the hand

Numeric values for the ranks (bits 19-1) are:

#	**Rank**
14	Ace

13	King
12	Queen
11	Jack
10-2	Ten-Two
1	Ace (in lo straight)

The pokerval symbol maps every 0 to 7 card poker hand onto the 32bit number space in correct order of game precedence, allowing two poker hands can be compared to see which is the better hand - higher values are better. The lowest possible value is 0 (no cards). The highest possible value is 5-aces which is encoded as 0xFF0EEEEE.

Player/Pocket Card List

The symbol pcbits is a bit list that indicates where your dealt pocket cards are used in your 5 card poker hand. Ony the lower order 5 bits are significant - one bit per card in your 5 card poker hand. Each bit corresponds to a pokerval rank field as follows:

bit#	**pokerval**
bit4	rank5
bit3	rank4
bit2	rank3
bit1	rank2
bit0	rank1

If all bits in pcbits are zero then neither of your two cards are used in your 5 card hand.
The following formula fragment will be true if you have a sucker straight:

```
((nstraight==5) && (pcbits==1))
```

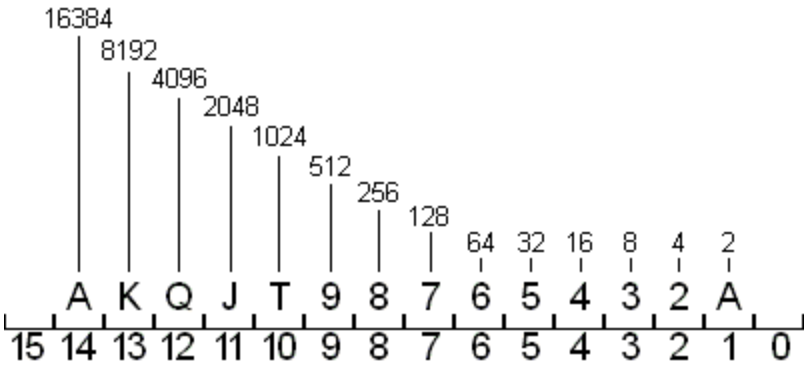
The following formula fragment will be true if you are filling an inside straight:

```
((nstraight==5) && (pcbits&14))
```

The formula symbol npcbits contains the number of your pocket cards being used in your 5 card poker hand. The possible values are: 0, 1 and 2.

Rank Bits

OpenHoldem uses the following method to classify a group of cards according to rank within a single value. Note that if an ace is present then both the 14 bit and the 1 bit will be set.



Bit 15 is not used
Bit 14 - A (16384)
Bit 13 - K (8192)
Bit 12 - Q (4096)
Bit 11 - J (2048)
Bit 10 - T (1024)
Bit 9 - 9 (512)
Bit 8 - 8 (256)
Bit 7 - 7 (128)
Bit 6 - 6 (64)
Bit 5 - 5 (32)
Bit 4 - 4 (16)
Bit 3 - 3 (8)
Bit 2 - 2 (4)
Bit 1 - 1 (2)
Bit 0 is not used

example: $JT = 3072
example: $65432 = 124
example: $KJ9753 = 10920
example: $AQT8642 = 21846

Parse-Time and Run-Time Errors

The OpenHoldem scripting language will detect various error conditions for you, and warn you of these conditions. Each time a formula is loaded, it is “parsed” for correctness – if there are any syntactical errors upon this parse, then a message box will be displayed that alerts you to the location of this error.

Additionally, several classes of run-time errors will be detected, including divide-by-zero and infinite recursion. Note that the OH-Script language does not support recursion in any form, due to the absence of the assignment operator.

5.4 Perl

The purpose of the Perl option is to provide a scripting language for beginners that is possibly easier to get productive with, if you either already familiar with Perl, or have a strong disposition for not using OH-Script.

Advantages of Perl

- Suitability for rapid prototyping
- Wide-spread usage
- Many libraries available and an active community at http://cpan.org
- Really good interfacing, e.g. to C.
- Syntax similar to C and some other script languages (bash, awk, tcl)

Disadvantages of Perl

- Performance: Perl compiles to intermediate code (similar to bytecode), which gets executed by the interpreter. This is slower than compiled binary code, however some built-in functions (e.g. string- and list-operations) are said to be faster than C.
- Clarity. Perl is a language, which bought good ideas from nearly everywhere. You can write clear and understandable programs, if you want; but most Perl "gurus" are famous for the opposite. It is quite easy to write non-maintainable obfuscated Perl code.

Installation and configuration of Perl

- Download ActivePerl from http://www.activestate.com/Products/activeperl and install
- Perl needs the package "Win32::API" to interact with the callback DLL. Download this package using Perl's package manager. Type at the command line (case sensitive; you should have administrator privileges to do this):

```
ppm install Win32::API
```

- OpenHoldem must be able to find "PerlEz.dll" which contains our embedded interpreter.
  You must change the DLL search path from the DOS prompt:

```
set PATH = C:\Programs\Perl\Bin;%PATH%
```

- You need Visual C++ 2005 redist. package from Microsoft (SP1), especially msvcrt.dll, otherwise Perl can't load correctly (you will get a message: "PerlEz.dll not found or not accessible").

Perl usage

- All symbols beginning with "pl_" (e.g. "pl_call") in your formula will be treated as Perl symbols and requested from OpenHoldem's built-in Perl interpreter. The same for gws-calls inside the dll. All symbols are expected to be either double constants, double variables or functions without parameters returning a double value.
- Perl for OpenHoldem has a built-in function "gws" (get_WinHoldem_Symbol), allowing you to request OpenHoldem symbols or DLL symbols in your Perl formulas. (Therefore you have full interoperability of OH script, DLL and Perl. You can further use all your existing code). This function "gws" resides in a package called "Perl_OH_Interaction.pm", which you have to include in your bot:

```
use Perl_OH_Interaction;
```

- The table name can be accessed using the function gwt. To get a player name for a given chair use gwp(N), where N is the chair number. Load a Perl file via the menu or using the preferences.

Perl introduction

- http://en.wikipedia.org/wiki/Perl

Perl recommended books

- Wainwright: Professional Perl Programing
- Schwartz: Learning Perl, Intermediate Perl, Mastering Perl
- Conway: Perl hacks, tips and tools for programing, debugging and surviving
- Till: Teach yourself Perl in 21 days

Perl editors

You may need an editor with syntax-highlighting for Perl.

- emacs (wonderful editor from the UNIX world; for windows and free)
  ftp://ftp.gnu.org/gnu/emacs/windows/
  Take the bin.tar.gz.
- Komodo-Edit (recommended by http://www.activestate.com)

Perl bot framework

The embedded archive below contains some files, ready to be filled with your bot logic, including:

- A classic formula, redirecting some symbols to Perl functions
- A nearly empty Perl bot, giving you a start
- The module Perl_OH_Interaction.pm, providing the "gws" function

Perl demo bot

You will find an extremely simple demo bot in the embedded archive below. To use it, you have to adapt the following lines according to your system settings:

```
use lib "C:\..."

use constant the_DebugFile => "..."
```

Perl starter files

The most recent version of this can always be found here:
http://www.maxinmontreal.com/forums/viewtopic.php?p=20129

PerlBotIntro_v6.rar

5.5 User DLL

OpenHoldem has the ability to interface with a user-compiled DLL. Microsoft Windows provides a feature whereby running code can load other code, presuming that code is provided in the form of DLL files. OpenHoldem utilizes this Windows DLL feature to allow you to extend the functions within OpenHoldem to practically

anything that you want, provided you can write the code for it and compile it. Generally, people who create DLLs write them in the C or C++ languages, as those are the native historical target languages for using DLL functions, as defined by Microsoft. There are ways to have OpenHoldem use a C/C++ DLL shell which then calls code written in other languages, such as managed languages like .NET, but that topic is beyond the scope of this manual. If you are interested in doing this, there are plenty of programming forums on the Internet that can help with this activity.

OpenHoldem provides a number of preferences for how a user-DLL will be loaded. It can be done on start-up, when a formula is loaded, or manually. See the {Preferences} section for more information.

From this point on, the assumption is that, if you are creating a user-DLL, it will be written in the C or C++ languages. To do this, you need a C/C++ compiler – the express editions of Microsoft Visual Studio will suffice for this, as these user-DLLs typically do not need the MFC library, which is only present in the paid versions of Microsoft Visual Studio. A starting framework for a user-DLL can be found in the source code library on Google Code, but is also embedded just below this paragraph. See the {References} section for links.


Reference User DLL Source Code.rar

How OpenHoldem calls DLL functions

The OpenHoldem to DLL interface is small and clean. OpenHoldem will send or request information from the DLL when needed, and it is up to the DLL's author to process these requests. The exported library function process_message is used for this interface and has the following prototype:

```
USERDLL_API double process_message (const char* pmessage, const
void* param)
```

The various messages that can be passed in the "pmessage" parameter are as follows:

pmessage	**meaning and parameters**
state	Each scrape cycle, OpenHoldem will send the current game state to the DLL. As the author of the DLL, you will decide what to do with this game state information. The game state information is stored in a structure within OpenHoldem, and a pointer to this structure is passed to your User DLL in the "param" parameter. An example of how to cast this void pointer to the game state structure is provided in the Reference User DLL source code. The details of what is included in this game state structure are in the header file in the Reference User DLL source code.
query	Any symbol that begins with the characters "dll$" is interpreted by the OpenHoldem parser as requiring a call to the loaded User DLL in order to determine its value. The "query" message is the mechanism for requesting the value of this symbol. Any symbol name can be created for

	use by your User DLL, as long as it begins with “dll$”, and your DLL code knows how to handle it. So, for example, if the OpenHoldem decision engine encounters a symbol called “dll$kill_phil” in the evaluation of the “f$alli” function, then a synchronous call will be made to the User DLL to provide the value of dll$kill_phil in order to complete the evaluation of f$alli. The evaluation of f$alli will block until your DLL handles this request and provides a result. The name of the symbol (in this case, “dll$kill_phil”) is provided in the “param” parameter, and should be cast to a (const char*). A simple example of how to process this message is provided in the Reference User DLL source code.
pfgws	Immediately after your User DLL is loaded, OpenHoldem will send a “pfgws” message once, which contains a pointer to a function within OpenHoldem that allows you to retrieve OpenHoldem symbols for use in your DLL code. “pfgws” is an acronym for “Pointer to Function for Get Winholdem Symbol”. This has not been renamed to something different for legacy compatibility reasons. The pointer to the OpenHoldem “get symbol” function is provided in the “param” parameter, and should be cast to the p_getsym_t typedef and stored in a variable provided by your DLL. From then on, any time your DLL needs to know the value of one of the OpenHoldem calculated symbols, call this function. The Reference User DLL source code provides a function that shows how to use the parameter provided in this message. The prototype for this function is: `double getsym( int chair, const char* name, bool& iserr )` This sample function takes the chair and the symbol name as input parameters and returns an error code in the “iserr” output parameter. An error will be returned if, for example, the “name” parameter contains an invalid OpenHoldem symbol name.
event	There are two event messages that are passed to your DLL. Both of these messages have NULL “param” parameters. - “load” – this message is sent once immediately after your DLL is loaded. Typically this is used for some kind of initialization activity by your DLL. - “unload” – this message is sent once just prior to OpenHoldem unloading your DLL from memory. This could be due to OpenHoldem shutdown, manual intervention from the DLL menu, or new formula loading. Typically this is used to clean up your DLL’s activities, such as connections to databases, closure of threads, etc.
phl1k	This message provides a pointer to OpenHoldem’s internal hand list array, for your DLL to read and modify as needed. The “param” parameter contains a pointer to this array and

	should be cast to the following type: `bool inlist[1000][13][13]` OpenHoldem can store 1000 different hand lists, and the first dimension of this array specifies the hand list number, the second and third dimensions specify the rank of the first and second card, where rank0>=rank1 is interpreted to mean suited cards, and rank0<rank1 is interpreted to mean unsuited cards. This is a very sparsely populated array, where the intersection of the three dimensions specifies if the particular card pair is present in the hand list or not. A non-zero value at this intersection indicates that the card pair is present; zero indicates that the card pari is not present.
prw1326	This message provides a pointer to a prw1326 structure stored in OpenHoldem. This prw1326 structure is used to describe the manner in which OpenHoldem calculates its prwin, prtie and prlos symbols, based on the information which the DLL developer places in this structure. Unless the DLL code sets a specific enabling flag in this structure, the current prwin, prtie and prlos calculation is unchanged. Please see the {prw1326} section for more information.
p_send_chat_message	This message provides a pointer to a SendChatMessage function in OpenHoldem that allows you to send any chat message you want at any time. The OH-Script only version of f$chat only allows certain pre-defined chat messages to be sent. This message is sent once immediately after your DLL is loaded, with the pointer to the SendChatMessage function in the “Param” parameter. This parameter should be cast to the following prototype: `void SendChatMessage(char* new_message)` Usage of this function is specified in more detail in the {f$chat} section.

The order that these messages are passed after the DLL is loaded is as follows:

- event / load
- pfgws
- phl1k
- prw1326
- p_send_chat_message

Note: Basic code examples for handling most of these DLL messages are provided in the Reference User DLL source code.

A Simple Example

An easy way to get started is to override all OH-Script processing with DLL functions. There are a number of reasons to do this, including:

- Limitations of the OH-Script language (no assignments is the biggest limitation)
- Speed of compiled code versus interpreted code
- Low level integration with other technologies (your own tracking database, for example)

To override all OH-Script processing, simply instruct each primary function to only have a call to a “dll$” symbol. For example, in f$alli, use the following code:

```
dll$alli
```

In f$rais, use the following code:

```
dll$rais
```

And so on. In your User DLL, handle each of these “dll$” symbols as described above. Now all of the decision logic has been moved from OH-Script code to code that you provide in your DLL. It is important to remember that even if you want to code up most of your bot logic in C/C++, you still must use the OpenHoldem primary functions themselves to call this logic.

prw1326

The prw1326 structure and associated individual chair structure have the following specification:

```
//prwin 1326 chair structure
struct sprw1326_chair
{
    int     level;         //indicates weighting level for 'always consider'
    int     limit;         //max index into weight array - used for computational efficiency
    int     ignore;        //if non-zero no weighting will be applied to this chair
    int     rankhi[1326];  //higher ranked card number in pocket cards
    int     ranklo[1326];  //lower ranked card number in pocket cards
    int     weight[1326];  //the significance value for this hand
    double  scratch;       //for future reference
};

//prwin 1326 structure
struct sprw1326
{
    int     useme;         //unless set to 1326 the normal OH prwin will be used
    int     preflop;       //unless set to 1326 the normal OH prwin will be used pre-flop
    int     usecallback;   //unless set to 1326 the callback function will not be invoked
    double  (*prw_callback)(void); //if enabled will be invoked before the prwin calculation pass
    double  scratch;       //for future reference
    int     bblimp;        //if non-zero no weighting will be applied if a chair has put nothing
in the pot
    sprw1326_chair vanilla_chair;    //will be precalculated by OH at startup - convenience
values
    sprw1326_chair  chair[10];  //structures for each chair
};
```

To use the structure from a User DLL, the following would also need to be added:

```
typedef sprw1326* pp13;
pp13 prw1326;
```

Also, in the process_message handling add:

```
if (strcmp(pmessage,"prw1326")==0)
{
      prw1326 = (pp13)param;
      return 0;
}
```

During OpenHoldem initialization the prw1326.vanilla_chair structure is set up so that .level is 1024, the .ranklo and .rankhi arrays contain the card order used for the normal OpenHoldem prwin calculation (but expanded from 169 to 1326 format), the weighting of the high third of these cards is set to 1024, the middle third is sloped down from 1024 to zero, and .limit is set so that the final third are never even considered as possible opponent cards. The .vanilla_chair is then copied to each of the ten .chair structures. This should mean that even if the capability is activated by setting .useme to 1326, the prwin calculation will not crash and will give an anodyne weighted result. The vanilla_chair is not referenced or used by OpenHoldem other than at start-up, and could be altered and used by the DLL developer to act as a default profile for new players.

A more complete explanation of how the prw1326 approach works can be found in the {Weighted prwin} section.

isfinalanswer

isfinalanswer is a calculated OpenHoldem symbol that is really only of use to User DLL developers. This symbols is true (non-zero) just prior to the Autoplayer acting, and false (zero) any other time. If Y scrapes in a row are identical (Y is set in Edit->Preferences->Autoplayer->Frame Delay) then when dll$iswait, dll$alli, dll$swag, dll$rais, dll$call are called, *isfinalanswer* is true. Unless dll$iswait returned zero, in that case *isfinalanswer* is false. More information on dll$iswait here: {dll$iswait}.

dll$iswait

Prior to the OpenHoldem Autoplayer taking action, it will query your User DLL, if one is loaded, for the value of *dll$iswait*. If this query returns non-zero, then OpenHoldem will not execute the Autoplayer's action this scrape cycle, but rather will skip the action and continue to the next screen scrape cycle. While *dll$iswait* returns non-zero, *isfinalanswer* will always be zero.

In this post: http://www.maxinmontreal.com/forums/viewtopic.php?f=174&t=6885, Matrix says:

> While dll$iwait is true don't have to respond to formula requests. You do have to somehow make sure that by the time dll$iswait is true that OpenHoldem has been returned the correct result.
> When you go down this route you may encounter the problem of OpenHoldem stop evaluating the formulas, because there has been no change in state and dll$iswait is false. The best thing to do in this case is to disable all caching inside of OpenHoldem and always return a

valid formula result and set dll$iswait to false on the very first valid frame with buttons you see from OpenHoldem.

cmd$recalc

A User DLL also has the ability to send the request "cmd$recalc", which will force OpenHoldem to recompute the versus tables and restart the Iterator thread.

5.6 Hand Lists

Hand List Editor

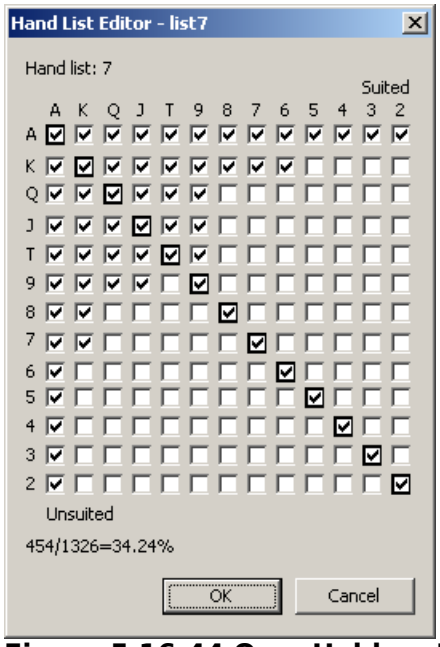


Figure 5.16-44 OpenHoldem Hand List Editor

The OpenHoldem Hand List editor is displayed when you click on the “9 yellow squares” toolbar button from the Formula Editor window. This provides a graphical interface for defining the Hand Lists as used within OpenHoldem for various symbols and functions. The output of this window is not all that difficult to understand, and if you choose your card pairs here, save your formula, then open the formula in a text editor, you can quite easily see the format used.

On this window, you specify which of your 169 starting hands (hole cards) are included in which Hand List. In the screen shot above, Hand List 7 was selected prior to clicking the “9 yellow squares” toolbar button. There is a diagonal group of starting hands running from top left to bottom right and identified with bolded checkboxes. These are the paired starting hands, i.e. AA, KK, JJ, 55. Above this diagonal line are checkboxes for the suited starting hands, i.e. AhKh, 3s2s, Jc7c. Below this diagonal line are checkboxes for the unsuited starting hands, i.e. AhJc, 3d2h.

When the “OK” button is clicked, the starting hand selections you have made are saved to the Hand List you selected.

Using Hand Lists

Hand lists are used by the following calculated symbols, and can be referenced in OH-Script code, or by Perl or DLL-based logic.

Symbol	**Meaning**
islist0 – islist999	true if your hand is in the numbered (0-999) list
islistcall	true if your hand is in list 0
islistrais	true if your hand is in list 1
islistalli	true if your hand is in list 7
isemptylistcall	true if the call list is empty
isemptylistrais	true if the rais list is empty
isemptylistalli	true if the alli list is empty
nlistmax	highest list number in which your hand is listed
nlistmin	lowest list number in which your hand is listed

5.7 How the “Green Circle Button” finds tables

The “Green Circle Button” on the OpenHoldem toolbar is the usual way that one connects an OpenHoldem instance to a poker window. This is done automagically, however an explanation of what happens behind the scenes is useful if you have problems connecting to your target poker window. The following steps are taken to identify which windows on your screen are valid poker tables.

Step 1: A list of all visible top level windows that have a non-blank caption is collected

Step 2: Each window in this list is compared against each Table Map that you have placed in your “scraper” directory. For each step that follows, if a non-match is detected, OpenHoldem will then immediately move on to the next window/Table Map.

Step 3: The size of the client portion of the window is compared to the *clientsize* records in the Table Map. The window is considered a match if the Table Map indicates the correct size as given by the record *clientsize*, or if the size is within the size limits given by *clientsizemin* and *clientsizemax*.

Step 4: The window’s title text is compared to the keyword text filters provided by *titletext* records in the Table Map. The window is considered a match if the free-form text in any of the *titletext* records is found in the window’s title. Additionally, if any *!titletext* records are present in the Table Map, then a window is considered a non-match if the free-form text in any of the !*titletext* records is found in the window’s title.

Step 5: Any *tablepoint* records that exist in the Table Map are then the compared to the window’s pixels. If the color of any pixel does not match its *tablepoint* record then the window is considered a non-match.

In summary, OpenHoldem will connect to a window if it matches the client size, title text and pixel colors as specified in a Table Map.

5.8 Preferences

The preferences for OpenHoldem's operation can be accessed via the Edit/Preferences menu item from the main window. A description of each preference follows.

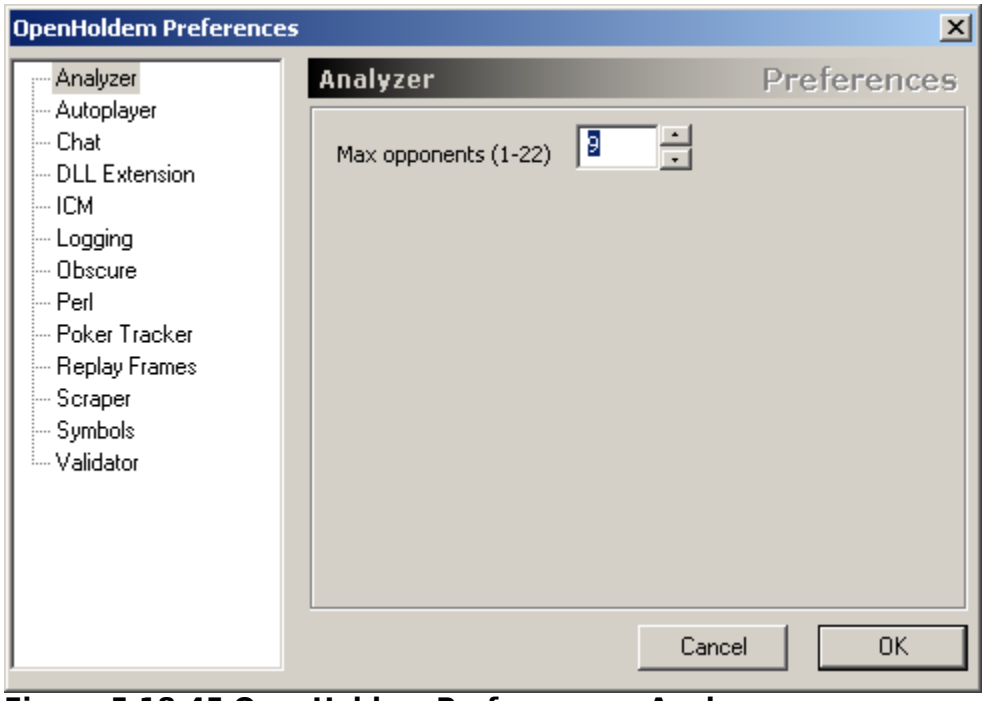


Figure 5.18-45 OpenHoldem Preferences – Analyzer

Analyzer preferences control how the OpenHoldem Iterator functions. At the moment there is only one preference available here. See the {The Iterator} section for more information.

- Max opponents: The OpenHoldem Iterator uses the value of the f$P function as the number of opponents in its Monte Carlo simulation. The maximum value that f$P can return is set here, any value of f$P greater than this will result in the number specified in this preference being used instead.

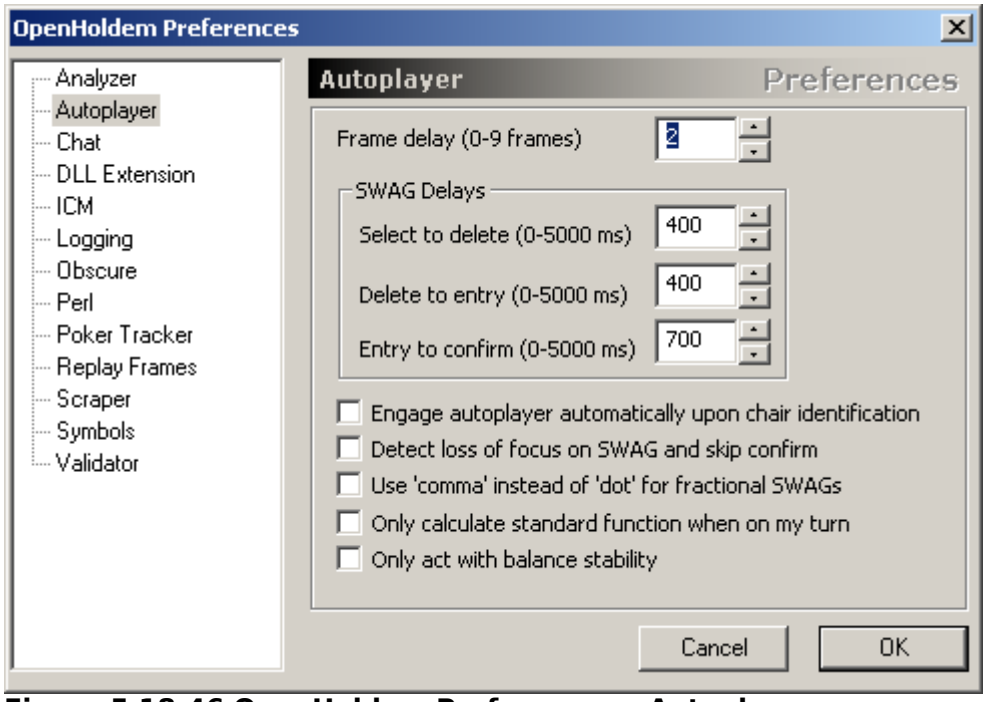


Figure 5.18-46 OpenHoldem Preferences – Autoplayer

Autoplayer preferences control how the OpenHoldem Autoplayer functions. See the {The Autoplayer} section for more information.

- Frame delay: This parameter specifies how many simultaneous identical screen scrapes must occur prior to the Autoplayer taking action. Generally, you want to have a stable poker screen before your take action, and this setting will allow various animations and so forth to finish, ensuring that your poker logic has correct game state information prior to making a decision.
- SWAG Delay, select to delete: For SWAG actions, this setting determines how long (in milliseconds) that OpenHoldem will pause after performing the SWAG Select action as specified in the Table Map, and prior to performing the SWAG Delete action as specified in the Table Map. These preferences are not set in the Table Map itself, as these are highly dependent on your CPU and network capacity and utilization, and thus are largely specific to your particular computing environment.
- SWAG Delay, delete to entry: For SWAG actions, this setting determines how long (in milliseconds) that OpenHoldem will pause after performing the SWAG Delete action as specified in the Table Map and prior to performing the SWAG Entry action. These preferences are not set in the Table Map itself, as these are highly dependent on your CPU and network capacity and utilization, and thus are largely specific to your particular computing environment.
- SWAG Delay, entry to confirm: For SWAG actions, this setting determines how long (in milliseconds) that OpenHoldem will pause after performing the SWAG Entry action, and prior to performing the SWAG Confirm action as specified in the Table Map. These preferences are not set in the Table Map itself, as these are highly dependent on your CPU and network capacity and utilization, and thus are largely specific to your particular computing environment.
- Engage Autoplayer automatically…: This setting instructs OpenHoldem to automatically start the Autoplayer when it has identified the user's chair.
- Detect loss of focus on SWAG..: The SWAG action is the longest running of the poker actions, due to the need to select the text, delete it, enter the bet and

confirm it. Occasionally, focus will be lost from the table that the SWAG action is being executed on, either due to multi-tabling (maybe the casino is set up to pop the table that requires your attention to the front?), or perhaps due to you or your Automation solution clicking elsewhere when the Autoplayer was working. When this loss of focus occurs, and if this setting is checked, then OpenHoldem will cancel the current SWAG attempt and retry on the next scrape cycle.

- Use 'comma' instead of 'dot'…: Some casinos require the use of the "comma" as a decimal separator rather than the "dot". This setting will instruct OpenHoldem to use a comma if needed. Eventually this setting should be moved to the Table Map file itself, as it is casino-specific. See the {Appendix B – Future TODOs} section.
- Only calculate standard function on my turn: Normally, OpenHoldem will calculate the primary functions on every scrape cycle. This setting will instruct OpenHoldem to only calculate these primary functions when it is your turn (card faces and action buttons are present). It will also prevent the calculation of most of the CPU expensive symbols including prwin/prlos/prtie, however the handrank symbols will be calculated as these are commonly used in the f$prefold calculations. f$prefold and f$play are still evaluated regardless if it is your turn or not.
- Only act with balance stability: In previous versions of OpenHoldem, the Autoplayer would act even if you had critical misscrapes. Once of those critical misscrapes is a missing player balance, due to the balance box changing to "All-In" instead of zero, for example, when a player goes all in. Enabling this option will prevent the Autoplayer from acting when a balance misscrape is detected. If you do not require balance information in your bot logic, you can safely leave this box unchecked.

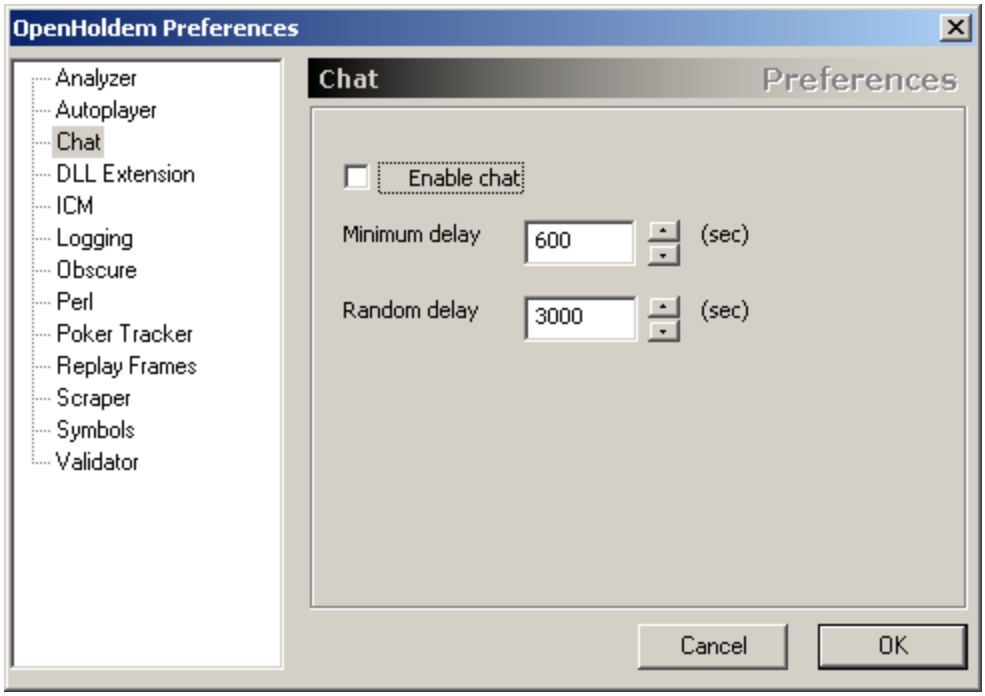


Figure 5.18-47 OpenHoldem Preferences – Chat

Chat preferences control how the OpenHoldem's Chat feature in the Autoplayer functions. See the {f$chat} section for more information.

- Enable Chat: This parameter will completely disable the Chat feature if unchecked
- Minimum delay: This is the minimum amount of time that Chat will wait (in seconds) between the posting of messages to the chat box on the Casino client. This is a safety feature and is present to prevent accidental spamming of the chat box. Feel free to set this to "1" if you are overly confident in your configuration skills.
- Random delay: In addition to the "Minimum delay", the amount of time specified in this parameter (in seconds) will be used to randomly delay the entry of a chat message into the chat box. The setting here is the maximum amount of random delay.

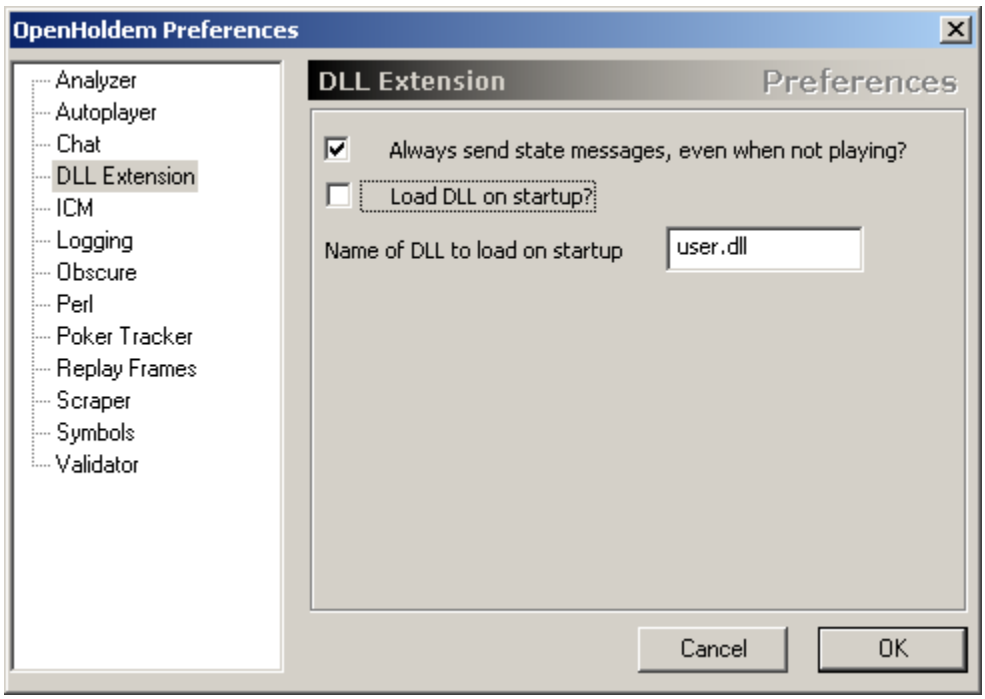


Figure 5.18-48 OpenHoldem Preferences - DLL Extension

DLL Extension preferences control how the OpenHoldem works with a User DLL. See the {User DLL} section for more information.

- Always send state messages…: By default, OpenHoldem will send "state" messages to a loaded User DLL on every scrape cycle, regardless of whether or not your player is currently in the hand. If this box is unchecked, these "state" messages will not be sent when you are "not playing". This can be useful to avoid unnecessary consumption of CPU in the DLL when not needed.
- Load DLL on startup?: If this setting is checked, OpenHoldem will try to automatically load a User DLL when the program starts up. The order in which files are tried is as follows:
    - The DLL as specified in the loaded Formula file
    - The DLL as specified in this preferences window (below)
- Name of DLL to load on startup: This specifies a DLL file to load on startup. This file is only attempted if the "Load DLL on startup" setting is checked.

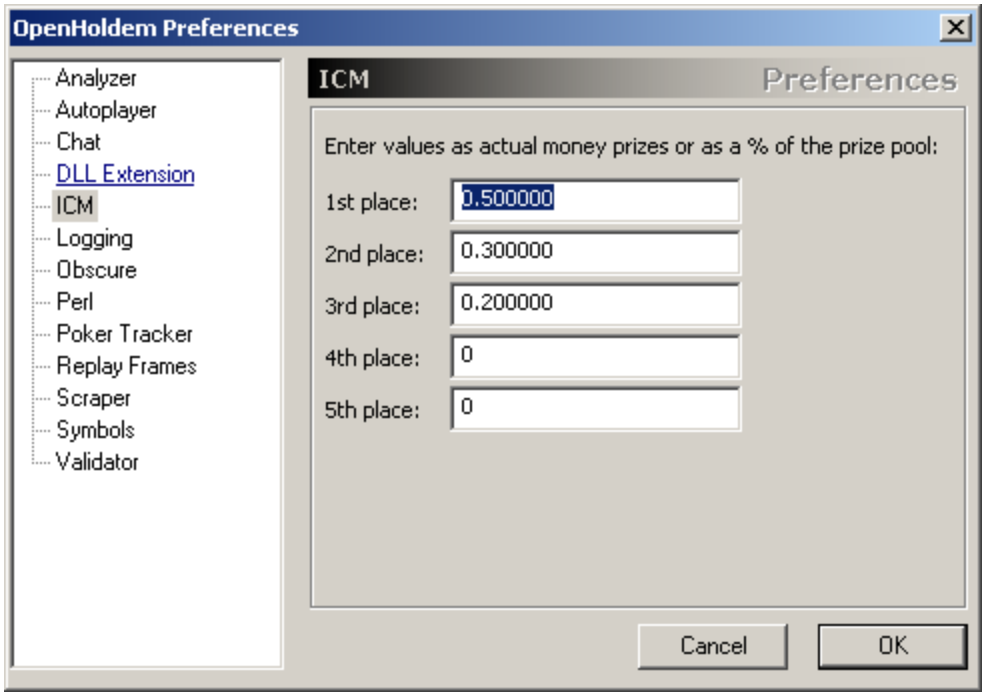


Figure 5.18-49 OpenHoldem Preferences – ICM

ICM preferences control how the OpenHoldem Independent Chip Model calculator operates. There are a plethora of resources online for understanding the Independent Chip Model. Start here:
http://www.google.com/#hl=en&q=independent+chip+model

- 1st through 5th place: These settings specify the values of 1st through 5th place of your tournament, and are directly used in the ICM Engine's calculations. Specify percentages or absolute prize values here. Percentages should add up to 100% if you want to ensure trusted results. Enter zero if a prize is not paid out for the structure of the game you are playing.

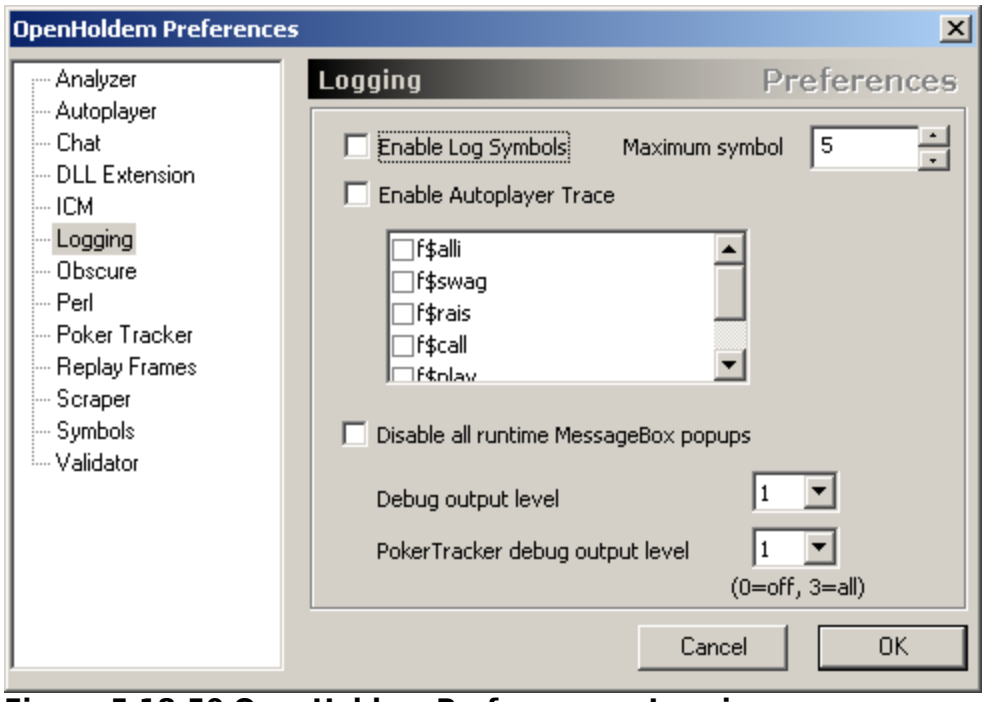


Figure 5.18-50 OpenHoldem Preferences – Logging

Logging preferences control what information OpenHoldem will periodically write to various log files.

- Enable Log Symbols: If this setting is checked, any *log$* symbols that are encountered in your OH-Script will be added to OpenHoldem's log file. Everytime a *log$* symbol is accessed it will be flagged for logging. You can use this to track which part of your formula is responsible for the action taken by OpenHoldem. You want to use the *log$* symbol at the end of a logic block, to make sure it only gets flagged when all the previous statements are true.
- Maximum symbols: This is used in conjunction with the "Enable Log Symbols" preference, and specifies the maximum number of *log$* symbols that will be logged per action. When "Enable Log Symbols" enabled, up to four *log$* symbols will be displayed in OpenHoldem's main window in the center information box.
- Enable Autoplayer Trace: This setting will enable tracing to the log file of the Autoplayer's evaluation process. The primary functions that are traced are selected in the listbox control directly below. See the {Autoplayer Trace} sub-section in the {Validator} section for more information.
- Disable all runtime MessageBox...: This preference will prevent OpenHoldem from displaying warning and error message boxes during runtime (when OpenHoldem is connected to a table). This is extraordinarily dangerous and should only be used when you have a 100% stable bot that you want to move to a fully automated execution model. It is strongly advised that you do not check this option until you have a very stable and thoroughly tested botting environment.
- Debug output level: This will control the verbosity of the output written to the log files. Zero will disable all logging, and 3 is the highest level of logging. At the time of this writing, detailed level-3 logging is available for critical parts of OpenHoldem, including the heartbeat thread, the Screen Scraper engine, and the Autoplayer engine.

- Poker Tracker debug output level: This will control the verbosity of the output written to the Poker Tracker log files. Zero will disable all logging, and 3 is the highest level of logging.

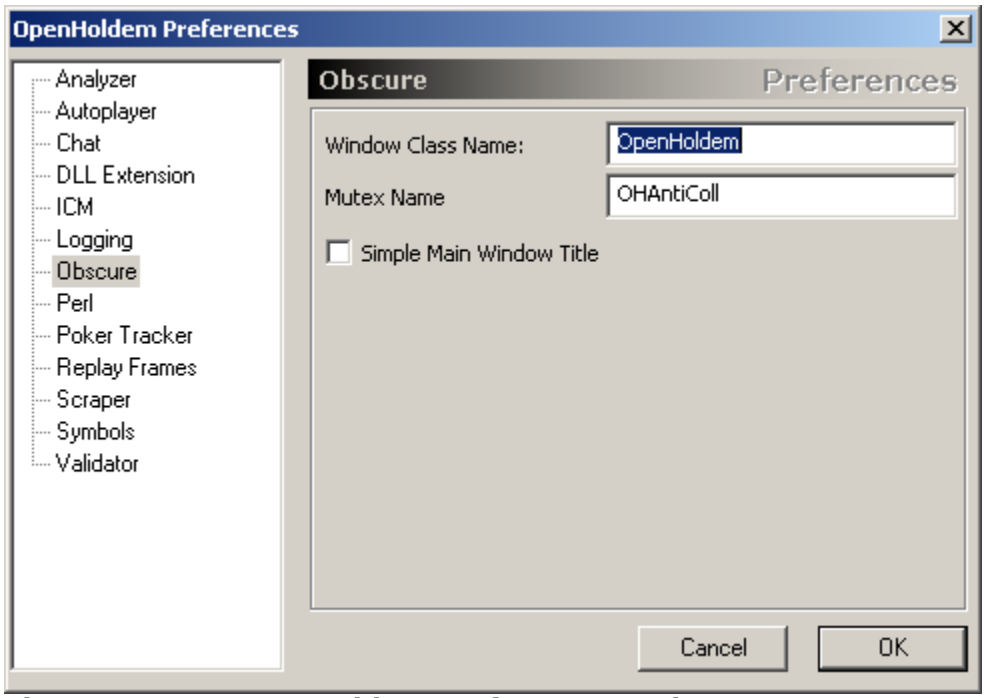


Figure 5.19-51 OpenHoldem Preferences – Obscure

Obscure preferences control various detectable components of OpenHoldem, and are primarily useful in setting up a highly stealthed environment. See the {Stealth} chapter for more information.

- Window Class Name: Every window running under Microsoft Windows has a specific "class name" associated with it. This is easily identifiable by the casinos' poker software and it is suggested that this be changed to something else.
- Mutex Name: OpenHoldem uses system-wide mutexes to ensure that multiple running instances of OpenHoldem do not interfere with each other when performing Autoplayer actions. The name of this system-wide mutex is easily identifiable by the casinos' poker software and it is suggested that this be changed to something else.
- Simple Main Window Title: By default, OpenHoldem will include all sorts of useful information in its window title bar. This useful information is also easily identifiable by the casinos' poker software. Check this preference to use a simple window title instead.

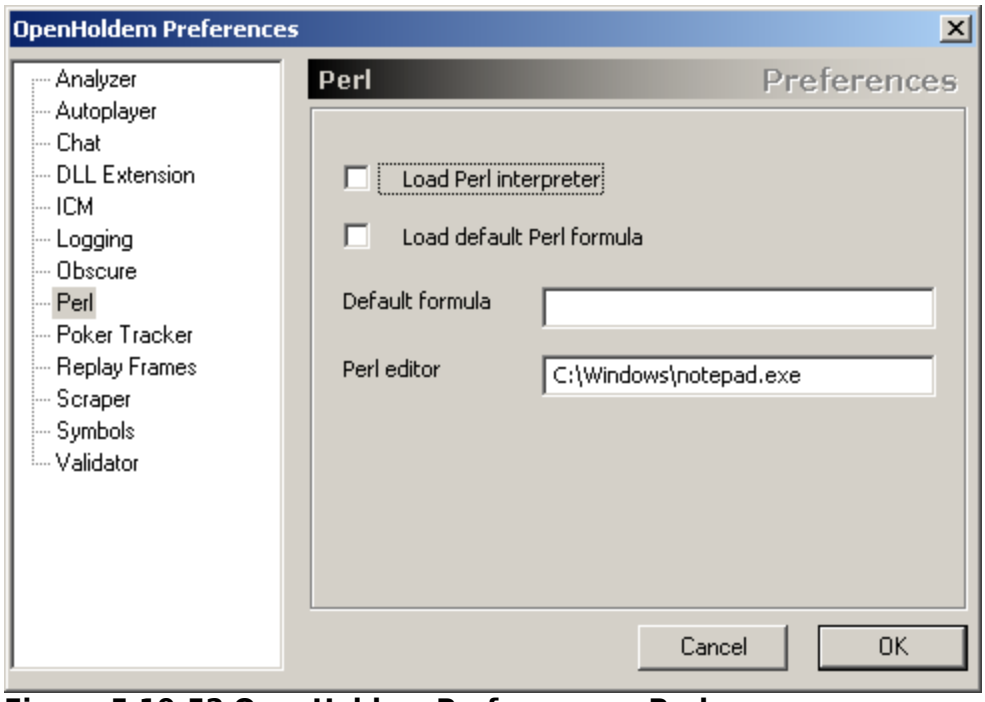


Figure 5.19-52 OpenHoldem Preferences – Perl

Perl preferences control how OpenHoldem interacts with the installed Perl interpreter and your Perl formulas. See the {Stealth} section for more information on installing and configuring Perl.

- Load Perl interpreter: When checked, this preference will cause the Perl interpreter to be loaded into memory.
- Load default Perl formula: When checked, this preference will cause the default Perl formula (as specified in "Default formula" directly below) to be loaded on startup.
- Default formula: This is the path to the Perl formula that will be loaded if the "Load default Perl formula" preference is checked.
- Perl editor: This is the path to the editor that will be launched when a Perl formula is selected for editing from the Perl menu on the main window.

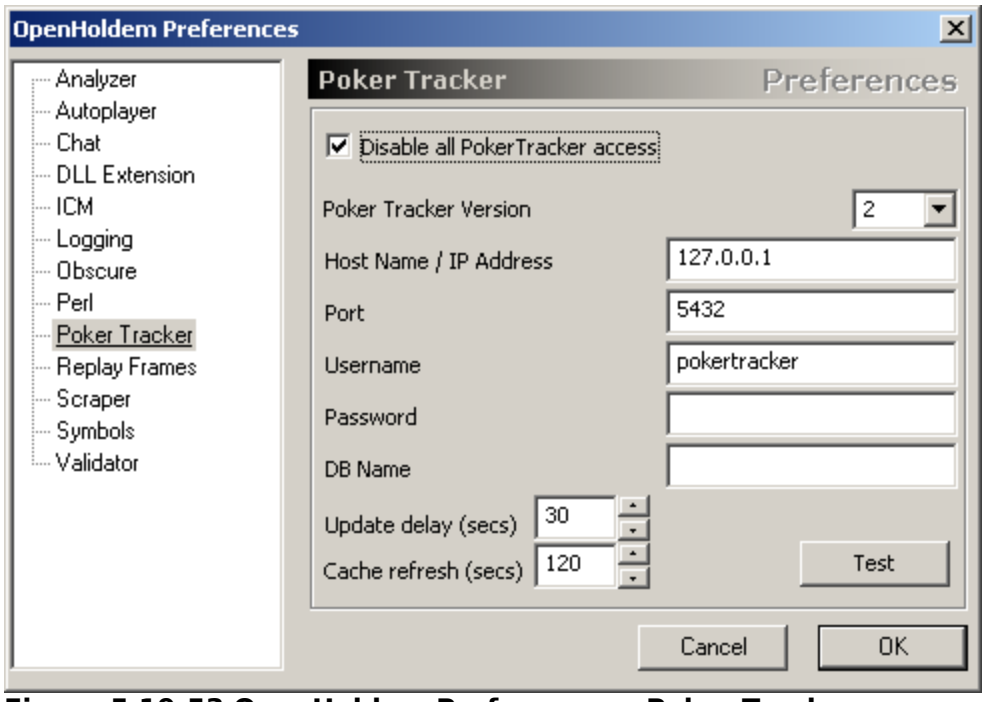


Figure 5.19-53 OpenHoldem Preferences - Poker Tracker

Poker Tracker preferences control how OpenHoldem interacts with a Poker Tracker database. See the {PokerTracker} section for more information on using Poker Tracker with OpenHoldem.

- Disable all Poker Tracker access: Checking this preference will completely disable all Poker Tracker functionality in OpenHoldem. If you make no use of any Poker Tracker symbols, then it is safe to check this box.
- Poker Tracker version: OpenHoldem supports interacting with both Poker Tracker v2 and Poker Tracker v3 databases. This setting will instruct OpenHoldem as to which flavor of Poker Tracker you are using.
- Host Name / IP Address, Port, Username, Password, DB Name: These are the parameters that you entered when you setup your Poker Tracker PostgreSQL database. Enter them here so OpenHoldem knows how to connect to your database. If you do not understand what these settings mean, please go here for more information: http://www.pokertracker.com
- Update delay: OpenHoldem maintains an internal cache of Poker Tracker statistics that it periodically queries from the Poker Tracker database. This setting determines the frequency that OpenHoldem will use (in seconds) to consider re-querying those statistics. This setting is akin to the “Scrape Delay” setting in the Scraper preferences for the screen scraper engine. This parameter does not need to be a particularly small value, as the information in Poker Tracker does not change that often itself. Note: Just because this delay has expired, and the Poker Tracker query engine “wakes up”, does not guarantee that every statistic will be updated. Rather, this works in conjunction with the “Cache refresh” parameter below to decide if a statistic needs updating or not.
- Cache refresh: After the Poker Tracker query engine has been “woken up”, because the delay specified in the “Update delay” parameter has expired, the engine will scan through each of its statistics for every chair and determine if that statistic needs updating or not. If the duration between now and the last time that a given statistic was retrieved from Poker Tracker is greater than this

parameter (in seconds), then the engine considers the statistic to be stale and will query Poker Tracker for a new value.

- Test: This button will test the connection to the Poker Tracker database as specified by the above parameters and will report success or failure codes. Help with connecting to PostgreSQL databases can be found here: http://www.pokertracker.com, or here: http://www.postgresql.org

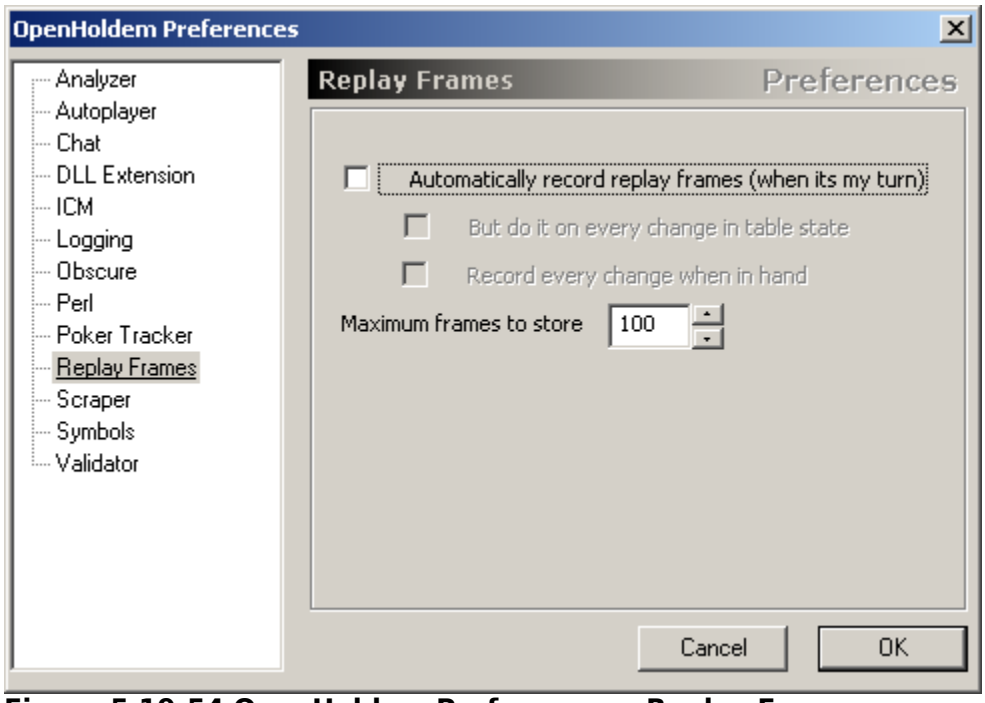


Figure 5.19-54 OpenHoldem Preferences - Replay Frames

Replay Frames preferences control how OpenHoldem collects Replay Frames for offline development and debugging use. See the {Replay Frames} section for more information on using Replay Frames.

- Automatically record replay frames: If this setting is checked, OpenHoldem will automatically record Replay Frames. The alternative is manual Replay Frame collection by using the “camera” button on the main toolbar.
- But do it on every change in table state: If the “Automatically record replay frames” preference is checked, this option will be available to instruct OpenHoldem to collect Replay Frames on every change in table state. Typically, OpenHoldem only collects Replay Frames when it is your turn to act.
- Record every change when in hand: If the “Automatically record replay frames” preference is checked, this option will be available to instruct OpenHoldem to collect Replay Frames on every screen scrape cycle when you are in the hand. Typically, OpenHoldem only collects Replay Frames when it is your turn to act.
- Maximum frames to store: OpenHoldem will collect a maximum number of Replay Frames as specified by this parameter, then it will reuse filenames starting at the beginning.

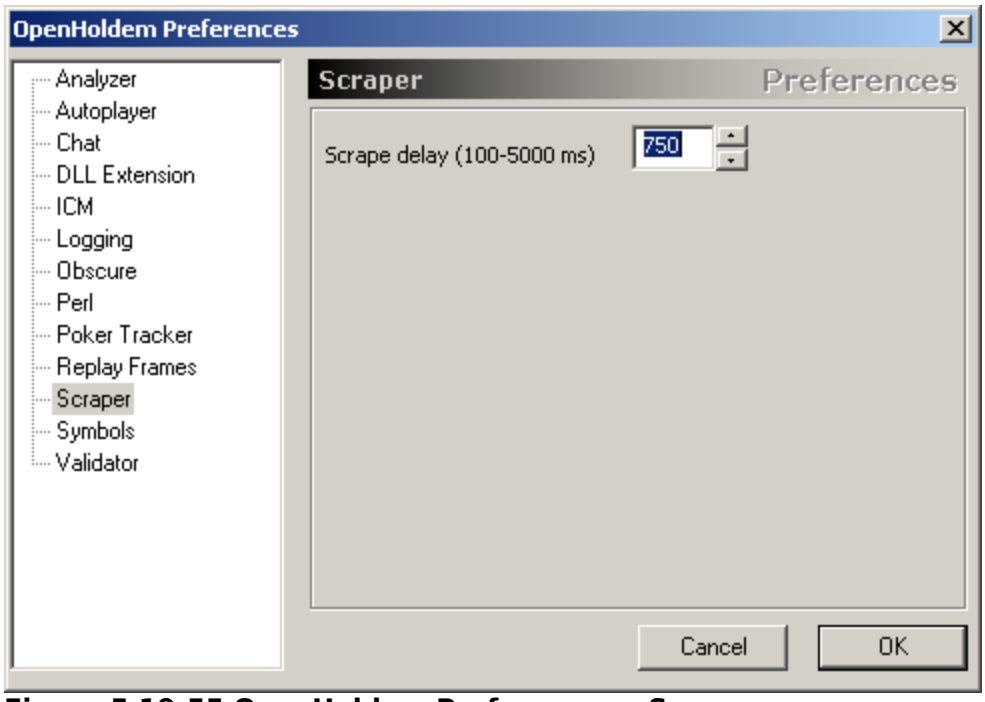


Figure 5.19-55 OpenHoldem Preferences – Scraper

Scraper preferences control how the OpenHoldem Screen Scraper engine functions. At the moment there is only one preference available here. See the {Creating a Table Map} section for more information.

- Scrape Delay: This parameter specifies the amount of time (in milliseconds) that the OpenHoldem Screen Scraper engine will pause in between subsequent screen scrape passes. This amount of time does not specify the time from the beginning of scrape 1 to the beginning of scrape 2, but rather the amount of delay between the end of scrape 1 and the beginning of scrape 2. The primary consideration for this parameter is that of CPU consumption, and the default of 750ms seems to work well for most people.

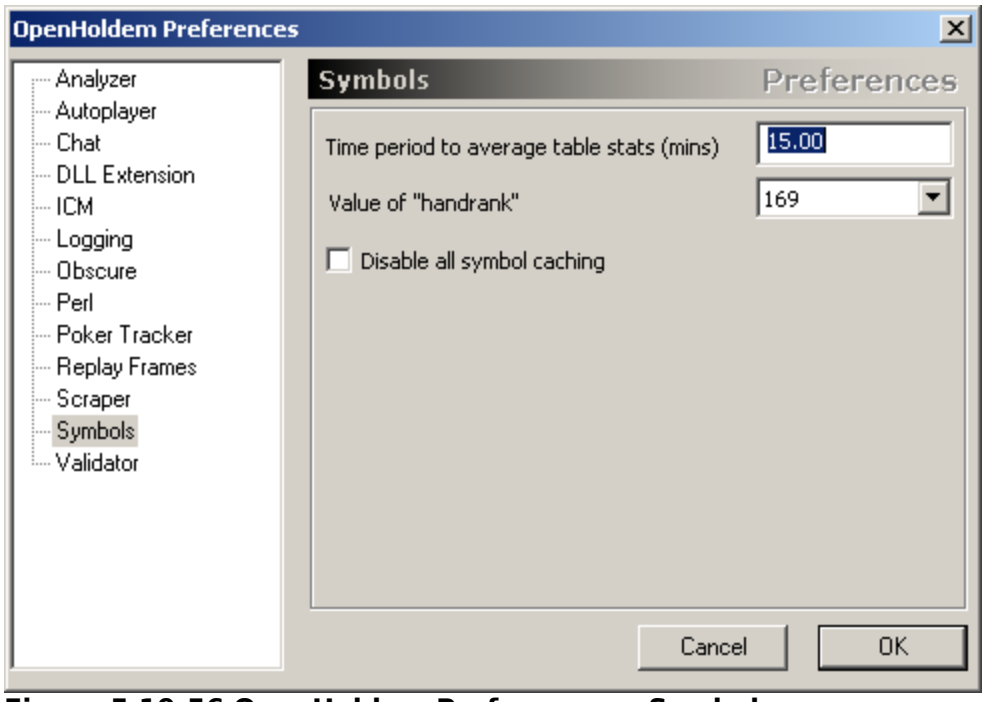


Figure 5.19-56 OpenHoldem Preferences – Symbols

Symbols preferences control various aspects of OpenHoldem's calculated symbols. See the {Calculated SymbolsCreating a Table Map} section for more information.

- Time Period to Average Table Stats: As discussed in the {Calculated Symbols} section, the Table Stats symbols (*floppct*, *turnpct*, *riverpct*, *avgbetspf* and *tablepfr*) are calculated on a rolling time average. The time frame (in minutes) specified by this parameter determines how far back in time these averages are calculated. For example, if this is set to 15 minutes, then the averages will be for the last 15 minutes of activity at the table.
- Value of "handrank": *handrank* is a calculated symbol, and is set to one of 169, 1000, 1326, 2652 or p. Depending on the option selected here, *handrank* will be set equal to the calculated *handrank169*, *handrank1000*, *handrank1326*, *handrank2652* or *handrankp* symbols. This setting also determines the hand rank value that is displayed on the main window's status bar (see {Status Bar} for more information). See section {Starting Hand } for more information on hand ranks.
- Disable all symbol caching: The normal operation for the OpenHoldem parsing and decision engine is to cache the results of each symbol that it calculates in a given scrape cycle. The reason for this, is that if a UDF is referenced 100 times in a given function calculation, it makes sense to save the results of the first calculation, and then reuse the results in the other 99 references. This is just an efficient use of CPU resources. By checking this preference, you instruct OpenHoldem to bypass this caching and guarantee that each symbol is re-calculated each time it is needed. Whether this option is checked or unchecked, *dll$* symbols are never cached, as if a caching of User DLL symbols is needed, that can easily be accommodated in the DLL itself by the DLL developer.

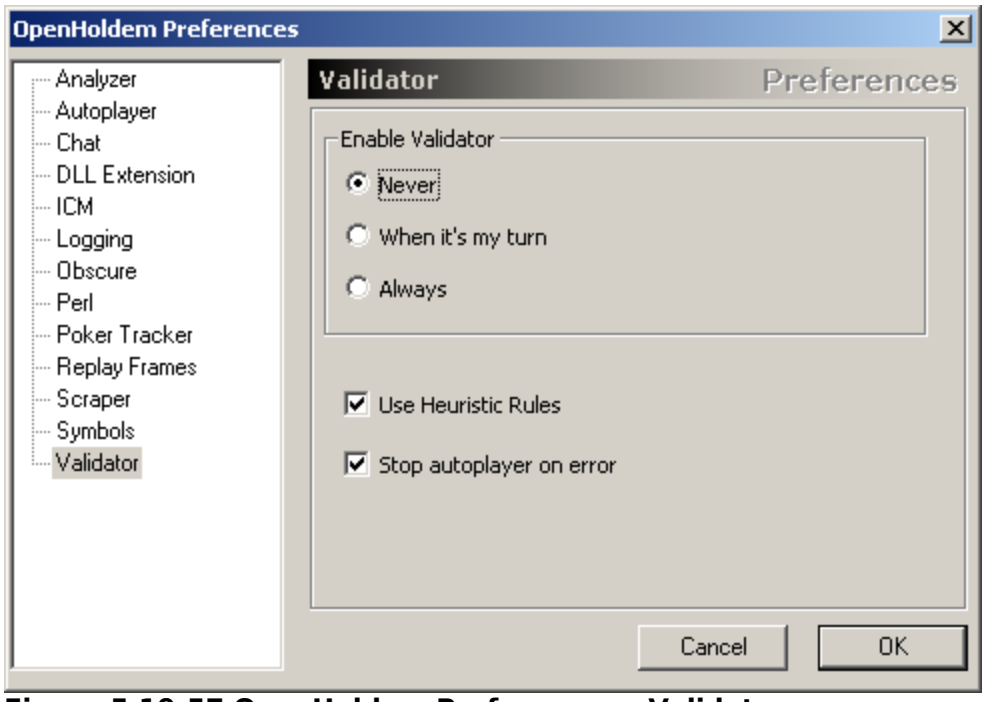


Figure 5.19-57 OpenHoldem Preferences – Validator

Validator preferences control various aspects of OpenHoldem's Validator engine. See the {Validator} section for more information. This is a new feature in OpenHoldem version 2.0. The purpose of the Validator is to help people to develop more reliable bots and Table Maps. It will execute several hundred consistency-checks at the symbol level to detect invalid game-states, mis-scrapes or incomplete information. The Validator will write possible problems to the log-file and also show a message box, as long as messages are not disabled. The rules used by the Validator are always being refined and developed. The discussion forum is the best place to keep on top of this: http://www.maxinmontreal.com/forums/viewtopic.php?f=189

- Enable Validator: This preference determines when the Validator is engaged to do its work, either Never (completely off), when it is my turn, or always.
- Use Heuristic Rules: Some rules are heuristic, i.e. they are "common sense", but not always true. Example: *A game does usually not last longer than 2 minutes, so the autoplayer has to act at least once per 120 seconds. If it does not do so, there seems to be a problem. Maybe the Table Map does not detect the players' cards or the buttons.* This setting can result in false positives and is recommended for testing only, but not for real-money-play.
- Stop Autoplayer on error: If the Validator discovers an inconsistency, and this preference is checked, the Autoplayer will be instructed to stop Autoplaying.

5.9 Locking Blinds

There is a button on OpenHoldem's main toolbar, that looks like a padlock and dollar symbol, which opens up the following window to allow you to "lock blinds".

Figure 5.20-58 OpenHoldem Lock Blinds window

OpenHoldem has many methods for determining the blinds at any given time, including:

- Scraping of Title Bar text (as specified in the Table Map)
- Scraping of Regions from the Table Map
- Intuiting the blinds from patterns in betting round 1

In some cases, you may want to bypass all of these blind detection methods and force OpenHoldem to use a certain blind setup. A good example of this is if you are playing at a Ring Table, where the blinds never change. This would not be so applicable to SNGs or MTTs, as the blinds regularly change in those games. This window will allow you to specify blinds to be used as long as the "padlock and dollar symbol" button on the main toolbar is depressed.

5.10 The Iterator

OpenHoldem includes a Monte Carlo Holdem simulator for use in calculating various statistics for use by your bot logic, the primary of which are the *prwin*/*prtie*/*prlos* trio. There are many resources on the Internet for learning about Monte Carlo poker simulations, a good place to start is here: http://pokerforprogrammers.blogspot.com, but there are plenty of other resources here too: http://www.google.com/#hl=en&q=monte+carlo+poker+simulation

There are a number of parameters that feed into the OpenHoldem iterator:

- f$P: The result of the f$P function directly determines the number of opponents that are simulated in the Iterator. The greater the number of opponents, the lower your chances of winning. The lower the number of opponents, the higher your chances of winning.
- NIT: The "Number of Iterations" parameter determines the number of cycles the Iterator processes each time the *prwin* trio of symbols are needed. Simplistically, the greater the number of iterations, the more precise will be the values in the *prwin* trio. A discussion on the appropriate number of iterations to choose for your needs can be found here: {Iterations and Standard Deviation}
- Table level hand weightings: This is discussed in detail here: {Weighted prwin}
- Opponent level hand weightings: This is discussed in detail here: {Enhanced prwin}

Note: The Iterator simulates a "nofoldem" game in all cases. This means that all players at the start of the hand (derived from f$P)

will see the hand through to the end, and that your hand is compared to all other players' hands to see if you won, lost or tied. In reality, this will rarely be the case, as inevitably, one or more players will fold before showdown.

Iterations and Standard Deviation

This material was originally produced by BuckyBall and can be found here: http://forum.winholdem.net/wbb/viewtopic.php?t=3742

Standard deviation of prwin/prtie/prlos by Number of Iterations (NIT)

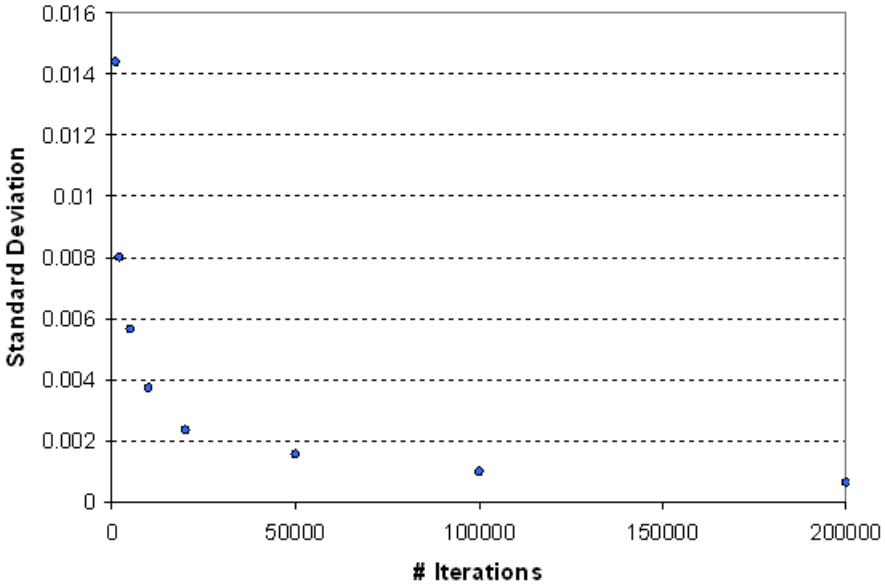


Figure 5.21-59 Standard Deviation by Iterations

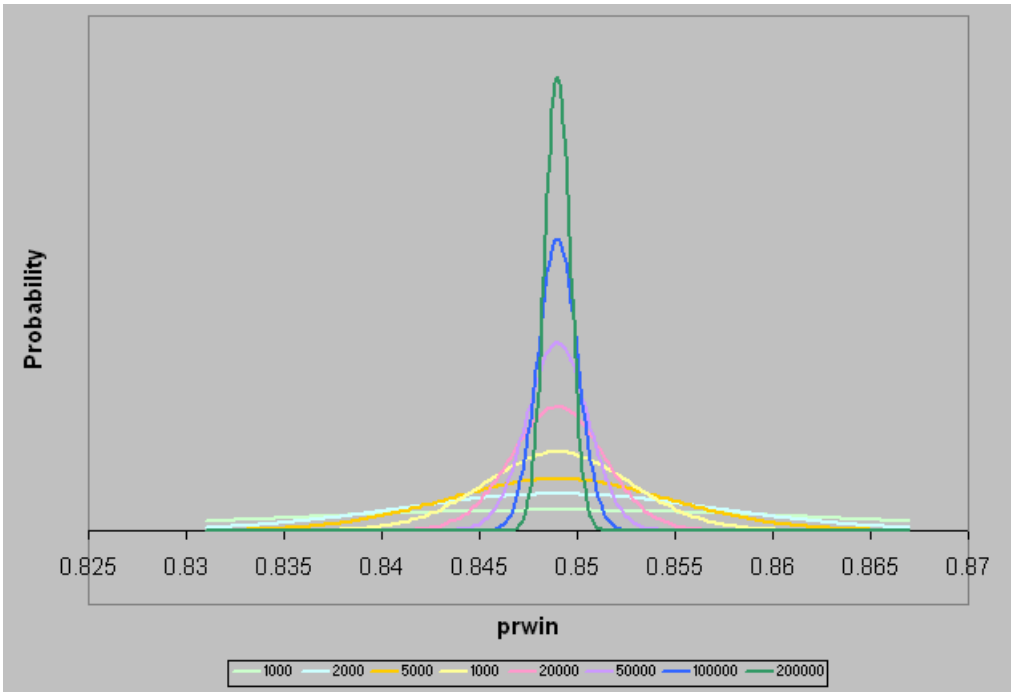


Figure 5.21-60 Probability by prwin

Here is how to use the data in the above graphs:

Decide what error you find acceptable 99% of the time. For example, if you are happy with getting prwin's of 0.846 - 0.852 99% of the time for AA headsup preflop, your error would be (0.849) +- 0.003.
That error is "3 standard deviations". To get 1 standard deviation, divide by 3 (to give std. dev. = 0.001).
Find that standard deviation in the top chart and read off the number of iterations required. In our case, it would be 100,000 required iterations.

The prwin values you obtain will be within 1 standard deviation 67% of the time, within 2 standard deviations 95% of the time, and within 3 standard deviations 99% of the time.

5.11 Weighted *prwin*

A weighted *prwin* is one which is calculated by making informed selections of the cards that opponents are likely to hold, and using these cards in the calculation of your win probability. Factors which can be used to make inferences about cards include willingness to pay to see flop, betting behavior when faced by common cards, and historical information about opponent behavior. Essentially it aims to put your opponents on cards as accurately as possible.

OpenHoldem's "Weighted *prwin*" is based on a fairly simple model using pre-flop behavior. For DLL developers, there is also the option to use {Enhanced prwin}.

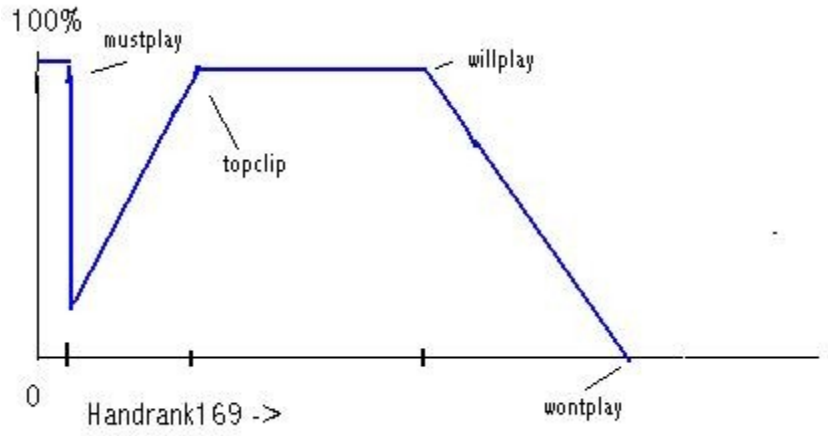


Figure 5.11-61 Weighted prwin model

To enable Weighted *prwin*, various user-defined functions in a formula file must be defined to return the following values, expressed as handrank169 values (see {Calculated Symbols}):

function	**explanation**
f$mustplay	Opponents holding cards better than this are assumed always to play them. This allows for the situation where opponents slow play the strongest hands in the hope of milking more from the pot with them.
f$topclip	Indicates that hands better than this are less likely to be played, since the betting level is too low.
f$willplay	Hands better than this will be played.
f$wontplay	Hands worse than this are highly unlikely to be played.

The success in using a model of this sort depends on the data available to allow the formula developer to devise appropriate expressions for the *f$functions* outlined above. Currently in OpenHoldem the most accessible is *floppct* - the percentage of opponents seeing the flop averaged over a user definable period. This is a reasonably accurate expression of the strength of cards which opponents are prepared to play. An example of its use could be:

f$willplay

```
handsplayed<10?72:(180*floppct)>(nopponents*2+1)?
180*floppct:nopponents*2+1
```

f$wontplay

```
f$willplay*1.4
```

Note: If *f$willplay* evaluates to zero, prwin weighting will be turned off. *f$wontplay* should not be less than *f$willplay*.

Although this is primitive it gives the single greatest advantage of using a weighted *prwin* rather than an unweighted one; it eliminates the cards from the calculation which opponents are unlikely to invest any money in. Since it depends on statistical

information a constant value is given until *floppct* has had time to establish itself. The '180' is because 90 is the midpoint of handrank169 at which the number of hands above and below are the same (the stronger hands tend to have fewer possible examples).

Ideally the calculation should be based on betting averages at different levels, since it is what opponents are prepared to pay that determines the probable strength of their hand, and should be recorded on an opponent (and even position) basis. The problem with more refined data accumulation is that it takes an increasing number of hands to gain significance, and is sensitive to changes in playing conditions, such as the number of players, which can alter opponent behavior. Nevertheless statistics of this sort have been used successfully in other bots based on a weighted *prwin*, and may be available from OpenHoldem in the future.

The OpenHoldem weighted *prwin* uses a handrank169 table (see {Starting Hand }) which is based on published data about hands players actually take to flop, and varies from conventional ones giving win probabilities, particularly in the way it favors suited hands! In the absence of statistical support it is not recommended that you implement *f$topclip* or *f$mustplay* unless you have a good picture of the standard of play at a table.

If *f$willplay* is absent, or returns a zero value, the *prwin* weighting is turned off. It will also be turned off if your *f$P* function specifies more than 13 opponents. If your opponents at flop include a big blind who has not been raised, no weighting will be applied to his hand, since no assumptions can be made about it.

You may need to change your *f$P* function if using weighted *prwin*. The “P factor” established in your *f$P* function is in fact a kludge to compensate for the fact that opponents who fold have weaker hands. *prwin* weighting allows for this, and therefore a *prwin* bot is best coded using the actual number of opponents playing. A suggestion for an *f$P* function, if you use *prwin* pre-flop, could be:

```
(nopponentsplaying==0) ? 0 : //zero if no opponents
 (br>1)? nopponentsplaying : //just use actual opponents post-flop
  (handsplayed<10)? nopponentsplaying : //fallback until floppct becomes reliable
   (nopponentscalling+nopponentsraising+nopponentschecking*floppct+0.49)<1 ?
1 : //this should never happen, but .....
    nopponentscalling+nopponentsraising+nopponentschecking*floppct+0.49 //make an
estimate for opponents who have still to act
```

This attempts to estimate the number of opponents you are likely to face when you act pre-flop, and is most significant when you are in early position. Whilst the principle is valid, that actual function should be thoroughly tested in practice.

5.12 Enhanced *prwin*

Overview

OpenHoldem’s Iterator can be switched from its normal table-level weighting calculation to one which considers hands likely to be held by individual opponent chairs. This can only be activated at the DLL level using the {prw1326} structure whose address is passed to the DLL at load time. The mechanism that OpenHoldem uses to select opponent cards for each pass of the Iterator is, for each active opponent, to choose a random number between zero and .limit and use this as an index into the chair [1326] arrays. .ranklo and .rankhi are checked to ensure that neither of the cards is already in use, if .weight is less than .level another random

number is used so that the hand will only be selectable .weight/.level of the time, and if the hand is available after these tests it is assigned for the opponent for that pass of the Iterator.

This approach allows the DLL coder to implement something very like the {Weighted prwin}, but at opponent chair level; to trim the probabilities of certain opponent hands as the hand develops; or to switch to an approach where only limited Hand Lists are considered for an opponent (possibly without any relative weighting). Please bear in mind that with Enhanced *prwin*, OpenHoldem is completely dependent on you to set up the prw1326 structure, and any errors you make will affect the reliability of the result; or cause OpenHoldem to hang or crash. It is not a feature for the novice coder to experiment with.

How to use it from a DLL

To use Enhanced *prwin* from a DLL, you need to declare the structures and pick up the pointer to them as described here: {prw1326}. To switch OpenHoldem to a calculation using data in the structure which you have generated you need to set:

```
prw1326->useme = 1326;
```

If this is all you do OpenHoldem will switch at flop and after to an enhanced calculation using the vanilla hand profiles it set up for each chair at initialization. This will probably give slightly worse results than a well-configured {Weighted prwin}, since the profiles are static!

Now assume that at the flop you become convinced that chair 2 has a pair of aces. No other ace is known, so he can have six possible hands. You want to set the structure so that OpenHoldem only calculates *prwin* using those hands for that opponent. First we need to note how OpenHoldem (or more specifically the PokerEval library) represents cards. Card rank 2 to Ace is represented by 0 to 12. You then add a number representing the suit. For hearts add 0, for diamonds add 13, for clubs add 26, and for spades add 39. Ac is thus 12+26=38. The six pairs he can have are AhAd, AhAc, AhAs, AdAc, AdAs and AcAs. You set the first six positions in the [1326] arrays:

```
prw1326->chair[2].rankhi[0]=12; //Ah
prw1326->chair[2].ranklo[0]=25; //Ad
...
prw1326->chair[2].rankhi[5]=25; //Ad
prw1326->chair[2].ranklo[5]=51; //As
```

Since you only want six card pairs to be considered you set:

```
prw1326->chair[2].limit=6;
```

Finally you set the weighting for each hand. As each hand is considered equally probable:

```
for(i=0;i<6;i++)prw1326->chair[2].weight[i]=prw1326->chair[2].level;
```

Once the hand was over you would not want to leave chair 2 as having a pair of aces.

```
prw1326->chair[2]=prw1326->vanilla_chair;
```

would reset it.

This was an example of using prw1326 to set an opponent on a hand list. We chose a very small list for ease of illustration, but it is worth noting that this can be dangerous. If another opponent had a hand list which included a pair of aces, and an ace appeared on common, then there would be a risk that OpenHoldem could not find enough cards to satisfy an iteration and would hang. Since the iteration process is so CPU intensive we did not include code to catch this situation.

Now let us suppose that chair 3 is a tight player. You think he will only place money on the upper half of the cards, and will only reliably pay for the top 25%. In this case you would want to take the vanilla profile and change the weightings on it.

```
for(i=0;i<332;i++) prw1326->chair[3].weight[i]=prw1326->chair[3].level;
for(i=332;i<663;i++) prw1326->chair[3].weight[i]=prw1326->chair[3].level*(663-i)/332;
prw1326->chair[3].limit=663;
```

Because we have set .limit for the chair we do not need to bother about the weightings for the bottom half of the hand list, because they will never be considered.

The procedure above would set a general profile for chair 3 based on pre-flop behavior. You might have a situation in which his betting behavior indicated that he might well have a jack in his hand. One response would be to traverse the hand list and reduce the weighting for any hand which did not contain a jack. A quicker technique is to increase the .level value (which automatically reduced the probability for all hands) and then increase the weighting only for hands containing a jack. It can be very useful to build up indexes in your DLL which show the hands which contain aces, kings etc, and also for those which contain suited cards. This can speed up this sort of operation considerably.

prw1326->preflop

If you use *prwin* at all pre-flop, there are real advantages to using the normal {Weighted prwin} before the flop. Enhanced *prwin* works by considering specific hand profiles for opponents, and in general, pre-flop you are not sure who your opponents later on will be. It also does not use the P factor to calculate against a number of opponents; it works strictly on active opponents. Using table tightness and position in an *f$P* formula to estimate likely number of opponents means that {Weighted prwin} is more useful pre-flop. Nevertheless, if you set…

```
prw1326->preflop=1326;
```

…then enhanced prwin will also be used pre-flop.

prw1326->bblimp

If this is set non-zero then Enhanced prwin will not attempt any weighting for a chair who has not voluntarily put any money into the pot pre-flop (i.e. an unraised BB). Please note that this is not set by default, but is a flag which should be set for the *prwin* calculation to be wholly realistic.

prw1326->chair[x].ignore

If this is set non-zero then Enhanced *prwin* will not attempt any weighting for chair x. The Iterator will make no assumptions about the chair hand list. The chair itself will still be included in the *prwin* calculation if it is still playing, but it is just treated as if it was an unraised BB, so that it could have any available hand.

Callback

There is a mechanism for synchronizing your setting of the prw1326 structure with the *prwin* cycle. You can specify a callback function which will be invoked immediately before the *prwin* iteration loop is started. You can use this to copy a local structure image over prw1326. Any code here should be short, simple and clean. Avoid get_symbol calls, logging or complex external functions. If you think the callback function is a good place to make a couple of hundred Poker Tracker queries then the OpenHoldem developers reserve the right to be offensively unsympathetic to you. If you have callback function in your code of type:

```
double mycallback(void)
{
//do something here
return 0;
}
```

then you can set

```
prw1326->prw_callback=mycallback;
prw1326->usecallback=1326;
```

The return value is not used currently by OpenHoldem.

Notes

The simple minded technique for getting a random number between 0 and X is rand() %X. Unfortunately the rand() function returns an integer between 0 and 32767, and if X is not a sub-multiple of 32768 then the result is biased (only slightly if X is not too large). You can code to eliminate this, but in an iterated piece of code this introduces an extra CPU load. Currently in prw1326 we do this elimination on the random numbers used to select the hand from the hand list, but not on the .weight / .level assessment since the default .level of 1024 does not introduce any bias. The effect of an unfortunate choice of .level (such as one > 16384) would be to cause hands with a low .weight to be over-represented in the calculations.

If you want to set an opponent on a limited hand list there are two ways you can do it with prw1326; you can take a 1326 list and set the weight of the unwanted hands to zero, or you can create a sequence of just those hands you are interested in and set .limit so that only they are considered. Both will produce the same result, but the latter is more CPU-efficient, particularly for a short list, since the Iterator does not have to waste time rejecting many hands with a zero .weight.

When you unload a DLL from the OpenHoldem menu, the .useme and .usecallback flags will be cleared but nothing else in the prw1326 structure will be changed. Another loaded dll will then find the structure as the previous one left it, so you cannot assume the initial OpenHoldem initialization unless you do careful housekeeping to restore it on DLL exit.

5.13 Starting Hand Ranking

OpenHoldem uses a starting hand rank table that is based on the 2652 possible starting two card hands. The original discussion around these hand rankings was developed by Ray Bornert and can be found here:
http://forum.winholdem.net/wbb/viewtopic.php?t=94
http://forum.winholdem.net/wbb/viewtopic.php?t=316

This table is as follows:

Number of opponents on the X axis, hand rank (1 is best) on the Y axis

---	--9-------	--8-------	--7-------	--6-------	--5-------	--4-------	--3-------	--2-------	--1-------	---
1	*AAo*- 12	*AAo*- 12	*AAo*- 12	*AAo*- 12	*AAo*- 12	*AAo*- 12	*AAo*- 12	*AAo*- 12	*AAo*- 12	1
2	*KKo*- 24	*KKo*- 24	*KKo*- 24	*KKo*- 24	*KKo*- 24	*KKo*- 24	*KKo*- 24	*KKo*- 24	*KKo*- 24	2
3	*QQo*- 36	*QQo*- 36	*QQo*- 36	*QQo*- 36	*QQo*- 36	*QQo*- 36	*QQo*- 36	*QQo*- 36	*QQo*- 36	3
4	*AKs*- 44	*AKs*- 44	*JJo*- 48	*JJo*- 48	*JJo*- 48	*JJo*- 48	*JJo*- 48	*JJo*- 48	*JJo*- 48	4
5	*JJo*- 56	*JJo*- 56	*AKs*- 56	*AKs*- 56	*AKs*- 56	*TTo*- 60	*TTo*- 60	*TTo*- 60	*TTo*- 60	5
6	*AQs*- 64	*AQs*- 64	*AQs*- 64	*AQs*- 64	*TTo*- 68	*AKs*- 68	*99o*- 72	*99o*- 72	*99o*- 72	6
7	*KQs*- 72	*KQs*- 72	*KQs*- 72	*TTo*- 76	*AQs*- 76	*AQs*- 76	*AKs*- 80	*AKs*- 80	*88o*- 84	7
8	*AJs*- 80	*AJs*- 80	*TTo*- 84	*KQs*- 84	*KQs*- 84	*99o*- 88	*AQs*- 88	*88o*- 92	*AKs*- 92	8
9	*TTo*- 92	*TTo*- 92	*AJs*- 92	*AJs*- 92	*AKo*- 108	*KQs*- 96	*AKo*- 112	*AQs*- 100	*77o*- 104	9
10	*KJs*- 100	*AKo*- 116	*AKo*- 116	*AKo*- 116	*AJs*- 116	*AKo*- 120	*AJs*- 120	*AKo*- 124	*AQs*- 112	10
11	*AKo*- 124	*KJs*- 124	*KJs*- 124	*KJs*- 124	*99o*- 128	*AJs*- 128	*KQs*- 128	*AJs*- 132	*AKo*- 136	11
12	*ATs*- 132	*ATs*- 132	*ATs*- 132	*ATs*- 132	*KJs*- 136	*KJs*- 136	*88o*- 140	*77o*- 144	*AJs*- 144	12
13	*QJs*- 140	*QJs*- 140	*QJs*- 140	*QJs*- 140	*ATs*- 144	*ATs*- 144	*ATs*- 148	*KQs*- 152	*AQo*- 168	13
14	*KTs*- 148	*KTs*- 148	*KTs*- 148	*99o*- 152	*QJs*- 152	*AQo*- 168	*AQo*- 172	*ATs*- 160	*ATs*- 176	14
15	*QTs*- 156	*QTs*- 156	*99o*- 160	*KTs*- 160	*AQo*- 176	*88o*- 180	*KJs*- 180	*AQo*- 184	*66o*- 188	15
16	*JTs*- 164	*99o*- 168	*QTs*- 168	*AQo*- 184	*KTs*- 184	*QJs*- 188	*QJs*- 188	*KJs*- 192	*AJo*- 212	16
17	*99o*- 176	*JTs*- 176	*AQo*- 192	*QTs*- 192	*KQo*- 208	*KTs*- 196	*KTs*- 196	*AJo*- 216	*KQs*- 220	17
18	*AQo - 200	*AQo*- 200	*JTs*- 200	*JTs*- 200	*QTs*- 216	*KQo*- 220	*KQo*- 220	*KTs*- 224	*ATo*- 244	18
19	*A9s - 208	*KQo - 224	*KQo*- 224	*KQo*- 224	*88o*- 228	*QTs*- 228	*AJo*- 244	*KQo*- 248	*A9s*- 252	19
20	*KQo - 232	*A9s - 232	*A9s*- 232	*88o*- 236	*JTs*- 236	*AJo*- 252	*77o*- 256	*A9s*- 256	*KJs*- 260	20
21	*88o - 244	*88o - 244	*88o*- 244	*A9s*- 244	*AJo*- 260	*JTs*- 260	*QTs*- 264	*QJs*- 264	*A8s*- 268	21
22	*K9s - 252	*K9s - 252	*AJo - 268	*AJo*- 268	*A9s*- 268	*A9s*- 268	*A9s*- 272	*ATo*- 288	*KTs*- 276	22
23	*T9s - 260	*T9s - 260	*K9s - 276	*K9s*- 276	*KJo*- 292	*KJo*- 292	*ATo*- 296	*66o*- 300	*KQo*- 300	23
24	A8s - 268	*AJo - 284	*A8s - 284	*KJo - 300	*K9s*- 300	*77o*- 304	*JTs*- 304	*A8s*- 308	*55o*- 312	24
25	J9s - 276	*A8s - 292	*T9s - 292	*A8s - 308	*A8s*- 308	*ATo*- 328	*KJo*- 328	*KJo*- 332	*A7s*- 320	25
26	Q9s - 284	J9s - 300	*KJo - 316	*Q9s - 316	*QJo*- 332	*K9s*- 336	*A8s*- 336	*QTs*- 340	*A9o*- 344	26
27	77o - 296	Q9s - 308	*Q9s - 324	*QJo - 340	*ATo - 356	*A8s*- 344	*K9s*- 344	*K9s*- 348	*KJo*- 368	27
28	AJo - 320	KJo - 332	J9s - 332	*T9s - 348	*77o - 368	*QJo*- 368	*QJo*- 368	*JTs*- 356	*QJs*- 376	28
29	A5s - 328	77o - 344	QJo - 356	*J9s - 356	*Q9s - 376	*Q9s*- 376	*66o*- 380	*A7s*- 364	*K9s*- 384	29
30	A7s - 336	A5s - 352	77o - 368	ATo - 380	*T9s - 384	*KTo*- 400	*KTo*- 404	*KTo*- 388	*KTo*- 408	30
31	A4s - 344	A7s - 360	A7s - 376	77o - 392	*J9s - 392	*A7s*- 408	*A7s*- 412	*QJo*- 412	*A8o*- 432	31
32	KJo - 368	QJo - 384	A5s - 384	A7s - 400	*KTo - 416	*J9s*- 416	*Q9s*- 420	*A9o*- 436	*A6s*- 440	32
33	A3s - 376	A4s - 392	ATo - 408	KTo - 424	*A7s - 424	*T9s - 424	*QTo*- 444	*55o*- 448	*QTs*- 448	33
34	66o - 388	A3s - 400	A4s - 416	A5s - 432	*A5s - 432	*QTo - 448	*J9s*- 452	*A5s*- 456	*A5s*- 456	34
35	A6s - 396	A6s - 408	KTo - 440	A4s - 440	QTo - 456	*A5s - 456	*A5s*- 460	*A6s*- 464	*A4s*- 464	35
36	QJo - 420	ATo - 432	A6s - 448	QTo - 464	JTo - 480	*66o - 468	*T9s*- 468	*Q9s*- 472	*A7o*- 488	36
37	K8s - 428	K8s - 440	K8s - 456	K8s - 472	K8s - 488	*JTo - 492	*A9o*- 492	*QTo*- 496	*QJo*- 512	37
38	A2s - 436	66o - 452	A3s - 464	A6s - 480	A4s - 496	*A6s - 500	*A6s*- 500	*A8o*- 520	*K8s*- 520	38
39	T8s - 444	A2s - 460	JTo - 488	JTo - 504	A6s - 504	*K8s - 508	*JTo*- 524	*K8s*- 528	*A3s*- 528	39
40	98s - 452	T8s - 468	QTo - 512	A3s - 512	66o - 516	*A4s - 516	*K8s*- 532	*A4s*- 536	*K9o*- 552	40
41	J8s - 460	KTo - 492	66o - 524	66o - 524	A3s - 524	A9o - 540	*A4s*- 540	*J9s*- 544	*44o*- 564	41
42	ATo - 484	98s - 500	T8s - 532	T8s - 532	Q8s - 532	A3s - 548	*55o - 552	*K9o*- 568	*Q9s*- 572	42
43	Q8s - 492	Q8s - 508	A2s - 540	Q8s - 540	T8s - 540	Q8s - 556	*A3s - 560	*A3s*- 576	*JTs*- 580	43
44	55o - 504	J8s - 516	Q8s - 548	A2s - 548	K7s - 548	K7s - 564	*K7s - 568	*JTo*- 600	*QTo*- 604	44
45	K7s - 512	JTo - 540	98s - 556	J8s - 556	J8s - 556	J8s - 572	*Q8s - 576	*K7s*- 608	*A6o*- 628	45
46	JTo - 536	QTo - 564	J8s - 564	K7s - 564	98s - 564	T8s - 580	*A8o - 600	*A7o*- 632	*K7s*- 636	46
47	KTo - 560	K7s - 572	K7s - 572	98s - 572	A2s - 572	A2s - 588	*K9o - 624	*T9s*- 640	*A5o*- 660	47
48	44o - 572	55o - 584	K6s - 580	K6s - 580	A9o - 596	98s - 596	*J8s - 632	*Q8s*- 648	*A2s*- 668	48
49	33o - 584	87s - 592	55o - 592	55o - 592	K6s - 604	K9o - 620	*A2s - 640	*A2s*- 656	*K6s*- 676	49
50	22o - 596	K6s - 600	87s - 600	A9o - 616	K9o - 628	55o - 632	*T8s - 648	*K6s*- 664	*A4o*- 700	50
51	QTo - 620	44o - 612	K5s - 608	87s - 624	55o - 640	K6s - 640	*K6s - 656	*44o*- 676	*K8o*- 724	51
52	87s - 628	97s - 620	97s - 616	K5s - 632	K5s - 648	A8o - 664	98s - 664	*Q9o*- 700	*Q8s*- 732	52
53	K6s - 636	33o - 632	44o - 628	97s - 640	87s - 656	Q9o - 688	Q9o - 688	*A5o*- 724	*J9s*- 740	53
54	97s - 644	22o - 644	K4s - 636	Q7s - 648	Q7s - 664	K5s - 696	A7o - 712	*A6o*- 748	*A3o*- 764	54
55	K5s - 652	K5s - 652	T7s - 644	K4s - 656	A8o - 688	T9o - 720	K5s - 720	*J8s*- 756	*K5s*- 772	55
56	76s - 660	T7s - 660	Q7s - 652	T7s - 664	Q9o - 712	Q7s - 728	J9o - 744	*K5s*- 764	*Q9o*- 796	56
57	T7s - 668	K4s - 668	J7s - 660	K9o - 688	T7s - 720	J9o - 752	T9o - 768	*K8o - 788	*JTo*- 820	57
58	K4s - 676	76s - 676	K3s - 668	J7s - 696	97s - 728	K4s - 760	Q7s - 776	*T8s - 796	*K7o*- 844	58
59	K3s - 684	Q7s - 684	A9o - 692	K3s - 704	K4s - 736	T7s - 768	A5o - 800	*A4o - 820	*A2o*- 868	59
60	Q7s - 692	K3s - 692	76s - 700	T9o - 728	T9o - 760	J7s - 776	44o - 812	*J9o - 844	*K4s*- 876	60
61	K2s - 700	J7s - 700	33o - 712	44o - 740	J7s - 768	87s - 784	K4s - 820	*Q7s - 852	*33o*- 888	61
62	J7s - 708	K2s - 708	K2s - 720	Q6s - 748	J9o - 792	A7o - 808	A6o - 844	*K4s - 860	*Q7s*- 896	62
63	86s - 716	86s - 716	Q6s - 728	Q9o - 772	K3s - 800	97s - 816	K8o - 868	*98s - 868	*K6o*- 920	63
64	65s - 724	65s - 724	22o - 740	J9o - 796	Q6s - 808	Q6s - 824	J7s - 876	A3o - 892	*T9s*- 928	64
65	54s - 732	Q6s - 732	K9o - 764	K2s - 804	44o - 820	K3s - 832	Q6s - 884	K7o - 916	*J8s*- 936	65
66	Q6s - 740	54s - 740	86s - 772	76s - 812	K2s - 828	44o - 844	T7s - 892	T9o - 940	*K3s*- 944	66
67	75s - 748	A9o - 764	T9o - 796	A8o - 836	76s - 836	A5o - 868	97s - 900	Q6s - 948	*Q8o*- 968	67
68	Q5s - 756	Q5s - 772	65s - 804	86s - 844	Q5s - 844	K8o - 892	87s - 908	K3s - 956	*Q6s*- 976	68
69	96s - 764	T9o - 796	Q5s - 812	Q5s - 852	A7o - 868	K2s - 900	K3s - 916	J7s - 964	*J9o*-1000	69
70	Q4s - 772	96s - 804	J9o - 836	33o - 864	86s - 876	Q5s - 908	A4o - 940	Q8o - 988	*K5o*-1024	70
71	Q3s - 780	75s - 812	Q9o - 860	65s - 872	A5o - 900	A6o - 932	Q5s - 948	A2o -1012	*K2s*-1032	71
72	64s - 788	Q4s - 820	96s - 868	Q4s - 880	Q4s - 908	76s - 940	K2s - 956	33o -1024	*Q5s*-1040	72

73	T9o - 812	K9o - 844	54s - 876	96s - 888	96s - 916	A4o - 964	Q8o - 980	K6o -1048	*T8s*-1048	73
74	Q2s - 820	J9o - 868	Q4s - 884	22o - 900	T6s - 924	86s - 972	K7o -1004	Q5s -1056	*J7s*-1056	74
75	A9o - 844	Q3s - 876	A8o - 908	T6s - 908	33o - 936	Q4s - 980	A3o -1028	K2s -1064	*K4o*-1080	75
76	T6s - 852	64s - 884	75s - 916	Q3s - 916	K8o - 960	Q8o -1004	J8o -1052	T7s -1072	*Q7o*-1104	76
77	53s - 860	T6s - 892	T6s - 924	A7o - 940	J6s - 968	J6s -1012	Q4s -1060	J8o -1096	*T9o*-1128	77
78	J6s - 868	Q9o - 916	Q3s - 932	J6s - 948	65s - 976	T8o -1036	T8o -1084	97s -1104	*Q4s*-1136	78
79	85s - 876	Q2s - 924	J6s - 940	75s - 956	Q3s - 984	T6s -1044	J6s -1092	87s -1112	*J8o*-1160	79
80	J9o - 900	J6s - 932	Q2s - 948	54s - 964	A6o -1008	96s -1052	76s -1100	Q4s -1120	*K3o*-1184	80
81	K9o - 924	85s - 940	64s - 956	Q2s - 972	A4o -1032	J8o -1076	A2o -1124	K5o -1144	*22o*-1196	81
82	43s - 932	53s - 948	85s - 964	A5o - 996	T8o -1056	A3o -1100	98o -1148	T8o -1168	*Q6o*-1220	82
83	J5s - 940	A8o - 972	J5s - 972	J5s -1004	J5s -1064	K7o -1124	33o -1160	J6s -1176	*Q3s*-1228	83
84	Q9o - 964	J5s - 980	53s - 980	K8o -1028	Q8o -1088	Q3s -1132	K6o -1184	Q3s -1184	*98s*-1236	84
85	74s - 972	J4s - 988	A7o -1004	85s -1036	Q2s -1096	98o -1156	T6s -1192	Q7o -1208	*T7s*-1244	85
86	J4s - 980	74s - 996	J4s -1012	64s -1044	75s -1104	33o -1168	86s -1200	98o -1232	*J6s*-1252	86
87	J3s - 988	43s -1004	A5o -1036	A4o -1068	J8o -1128	J5s -1176	96s -1208	K4o -1256	*K2o*-1276	87
88	J2s - 996	J3s -1012	J3s -1044	T8o -1092	98o -1152	65s -1184	Q3s -1216	T6s -1264	*Q2s*-1284	88
89	95s -1004	95s -1020	95s -1052	J4s -1100	54s -1160	Q2s -1192	J5s -1224	J5s -1272	*Q5o*-1308	89
90	63s -1012	J2s -1028	74s -1060	A6o -1124	22o -1172	A2o -1216	Q2s -1232	Q2s -1280	*J5s*-1316	90
91	A8o -1036	63s -1036	T8o -1084	98o -1148	A3o -1196	75s -1224	K5o -1256	96s -1288	T8o -1340	91
92	T5s -1044	T5s -1044	J2s -1092	Q8o -1172	K7o -1220	J4s -1232	J4s -1264	Q6o -1312	J7o -1364	92
93	52s -1052	A5o -1068	43s -1100	J8o -1196	J4s -1228	K6o -1256	65s -1272	76s -1320	Q4o -1388	93
94	42s -1060	A7o -1092	A4o -1124	J3s -1204	85s -1236	85s -1264	Q7o -1296	86s -1328	97s -1396	94
95	T4s -1068	T8o -1116	K8o -1148	53s -1212	J3s -1244	54s -1272	75s -1304	J7o -1352	J4s -1404	95
96	T3s -1076	T4s -1124	T5s -1156	A3o -1236	64s -1252	J3s -1280	J7o -1328	22o -1364	T6s -1412	96
97	84s -1084	98o -1148	98o -1180	95s -1244	95s -1260	22o -1292	J3s -1336	K3o -1388	J3s -1420	97
98	98o -1108	T3s -1156	J8o -1204	J2s -1252	A2o -1284	T5s -1300	85s -1344	J4s -1396	Q3o -1444	98
99	T2s -1116	84s -1164	T4s -1212	T5s -1260	T5s -1292	95s -1308	K4o -1368	T7o -1420	98o -1468	99
100	A5o -1140	52s -1172	63s -1220	74s -1268	J2s -1300	J2s -1316	T7o -1392	Q5o -1444	T7o -1492	100
101	T8o -1164	A4o -1196	A6o -1244	K7o -1292	K6o -1324	64s -1324	T5s -1400	J3s -1452	J6o -1516	101
102	A7o -1188	T2s -1204	A3o -1268	T4s -1300	T4s -1332	K5o -1348	95s -1408	K2o -1476	87s -1524	102
103	73s -1196	42s -1212	Q8o -1292	43s -1308	53s -1340	Q7o -1372	97o -1432	T5s -1484	J2s -1532	103
104	32s -1204	K8o -1236	T3s -1300	A2o -1332	74s -1348	87o -1396	87o -1456	97o -1508	96s -1540	104
105	A4o -1228	A3o -1260	84s -1308	T3s -1340	87o -1372	T4s -1404	22o -1468	87o -1532	Q2o -1564	105
106	94s -1236	J8o -1284	T2s -1316	84s -1348	T3s -1380	T7o -1428	Q6o -1492	65s -1540	J5o -1588	106
107	93s -1244	A6o -1308	52s -1324	63s -1356	97o -1404	J7o -1452	J2s -1500	J2s -1548	T5s -1596	107
108	62s -1252	73s -1316	A2o -1348	T2s -1364	43s -1412	97o -1476	54s -1508	95s -1556	T4s -1604	108
109	A3o -1276	Q8o -1340	K7o -1372	87o -1388	T7o -1436	74s -1484	T4s -1516	75s -1564	97o -1628	109
110	K8o -1300	94s -1348	42s -1380	K6o -1412	T2s -1444	K4o -1508	K3o -1540	Q4o -1588	J4o -1652	110
111	J8o -1324	32s -1356	87o -1404	52s -1420	84s -1452	T3s -1516	64s -1548	T4s -1596	T6o -1676	111
112	92s -1332	93s -1364	94s -1412	94s -1428	K5o -1476	53s -1524	Q5o -1572	85s -1604	86s -1684	112
113	A6o -1356	A2o -1388	73s -1420	97o -1452	Q7o -1500	Q6o -1548	T3s -1580	J6o -1628	95s -1692	113
114	87o -1380	87o -1412	93s -1428	93s -1460	63s -1508	84s -1556	K2o -1604	T3s -1636	T3s -1700	114
115	Q8o -1404	92s -1420	92s -1436	73s -1468	J7o -1532	T2s -1564	74s -1612	Q3o -1660	J3o -1724	115
116	83s -1412	62s -1428	32s -1444	42s -1476	94s -1540	K3o -1588	84s -1620	54s -1668	76s -1732	116
117	A2o -1436	K7o -1452	97o -1468	T7o -1500	K4o -1564	94s -1596	T2s -1628	T6o -1692	87o -1756	117
118	82s -1444	83s -1460	K6o -1492	K5o -1524	93s -1572	43s -1604	76o -1652	J5o -1716	T2s -1764	118
119	97o -1468	97o -1484	62s -1500	92s -1532	52s -1580	63s -1612	Q4o -1676	T2s -1724	96o -1788	119
120	72s -1476	82s -1492	83s -1508	Q7o -1556	73s -1588	76o -1636	94s -1684	96o -1748	J2o -1812	120
121	K7o -1500	76o -1516	T7o -1532	J7o -1580	Q6o -1612	Q5o -1660	53s -1692	64s -1756	85s -1820	121
122	76o -1524	72s -1524	82s -1540	32s -1588	76o -1636	93s -1668	J6o -1716	86o -1780	T5o -1844	122
123	T7o -1548	K6o -1548	76o -1564	76o -1612	92s -1644	K2o -1692	T6o -1740	76o -1804	94s -1852	123
124	65o -1572	T7o -1572	K5o -1588	83s -1620	K3o -1668	86o -1716	86o -1764	Q2o -1828	T4o -1876	124
125	K6o -1596	65o -1596	J7o -1612	62s -1628	42s -1676	92s -1724	96o -1788	94s -1836	75s -1884	125
126	86o -1620	K5o -1620	72s -1620	K4o -1652	86o -1700	73s -1732	93s -1796	74s -1844	93s -1892	126
127	54o -1644	86o -1644	Q7o -1644	82s -1660	83s -1708	96o -1756	Q3o -1820	84s -1852	86o -1916	127
128	K5o -1668	J7o -1668	K4o -1668	86o -1684	K2o -1732	52s -1764	63s -1828	J4o -1876	65s -1924	128
129	J7o -1692	54o -1692	86o -1692	Q6o -1708	Q5o -1756	Q4o -1788	43s -1836	93s -1884	95o -1948	129
130	Q7o -1716	Q7o -1716	65o -1716	K3o -1732	82s -1764	J6o -1812	J5o -1860	53s -1892	T3o -1972	130
131	75o -1740	K4o -1740	K3o -1740	72s -1740	62s -1772	T6o -1836	92s -1868	J3o -1916	84s -1980	131
132	K4o -1764	K3o -1764	Q6o -1764	K2o -1764	32s -1780	83s -1844	73s -1876	92s -1924	92s -1988	132
133	K3o -1788	75o -1788	K2o -1788	65o -1788	96o -1804	42s -1852	Q2o -1900	T5o -1948	76o -2012	133
134	96o -1812	K2o -1812	54o -1812	96o -1812	65o -1828	65o -1876	65o -1924	63s -1956	T2o -2036	134
135	64o -1836	Q6o -1836	96o -1836	Q5o -1836	T6o -1852	82s -1884	83s -1932	65o -1980	74s -2044	135
136	K2o -1860	96o -1860	75o -1860	54o -1860	Q4o -1876	Q3o -1908	J4o -1956	73s -1988	85o -2068	136
137	53o -1884	64o -1884	Q5o -1884	T6o -1884	J6o -1900	J5o -1932	52s -1964	43s -1996	54s -2076	137
138	Q6o -1908	Q5o -1908	T6o -1908	75o -1908	72s -1908	62s -1940	75o -1988	95o -2020	64s -2084	138
139	85o -1932	T6o -1932	Q4o -1932	Q4o -1932	75o -1932	75o -1964	82s -1996	75o -2044	83s -2092	139
140	T6o -1956	53o -1956	64o -1956	J6o -1956	Q3o -1956	32s -1972	85o -2020	83s -2052	94o -2116	140
141	Q5o -1980	Q4o -1980	J6o -1980	Q3o -1980	54o -1980	Q2o -1996	42s -2028	J2o -2076	75o -2140	141
142	43o -2004	85o -2004	Q3o -2004	Q2o -2004	J5o -2004	72s -2004	J3o -2052	85o -2100	82s -2148	142
143	Q4o -2028	J6o -2028	85o -2028	85o -2028	Q2o -2028	J4o -2028	T5o -2076	T4o -2124	93o -2172	143
144	Q3o -2052	Q3o -2052	53o -2052	64o -2052	85o -2052	54o -2052	62s -2084	82s -2132	73s -2180	144
145	74o -2076	Q2o -2076	Q2o -2076	J5o -2076	J4o -2076	85o -2076	95o -2108	52s -2140	65o -2204	145
146	Q2o -2100	43o -2100	J5o -2100	53o -2100	64o -2100	95o -2100	54o -2132	T3o -2164	53s -2212	146
147	J6o -2124	74o -2124	74o -2124	J4o -2124	95o -2124	T5o -2124	72s -2140	54o -2188	63s -2220	147
148	63o -2148	J5o -2148	43o -2148	95o -2148	J3o -2148	J3o -2148	T4o -2164	62s -2196	84o -2244	148
149	J5o -2172	J4o -2172	J4o -2172	J3o -2172	T5o -2172	64o -2172	J2o -2188	42s -2204	92o -2268	149
150	95o -2196	95o -2196	95o -2196	74o -2196	53o -2196	J2o -2196	32s -2196	64o -2228	43s -2276	150
151	52o -2220	63o -2220	J3o -2220	T5o -2220	J2o -2220	T4o -2220	64o -2220	72s -2236	74o -2300	151
152	J4o -2244	J3o -2244	J2o -2244	43o -2244	74o -2244	74o -2244	T3o -2244	T2o -2260	72s -2308	152
153	42o -2268	J2o -2268	63o -2268	J2o -2268	T4o -2268	53o -2268	74o -2268	74o -2284	54o -2332	153
154	J3o -2292	52o -2292	T5o -2292	T4o -2292	43o -2292	T3o -2292	84o -2292	94o -2308	64o -2356	154
155	J2o -2316	T5o -2316	52o -2316	63o -2316	T3o -2316	84o -2316	T2o -2316	84o -2332	52s -2364	155
156	84o -2340	84o -2340	T4o -2340	T3o -2340	84o -2340	T2o -2340	53o -2340	32s -2340	62s -2372	156
157	T5o -2364	42o -2364	84o -2364	84o -2364	63o -2364	94o -2364	94o -2364	93o -2364	83o -2396	157
158	32o -2388	T4o -2388	T3o -2388	52o -2388	T2o -2388	43o -2388	93o -2388	53o -2388	82o -2420	158
159	T4o -2412	T3o -2412	42o -2412	T2o -2412	94o -2412	63o -2412	63o -2412	92o -2412	42s -2428	159
160	T3o -2436	32o -2436	T2o -2436	94o -2436	52o -2436	93o -2436	43o -2436	63o -2436	73o -2452	160
161	73o -2460	T2o -2460	73o -2460	42o -2460	93o -2460	73o -2460	92o -2460	43o -2460	53o -2476	161
162	T2o -2484	73o -2484	94o -2484	73o -2484	73o -2484	92o -2484	73o -2484	73o -2484	63o -2500	162
163	62o -2508	94o -2508	32o -2508	93o -2508	42o -2508	52o -2508	83o -2508	83o -2508	32s -2508	163
164	94o -2532	62o -2532	93o -2532	92o -2532	92o -2532	83o -2532	52o -2532	82o -2532	43o -2532	164
165	93o -2556	93o -2556	62o -2556	32o -2556	83o -2556	42o -2556	82o -2556	52o -2556	72o -2556	165
166	92o -2580	92o -2580	92o -2580	83o -2580	62o -2580	82o -2580	42o -2580	62o -2580	52o -2580	166
167	83o -2604	83o -2604	83o -2604	62o -2604	32o -2604	62o -2604	62o -2604	42o -2604	62o -2604	167

```
168    82o -2628   82o -2628   82o -2628   82o -2628   82o -2628   32o -2628   72o -2628   72o -2628   42o -2628   168
169    72o -2652   72o -2652   72o -2652   72o -2652   72o -2652   72o -2652   32o -2652   32o -2652   32o -2652   169
---   --9-------  --8-------  --7-------  --6-------  --5-------  --4-------  --3-------  --2-------  --1-------   ---
```

Yes, that is a small font. It is also embedded here as a text file for reference:

OpenHoldem Hand Rankings.7z

The rank order of hands changes depending on the number of opponents (*f$p* function).

From this hand rank table, OpenHoldem will derive the symbols *handrank*, *handrank169*, *handrank1326*, *handrank2652*, *handrank1000*, and *handrankp*.

There are two initial methods you can use to leverage these hand rankings in your formula. First, if you want a static rule that will not be affected by folding opponents then you would use this code snippet:

```
|| [ br==1 && handrank2652 <= 2652*dealposition/nplayersdealt**2 ]
```

Second, if you want a dynamic rule that will be affected by folding opponents then you would use this rule:

```
|| [ br==1 && handrank2652 <= 2652*betposition/nplayersplaying**2 ]
```

The table above also flags “good” hands with leading and trailing asterisks. The function used to determine the left asterisk is:

```
handrank2652 <= 2652/nplayersdealt
```

And the function used for the right asterisk is:

```
handrank2652 <= 2652 * (1.0 - e**(ln(0.500)/nopponents))
```

The right asterisk is the point where the probability that all opponent hands are worse becomes 50% or more.

For further reference, the following embedded file contains starting hand probabilities for 100,000 and 10 million hand samples.

Starting hand probabilities.7z

5.14 The Scrape Cycle (or “Heartbeat”)

Once OpenHoldem has been connected to a table, it will enter into a continuous loop that performs a variety of actions. The cycle of these actions is described here.

action	**explanation**
scrape window	Ask the screen scraper engine to scrape and interpret the current pixels on the poker window based on the parameters in

	the loaded Table Map {Creating a Table Map}
scraper override	If a Scraper Override DLL is loaded, then pass the current screen scrape to that DLL for possible overrides {Scraper Override DLL}
calculate symbols	Calculate all OpenHoldem symbols {Calculated Symbols}
create replay frame	If a replay frame {Replay Frames} is requested in preferences {Preferences}, create it now
validate game state	If the Validator {Validator} is enabled in preferences {Preferences}, then validate the current game state
send state to user dll	If a User DLL {User DLL} is loaded, then send the current game state to it
autoplay	If the Autoplayer {The Autoplayer} is engaged, then ask the Autoplayer to do its bit
wait	Sleep for the Scrape Delay period of time as specified in preferences {Preferences}

5.15 The Autoplayer

OpenHoldem's Autoplayer engine uses the following logic to decide what action to take on the poker window:

- If the *f$alli* function evaluates to non-zero, and an all-in control is present (slider/SWAG box) then move all-in
- Else if the *f$swag* function evaluates to non-zero, and the SWAG box is present, then enter the *f$swag* result into the SWAG box and confirm the bet
- Else if the *f$rais* function evaluates to non-zero, and the Raise button is present, then click the Raise button
- Else if the *f$call* function evaluates to non-zero, and the Call button is present, then click the Call button
- Else if the Check button is present, then click the Check button
- Else click the Fold button

To determine when and which action to take the Autoplayer performs the following actions:

action	**explanation**
handle *f$play*	Evaluate the *f$play* function and take action depending on its return value
handle spam	Evaluate any i86 regions, and dismiss any popups/spam
handle *f$prefold*	If *f$prefold* evaluates to non-zero, then engage the prefold button as defined in the Table Map
handle *f$chat*	If *f$chat* evaluates to non-zero and safety constraints have been met, then enter a chat message into the chat box as defined in the Table Map
isfinalanswer	Determine the value of the *isfinalanswer* symbol – if the Iterator has finished, at least one action button is visible, we are "playing", the number of sequential stable frames have been seen and the time value in f$delay has passed, then *isfinalanswer* is true
evaluate primary	*f$alli*, *f$swag*, *f$rais* and *f$call* are evaluated

functions	
take action based on the above logic	Use the if-then-else logic as described above to take the appropriate Autoplayer action

5.16 Replay Frames

OpenHoldem has the ability to create “Replay Frames”, either automatically as set in {Preferences}, or when the “camera” icon on the main toolbar {Toolbars} is clicked. A “Replay Frame” (or “frame” for short), is a combination of a Windows bitmap-formatted image file (.bmp) and an associated HTML file. These two files, in combination, provide an easy way to reproduce and diagnose a particular game situation. This is extremely helpful in the building and testing of your Table Maps and bot logic in an offline fashion. Simply connect OpenHoldem to a live poker table, and collect interesting frames for later use. Then at a later time, use {OHReplay} to display these frames. Further connect OpenScrape or OpenHoldem to these OHReplay-displayed frames and work at your leisure.

The HTML provides crucial context for the frame, the most important of which is the title text. This text is used exhaustively in most Table Maps to understand the game state.

Often, when you post a problem or discussion on the forums, you will be asked to provide a frame and Table Map to help the developers diagnose your problem. Feel free to open the bitmap image in your favorite image editor (MS Paint?) and black out the sensitive portions if needed. Be aware that blacking out some parts of the image may prevent the developers or community from helping you, so be careful with that.

> Note: Both parts of the frame (.bmp and .html) and the associated Table Map are required to be provided when this is asked from you to help solve your problem!

Frames are created in a “replay” directory directly under the directory from which you launched OpenHoldem. The .bmp files are large, so ensure that you have enough space to hold the maximum number of frames you specified in {Preferences}. Once the maximum number is reached, OpenHoldem will start reusing frame numbers from the beginning.

5.17 PokerPro

PokerPro is a poker server developed by Ray Bornert. OpenHoldem has the ability to connect to this server, the primary benefit of which is the development of bot logic without risking real money. The configuration of the PokerPro server and a description of its network protocols is beyond the scope of this documentation, but extensive support on the topic can be found here:
http://forum.winholdem.net/wbb/viewforum.php?f=22

Due to undocumented changes in the communication-protocol the new PokerPro-server is no longer useable for tournament-players. To simulate tournaments the older 2007-version is required.

1. Source: http://forum.winholdem.net/wbb/viewtopic.php?t=10941

When the main menu item PokerPro/Go is selected, the following window will appear:

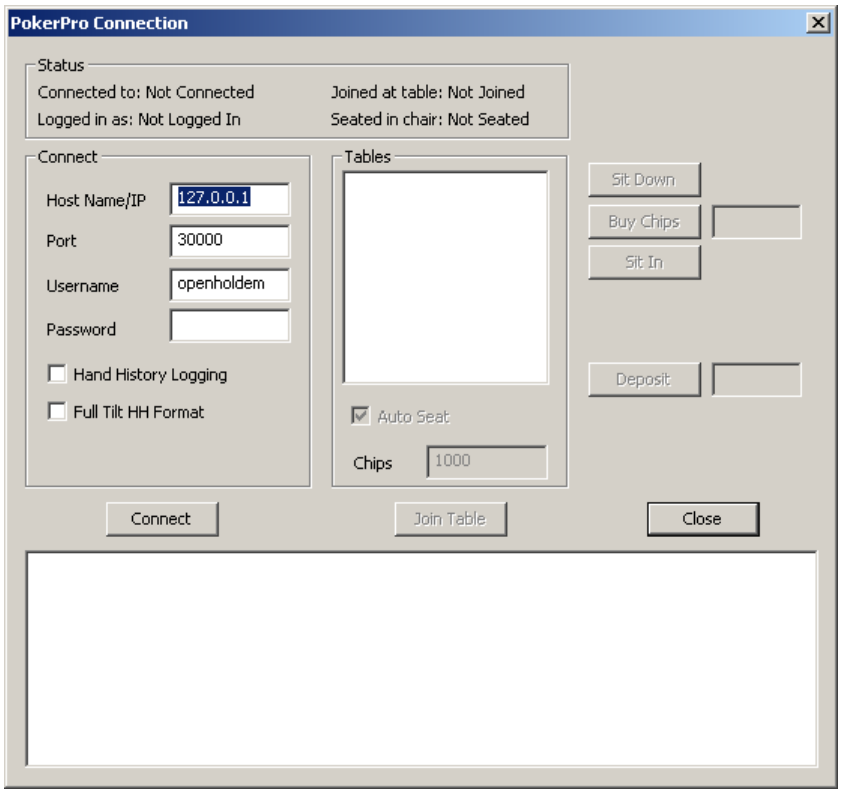


Figure 5.28-62 OpenHoldem PokerPro control

The top of this window provides a group box that informs you of the connection status to the PokerPro server.

On the left third of the window, in the “Connect” group box is where the connection details of the PokerPro server are specified. Additionally, you can choose to log hand histories either in the default PokerPro format or in the Full Tilt format here.

Once the server details are entered in the “Connect” group box, the “Connect” button can be used to connect to the server.

Upon connecting to the PokerPro server with the “Connect” button, the “Tables” group box in the middle third of the window will be populated with tables that are available on the connected PokerPro server. If you would like PokerPro to automatically seat you at a table when you join it, check the “Auto Seat” box and enter the amount of starting chips you would like to have.

After a table is selected from the “Tables” group box, you can join the selected table by clicking the “Join Table” button.

Once a table has been joined, the “Sit Down”, “Buy Chips”, “Sit In” and “Deposit” controls on the right third of the window will be enabled.

The bottom portion of this window contains a large scroll box that displays messages from the PokerPro server.

This window can be closed, and OpenHoldem will maintain its connection to the PokerPro server. To close the connection, reopen this window and choose the “Leave Table” or “Disconnect” buttons as needed.

5.18 PokerTracker

Before you try to configure OpenHoldem to work with a Poker Tracker database, you must have a properly working Poker Tracker configuration set up first. This is not the most trivial of activities, and does have a learning curve. The best information on how to set up Poker Tracker can be found here: http://www.pokertracker.com

Assuming you have a working Poker Tracker installation, let's move on to getting OpenHoldem to work with your Poker Tracker database. OpenHoldem only works with PostgreSQL Poker Tracker databases, so if you do not have Poker Tracker configured in that way (and you know if you do), go back here and get that set up first. http://www.pokertracker.com

OK, so now you have Poker Tracker installed using PostgreSQL as the underlying database. Do you have Poker Tracker setup to automatically import hand histories into your database? If not, OpenHoldem will not be able to provide you any useful information, as there will be no information in your database to pull from. Learn how to automatically import hand histories here: http://www.pokertracker.com

After all of that is configured, you can use the settings in {Preferences} to instruct OpenHoldem how to connect to your Poker Tracker database.

When connected, OpenHoldem will query the database using the sitename and playername as key parameters for the query.

Sitename

Everything in Poker Tracker is first organized by what it calls a "site id". When querying the database, the first thing OpenHoldem will try to do is associate your *sitename* or *network*, as specified in the Table Map, to a Poker Tracker "site id". This is done by doing a case-insensitive substring lookup against the following lists. This means that if your *sitename* string is "stars10sng" it will still be able to associate that to the proper "site id", as "stars" is within that *sitename*.

Version 2 list: "Stars", "Party", "Ultimate", "Absolute", "Microgaming", "Ongame", "Cryptologic", "Pacific", "FullTilt", "B2B", "Tribeca", "Worldpex", "iPoker", "Tain", "Bodog", "Everest", "Boss", "Betfair"

Version 3 list: "Stars", "Party", "FullTilt", "iPoker", "Everest", "Ongame"

If one of the above character strings cannot be found in either the *sitename* or *network* strings, as specified in your Table Map file, using a case-insensitive substring search, then OpenHoldem will not be able to lookup statistics for the players playing at your table.

As a note, the *sitename* string is searched first, and if a match cannot be found there, then the *network* string is searched.

Playername

The second critical piece of information that OpenHoldem needs to query the database is the name of the player. OpenHoldem gets this from the *pXname*/*uXname* regions on the Table Map. Note that this scraped name does not need to exactly match the name as stored in the database. This is useful, as there is

currently no method for scraping spaces in players' names. OpenHoldem will in all cases, return the statistics for an exact name match if it exists, and if it does not exist, will find the closest match using a Levenshtein distance algorithm (http://en.wikipedia.org/wiki/Levenshtein_distance). Note that Levenshtein distances that exceed one-half the length of the name as stored in Poker Tracker are discarded as invalid.

Poker Tracker symbols

Assuming all the prerequisites as described above are met, the following symbols will be available to your bot for use in its logic processing. The appropriate use of these symbols in opponent modeling is beyond the scope of this document, but plenty of references can be found on the Internet with a Google search.

Ring symbols

Symbol	**Explanation**
pt_iconx (x=0-9)	Poker Tracker auto-rate icon code for chair x
pt_iconlastrx (x=1-4)	Poker Tracker auto-rate icon code for the last raiser in round x
pt_pfrx (x=0-9)	Poker Tracker pre-flop raise percentage for chair x
pt_aggtotx (x=0-9)	Poker Tracker total aggression for chair x
pt_aggtotnopfx (x=0-9)	Poker Tracker total aggression excluding preflop for chair x
pt_aggpx (x=0-9)	Poker Tracker preflop aggression for chair x
pt_aggfx (x=0-9)	Poker Tracker flop aggression for chair x
pt_aggtx (x=0-9)	Poker Tracker turn aggression for chair x
pt_aggrx (x=0-9)	Poker Tracker river aggression for chair x
pt_floppctx (x=0-9)	Poker Tracker saw flop pct for chair x
pt_turnpctx (x=0-9)	Poker Tracker saw turn pct for chair x
pt_riverpctx (x=0-9)	Poker Tracker saw river pct for chair x
pt_vpipx (x=0-9)	Poker Tracker VP$IP for chair x
pt_handsx (x=0-9)	Poker Tracker number of hands that are in the database for chair x
pt_pf_rfix (x=0-9)	Poker Tracker "pre-flop raise first in" pct for chair x
pt_pf_crx (x=0-9)	Poker Tracker "pre-flop called raise" pct for chair x
pt_pfatsx (x=0-9)	Poker Tracker attempt to steal blinds for chair x
pt_wsdpx (x=0-9)	Poker Tracker went to showdown percentage for chair x
pt_wssdx (x=0-9)	Poker Tracker won $ at showdown for chair x
pt_fsbtsx (x=0-9)	Poker Tracker folded small blind to steal for chair x
pt_fbbtsx (x=0-9)	Poker Tracker folded big blind to steal for chair x

Ring symbols for the "raischair"

Symbol	**Explanation**
pt_ricon	Poker Tracker auto-rate icon code for raise chair
pt_rpfr	Poker Tracker pre-flop raise percentage for raise chair
pt_raggtot	Poker Tracker total aggression for raise chair

pt_raggtotnopf	Poker Tracker total aggression excluding preflop for raise chair
pt_raggp	Poker Tracker preflop aggression for raise chair
pt_raggf	Poker Tracker flop aggression for raise chair
pt_raggt	Poker Tracker turn aggression for raise chair
pt_raggrv	Poker Tracker river aggression for raise chair
pt_rfloppct	Poker Tracker saw flop pct for raise chair
pt_rturnpct	Poker Tracker saw turn pct for raise chair
pt_rriverpct	Poker Tracker saw river pct for raise chair
pt_rvpip	Poker Tracker VP$IP for raise chair
pt_rhands	Poker Tracker number of hands that are in the database for raise chair
pt_rpf_rfi	Poker Tracker "pre-flop raise first in" pct for raise chair
pt_rpf_cr	Poker Tracker "pre-flop called raise" pct for raise chair
pt_rpfats	Poker Tracker attempt to steal blinds for raise chair
pt_rwsdp	Poker Tracker went to showdown percentage for raise chair
pt_rwssd	Poker Tracker won $ at showdown for raise chair
pt_rfsbts	Poker Tracker folded small blind to steal for raise chair
pt_rfbbts	Poker Tracker folded big blind to steal for raise chair

Tournament symbols

Symbol	**Explanation**
ptt_iconx (x=0-9)	Poker Tracker auto-rate icon code for chair x
ptt_iconlastrx (x=1-4)	Poker Tracker auto-rate icon code for the last raiser in round x
ptt_pfrx (x=0-9)	Poker Tracker pre-flop raise percentage for chair x
ptt_aggtotx (x=0-9)	Poker Tracker total aggression for chair x
ptt_aggtotnopfx (x=0-9)	Poker Tracker total aggression excluding preflop for chair x
ptt_aggpx (x=0-9)	Poker Tracker preflop aggression for chair x
ptt_aggfx (x=0-9)	Poker Tracker flop aggression for chair x
ptt_aggtx (x=0-9)	Poker Tracker turn aggression for chair x
ptt_aggrx (x=0-9)	Poker Tracker river aggression for chair x
ptt_floppctx (x=0-9)	Poker Tracker saw flop pct for chair x
ptt_turnpctx (x=0-9)	Poker Tracker saw turn pct for chair x
ptt_riverpctx (x=0-9)	Poker Tracker saw river pct for chair x
ptt_vpipx (x=0-9)	Poker Tracker VP$IP for chair x
ptt_handsx (x=0-9)	Poker Tracker number of hands that are in the database for chair x
ptt_pf_rfix (x=0-9)	Poker Tracker "pre-flop raise first in" pct for chair x
ptt_pf_crx (x=0-9)	Poker Tracker "pre-flop called raise" pct for chair x
ptt_pfatsx (x=0-9)	Poker Tracker attempt to steal blinds for chair x
ptt_wsdpx (x=0-9)	Poker Tracker went to showdown percentage for chair x
ptt_wssdx (x=0-9)	Poker Tracker won $ at showdown for chair x
ptt_fsbtsx (x=0-9)	Poker Tracker folded small blind to steal for chair x

ptt_fbbtsx (x=0-9)	Poker Tracker folded big blind to steal for chair x

Tournament symbols for the "raischair"

Symbol	**Explanation**
ptt_ricon	Poker Tracker auto-rate icon code for raise chair
ptt_rpfr	Poker Tracker pre-flop raise percentage for raise chair
ptt_raggtot	Poker Tracker total aggression for raise chair
ptt_raggtotnopf	Poker Tracker total aggression excluding preflop for raise chair
ptt_raggp	Poker Tracker preflop aggression for raise chair
ptt_raggf	Poker Tracker flop aggression for raise chair
ptt_raggt	Poker Tracker turn aggression for raise chair
ptt_raggr	Poker Tracker river aggression for raise chair
ptt_rfloppct	Poker Tracker saw flop pct for raise chair
ptt_rturnpct	Poker Tracker saw turn pct for raise chair
ptt_rriverpct	Poker Tracker saw river pct for raise chair
ptt_rvpip	Poker Tracker VP$IP for raise chair
ptt_rhands	Poker Tracker number of hands that are in the database for raise chair
ptt_rpf_rfi	Poker Tracker "pre-flop raise first in" pct for raise chair
ptt_rpf_cr	Poker Tracker "pre-flop called raise" pct for raise chair
ptt_rpfats	Poker Tracker attempt to steal blinds for raise chair
ptt_rwsdp	Poker Tracker went to showdown percentage for raise chair
ptt_rwssd	Poker Tracker won $ at showdown for raise chair
ptt_rfsbts	Poker Tracker folded small blind to steal for raise chair
ptt_rfbbts	Poker Tracker folded big blind to steal for raise chair

5.19 Memory Symbols

Introduction

One of the most powerful features of OpenHoldem is its ability to remember. Very few other bots have this ability.

Planning

Working with memory symbols requires a bit of planning and specifically one should keep the following in mind:

- OpenHoldem does not know your memory symbol before you create it. This sounds simple but one should not forget it because it could create problems with your fist hand.
- Once a memory symbol is created, and a value is stored, OpenHoldem will remember that value until it is altered or replaced by another or until OpenHoldem is closed. This is great if you want OpenHoldem to remember what happened 10 hands ago but could bite you if you forget to reset them.
- It is of utmost importance to decide when and what should be remembered.

Formulation

The formulation of the memory symbol is as follows:

me_st_abc_123_45 - 'me_st_' stores a value "123.45" in variable "abc" (use "_" for the decimal in values)

A function name can also be passed in, instead of a number, for example: 'me_st_def_f$myfunc' would store the results of function *f$myfunc* in variable “def”.

me_re_abc - 'me_re_' retrieves the value from variable "abc"

Putting it together

To make things clear we will be putting two memory symbols into your bot.
A) We will create a “Stack to Pot Ratio” memory symbol (the function of this symbol is to store the ratio of our balance to the pot at the start of the betting on the flop to be retrieve later in order to make commitment decisions), and
B) We will create a “Slow Play” memory symbol, (the function of this symbol is to store our decision during a betting round that we will check-raise the next betting round regardless of what the next card is)

STEP 1: Create the symbols:
In *f$alli* we put:

```
(br==1 && !(didchecround1 || didcallround1 || didraisround1 ||
didswaground1) ? me_st_stpr_0 : 0) + (br==1 && !(didchecround1 ||
didcallround1 || didraisround1 || didswaground1) ?
me_st_slowplay_0 : 0) +
```

English translation: it is preflop and we have not yet acted then we will create the memory symbols “stpr” (stack to pot ratio) and “slowplay” and we will reset the values to zero.
Comment: we do this in *f$alli* because it is the first formula that is evaluated by OpenHoldem. If we had put it in a formula that is not evaluated then no value will be stored.

STEP 2 (a):
In *f$alli* we put

```
(br==2 && !(didchecround2 || didcallround2 || didraisround2 ||
didswaground2) ? me_st_stpr_f$stpr : 0) + (br==2 && !(didchecround2
|| didcallround2 || didraisround2 || didswaground2) ?
me_st_slowplay_f$slowplay : 0)
```

English translation: it is on the flop and we have not yet acted then we will store the values of the functions f$stpr and f$slowplay at that moment.
Comment: Again this is the first thing we want OpenHoldem to do. It is very important to decide exactly when to pass the value into the memory symbols. Please note that the last line does not have “+”

STEP 2(b)
We create the functions whose value we want to store:

f$stpr

```
balance/potcommon
```

Comment: we don't use "pot" because that would include any bets during round2 before us.

f$slowplay

```
br==2 && nhandshi==0 ? 1 : 0
```

Comment: this formula will return 1 when we have the nuts and this will be stored into the "slowplay" memory symbol

STEP 3
Retrieve and use the memory.
In f$srai we put

```
br==3 && me_re_slowplay && me_re_stpr<10 ? balance : 0
```

English translation: It is the turn and we, on the flop, decided to slow play and the ratio between the pot (at the start of br==2) and our balance at that time was 1:10 and therefore we will shove.

Conclusion

The memory symbols are very powerful but need a bit of planning. It is important to decide when to define, store and retrieve the values in these symbols.

5.20 f$chat

OpenHoldem's chat feature is provided for two reasons: deception and fun. The key to chat is to use it wisely and to not overdo it.

OH-Script

Create a new formula *f$chat* with OpenHoldem, which returns 0, if you wish no chat and one of the constants described below to send a simple chat message. These constants are not available as OpenHoldem symbols, you have to use the numerical values (gg=1, nh=2, ...).

OH-Script with DLL

Simply define *f$chat* as *dll$chat* and handle this symbol in "query_symbol" in your DLL. Return the same constants as used above to specify the message to send, especially "no_simple_Chat" to send no message.

DLL only

Using Chat in your DLL gives you two additional opportunities:

- Any chat message you want
- At any time

At startup your DLL will receive an additional message "p_send_chat_message", providing a pointer to access the chat function. The function has the following prototype:

```
void SendChatMessage(char* new_message)
```

Available chat messages

A complete list of available symbols for formula coders is defined in “PokerChat.hpp”

```
enum
{
no_simple_Chat = 0,
simple_Chat_gg,
simple_Chat_nh,
simple_Chat_vnh,
simple_Chat_n1,
simple_Chat_wow,
simple_Chat_lol,
simple_chat_rofl,
simple_Chat_haha,
simple_Chat_tu,
simple_Chat_thx,
simple_Chat_omg,
simple_Chat_sick,
simple_Chat_fu,
simple_Chat_fish,
simple_Chat_hi,
simple_Chat_hello,
simple_Chat_bye,
};
```

Safety measure:

To prevent a crazy bot flooding the chat, there are two settings in {Preferences}:

- Minimum delay (seconds)
- Additional random delay (seconds)

Adjust these to any value you are comfortable with, if you trust your bot.

Limitations:

- One message at a time. There is no waiting list, or queue, to handle multiple messages (as we are lazy and assume, there's no need for too much chat at a poker table).
- There is no way to specify any concrete delay. The messages will be sent only when {The Autoplayer} is active and there will be an additional delay per character, the intent being to simulate human-like input.
- Only predefined chat messages if you are not using a DLL.
- No support for non-alpha-numeric-characters, due to technical reasons.
- On some sites that temporarily block the chat (e.g. Cakepoker), multiple chat messages can queue up in the poker client. The obvious counter measure would be to always delete the chat box, but that could raise a red flag on other sites.

Chat and Table Maps

In order to use the chat function, OpenHoldem has to know where the chat box of your casino software is. Therefore the corresponding TableMap has to specify a region that defines the location of the chat box. The name of this region is “chatbox“, and describes a rectangle with no transform.

Without the chat box region, all attempts to chat will be ignored without any warnings, avoiding anything that could disturb the screen scraping process.

5.21 Validator

Overview

The Validator is a new tool introduced with OH 2.0.0, that is aimed to help people to develop more reliable bots and Table Maps. It will execute several hundred consistency-checks at the symbol level to detect invalid game-states, mis-scrapes or incomplete information. The Validator will write possible problems to the log-file and also show a message box, as long as messages are not disabled in {Preferences}.

This feature is being actively developed, please see the forum for more information: http://www.maxinmontreal.com/forums/viewtopic.php?f=189

Preferences

The Validator offers several options, available at "Edit -> Preferences -> Validator".

- **Never:** The Validator will be turned off completely. Not recommended for real-money-play.
- **When it is my turn**: The Validator will check the rules, when it is your turn, i.e when buttons are visible. This setting offers a good tradeoff between reliability and performance. Recommended for real-money-botting.
- **Always**: The validator will check all rules once per heartbeat-cycle. This setting will detect errors, even if it is not your turn, but it causes also a higher CPU-usage. Recommended for verification of new or modified TableMaps and for testing of bots.
- **Use heuristic rules**: Some rules are heuristic, i.e. they are "common sense", but not always true. Example: A game does usually not last longer than 2 minutes, so the Autoplayer has to act at least once per 120 seconds. If it does not do so, there seems to be a problem. Maybe the Table Map does not detect the players cards or the buttons. This setting can result in false positives and is recommended for testing only, but not for real-money-play.
- **Stop Autoplayer on error**: This setting will quit playing immediately, if an error occurs. Recommended for testing, if you want to investigate the error immediately, and maybe for very conservative real-money-botting.

Technical documentation

The Validator basically consists of a set of several hundred rules to be checked. Each rule is written in pseudo-code, that can be translated into C-macros easily (there is a little Perl-script for that purpose, "rules2cpp.pl"). Each rule consists of:

- a unique ID for the test case
- a flag to distinguish heuristic and non-heuristic rules
- some verbal reasoning.
- a precondition, that checks, if a rule can be applied to the current game state
- a postcondition, that has to be met, if a rule can be applied
- a list of symbols, that got used and are possibly affected in case of an error

In practice it looks like this:

```
BEGIN_TESTCASE
    TESTCASE_ID ("0501")
    HEURISTIC_RULE (false)
    REASONING ("If it's my turn, then there has to be always a check
or a call button.")
    PRECONDITION ("ismyturn")
    POSTCONDITION ((("myturnbits" & BUTTON_CHECK) != 0) ||
(("myturnbits" & BUTTON_CALL) != 0))
    SYMBOLS_POSSIBLY_AFFECTED ("ismyturn, myturnbits")
END_TESTCASE
```

5.22 Log Files

OpenHoldem will generate a number of log files as it operates. The verbosity of the standard log and the Poker Tracker log can be set in {Preferences}.

Standard Log

This contains key actions that OpenHoldem sees, such as connection to a new table, table reset, and hand resets, player actions, as well as detailed before and after Autoplayer action information. The before-after Autoplayer action information is saved as a single line with the following format:

```
# hand commoncard rank poker  win  los  tie  P      nit bestaction - play*      call
bet       pot   balance - FCRA FCRA swag
```

From left to right, the fields in this line are:

- # - Number of chairs at the table (as defined in the Table Map)
- hand - your hole cards
- commoncard – the board cards
- rank - your hand rank (as specified in {Preferences})
- poker - your poker hand
- win / los / tie - the prwin/prlos/prtie Iterator results
- P - the results from your *f$P* formula
- nit - your NIT setting from the Table Map
- bestaction - the action that the Autoplayer should take, based on the primary function results
- play - the actual action taken by the Autoplayer (may not be the same as best action, if the best action button is not present, for example)
- call / bet / pot / balance - the value of *call*, *bet*, *pot* and *balance* symbols
- first FCRA - the status of the fold, call, check, raise and allin buttons as seen by the scraper engine
- second FCRA - the status of the *f$call*, *f$rais* and *f$alli* functions
- swag - the value of the *f$swag* function

PokerTracker Log

The Poker Tracker log is created if you are connected to a Poker Tracker database, and depending on the level of verbosity, will provide exact query SQL that is sent to the database when statistics are needed.

f$debug log

This log is created when set from the menu options in the Formula Editor window.

Autoplayer Trace

This is an example of the log entries that are added to the standard log if the Autoplayer Trace option is enabled in {Preferences}.

```
************************************************************
HAND RESET (num:0 dealer:2 cards:KsKc): Poker Academy Pro
************************************************************
ROUND 1
>>> New hand 0
>>> My turn, br=1
>>> Chair 2 (6lleilli) posted the sb: $0.00
>>> Chair 3 (GusXensen) posted the sb: $0.00
>>> Chair 4 (JackPott) posted the sb: $0.00
>>> Chair 5 (Kale) posted the sb: $0.00
>>> Chair 1 (DanielXn) posted the sb: $0.00
log$ (Total: 19 | Showing: 19)
    f$swag=40.00 [Line: 1, Col: 49]
          f$srai=40.00 [Line: 4, Col: 9]
                f$preflop_srai=40.00 [Line: 12, Col: 87]
                      f$preflop_position=4.00 [Line: 5, Col: 50]
                            nplayersdealt=6.00
                            dealposition=6.00
                      ncallbets=1.00
                      nopponentsbetting=3.00
                      islist16=1.00
                      bblind=10.00
                br=1.00
          f$raise_min_amount=10.00 [Line: 1, Col: 12]
                currentbet=0.00
                call=10.00
          f$swag_adjust=0.00 [Line: 3, Col: 1]
                swagtextmethod=1.00
    f$call=1.00 [Line: 7, Col: 4]
          f$srai=40.00 [Cached]
          call=10.00
6 KsKc ..........    2 1pair  578  417    5  3     1000 $40.00     - SWAG      10.00
 10.00     25.00      0.00 - FCRA .C.. 40.00
```

How to read the log

f$swag returned 40 processing line 1, character 49. It called the following functions: f$srai, f$raise_min_amount, f$swag_adjust, and used no other symbols.

f$preflop_position (returning on line 5 processing character 50) was called from f$preflop_srai which was called from f$srai. It called no other functions and used 2 symbols; nplayersdealt, dealposition whose values were 6 and 6.

```
ROUND 2
log$ (Total: 34 | Showing: 34)
    Monster_Hand
    checkLast_Monster_Raise
    f$swag=115.33 [Line: 1, Col: 49]
          f$srai=115.33 [Line: 10, Col: 9]
                f$postflop_srai=115.33 [Line: 6, Col: 31]
                      f$postflop_scenario=2.00 [Line: 7, Col: 77]
                            f$round_first_action=1.00 [Line: 2, Col: 38]
                                  didswag=0.00
                                  didrais=0.00
                                  didcall=0.00
                                  didchec=0.00
                            betposition=3.00
                            ncallbets=0.00
                            nplayersround=3.00
                      f$flop_checklast_srai=115.33 [Line: 2, Col: 67]
                            f$handstrength=1.00 [Line: 8, Col: 105]
```

```
                                    f$c_hs_MONSTER=1.00 [Line: 1, Col: 1]
                                    br=2.00
                                    vs$80$prlos=0.06
                                    log$Monster_Hand=1.00
                              f$c_hs_MONSTER=1.00 [Cached]
                              f$cbet_amount=115.33 [Line: 1, Col: 23]
                                    random=0.60
                                    pot=160.00
                              log$checkLast_Monster_Raise=1.00
                        br=2.00
                  br=2.00
            f$raise_min_amount=0.00 [Line: 1, Col: 12]
                  currentbet=0.00
                  call=0.00
            f$swag_adjust=0.00 [Line: 3, Col: 1]
                  swagtextmethod=1.00
      f$call=1.00 [Line: 5, Col: 7]
            call=0.00
6 KsKc JsKd4c....    2 3kind  915   85    0  2     1000 $115.33    - SWAG       0.00
 10.00    160.00      0.00 - FCRA .C.. 115.33

log$ (Total: 30 | Showing: 30)
    Monster_Hand
    reraise_Monster_BIGRaise
    f$swag=2340.00 [Line: 1, Col: 49]
          f$srai=2340.00 [Line: 10, Col: 9]
                f$postflop_srai=2340.00 [Line: 12, Col: 31]
                      f$postflop_scenario=5.00 [Line: 9, Col: 30]
                            f$round_first_action=0.00 [Line: 2, Col: 2]
                                  didswag=1.00
                            didswag=1.00
                            nbetstocall=78.00
                      f$flop_reraised_srai=2340.00 [Line: 2, Col: 68]
                            f$handstrength=1.00 [Line: 8, Col: 105]
                                  f$c_hs_MONSTER=1.00 [Line: 1, Col: 1]
                                  br=2.00
                                  vs$80$prlos=0.06
                                  log$Monster_Hand=1.00
                            f$c_hs_MONSTER=1.00 [Cached]
                            f$raise_big_amount=2340.00 [Line: 1, Col: 5]
                                  pot=1170.00
                            log$reraise_Monster_BIGRaise=1.00
                      br=2.00
                br=2.00
          f$raise_min_amount=895.00 [Line: 1, Col: 12]
                currentbet=115.00
                call=780.00
          f$swag_adjust=0.00 [Line: 3, Col: 1]
                swagtextmethod=1.00
    f$call=1.00 [Line: 7, Col: 4]
          f$srai=2340.00 [Cached]
          call=780.00
6 KsKc JsKd4c....    2 3kind  969   31    0  1     1000 $2340.00   - SWAG     780.00
 10.00   1170.00      0.00 - FCRA .C.. 2340.00
```

5.23 Command Line Options

Version 2.0.0 of OpenHoldem introduced the ability to instruct OpenHoldem to use .ini files for storage of preferences, rather than using the registry. This is beneficial for two reasons:

- Stealth: .ini files are easier to hide than the registry
- Portability: .ini files can easily be deployed and moved from machine to machine

To specify the use of .ini files for preferences, rather than the registry, there are three options:

- “/ini” specifies the use of openholdem.ini, created in the OpenHoldem startup directory
- “/ini:[filepath]” specfies the use of [filename] as the ini file, must be a full path
- “/ini:~\[filename]” specifies the use of [filename] in the OpenHoldem startup directory as the ini file

5.24 Advanced Topics

Version 2.0.0 of OpenHoldem introduced the ability to instruct OpenHoldem to connect to a poker table by sending a connect message specifying a HWND. This is useful for automation reasons and permits you to explicitly control the connection process without having to display the OpenHoldem attach window. In fact, you can have OpenHoldem hidden entirely during this process.

Example AutoIt Script

```
#include <WinAPI.au3>
Func AttachOH($OHhwnd,$POKERhwnd)
      _SendMessage($OHhwnd, 0x8000+2, 0, $POKERhwnd )
      Return
EndFunc
```

ManualMode & OHReplay

6.1 ManualMode

Description

ManualMode is a tool for simulating a poker game state offine. It allows you to set any game state you can think of, including your cards, players' cards/backs, names, balances, blinds, game types, button, bets, buttons, etc.

A Table Map is included with the ManualMode tool to allow OpenHoldem to connect to the ManualMode main window and scrape the game state 100% correctly into the OpenHoldem engine. This will allow for offline construction of bot logic and the easy setup of various game state situations, some of which are very rare online (think Straight Flushes).

Familiarizing yourself with ManualMode

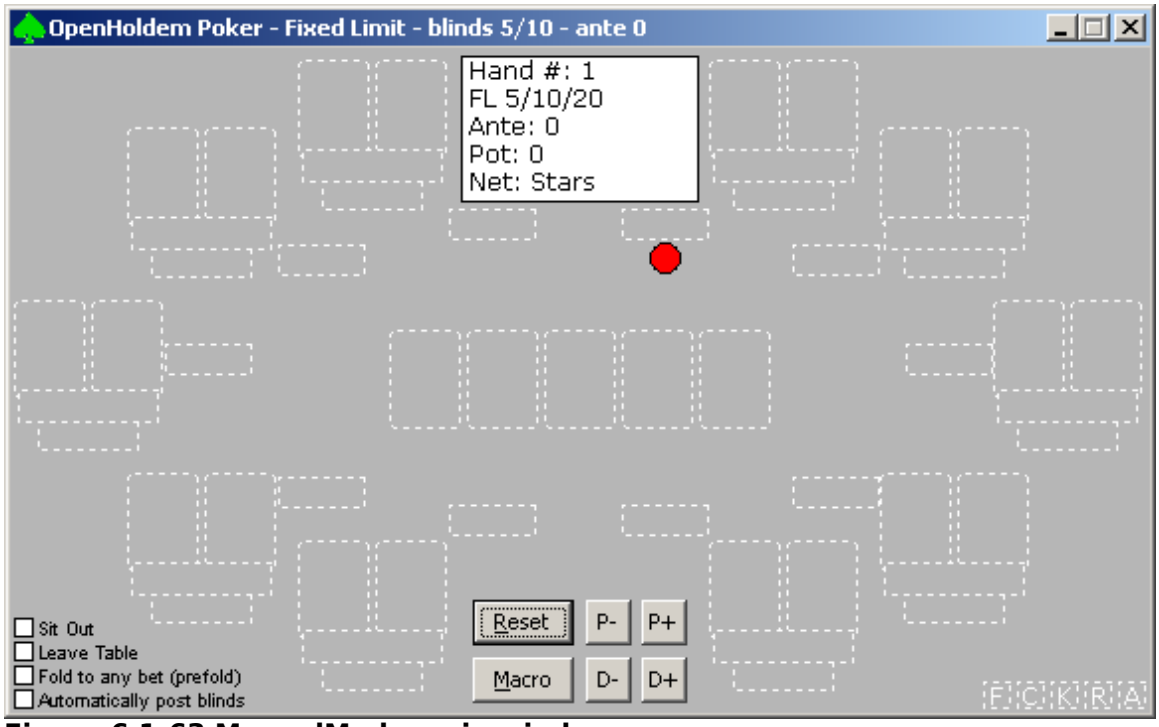


Figure 6.1-63 ManualMode main window

The ManualMode interface is quite intuitive. All relevant elements on the poker table are visible at all times, either populated with objects (like cards or bets), or outlined in white dashed lines if not populated. Each game state element is interacted with slightly differently and is discussed below. In general, however, right-clicking on any element will bring up a list of options you can take to set that element's state.

- **Common Cards**: Right-clicking on each common card will bring up a context menu allowing the selection of rank/suit, or "no card".
- **Sit Down/Stand Up**: OpenHoldem recognizes the difference between sitting in at a chair, and sitting in a game and being dealt a hand. Right-clicking on either

of a player's cards will bring up a context menu allowing the player to Sit Down or Stand Up from the selected chair.
- **Sit In/Sit Out**: OpenHoldem recognizes the difference between sitting in at a chair, and sitting in a game and being dealt a hand. Once you have "Sat Down" at a chair, right-clicking on either of a player's cards will bring up a context menu allowing the player to Sit In to the game or Sit out from the game.
- **Player Cards**: Right-clicking on either of a player's cards will bring up a context menu allowing the selection of rank/suit, "cardbacks" or "no card". Note: prior to assigning cards, a player must have "Sat In" the game.
- **Player Names**: Right-clicking on the player's name rectangle will pop up a window prompting for a string to use as the player's name. Note: prior to assigning names, a player must have "Sat Down" at the table.
- **Player Balances**: Right-clicking on the player's balance rectangle will pop up a window prompting for a string to use as the player's balance. Note: prior to assigning balances, a player must have "Sat Down" at the table.
- **Player Bets**: Right-clicking on the player's bet rectangle will pop up a window prompting for a string to use as the player's bet. Note: prior to assigning bets, a player must have "Sat In" the game.
- **Dealer Button**: Right-clicking on either of a player's cards will bring up a context menu allowing the assignment of the dealer button to the selected chair. The "D-" and "D+" buttons can also be clicked to move the dealer button around the table anti-clockwise or clockwise.
- **Blinds/Limits**: Right-clicking on the center information box will pop up a window in which the small blind, big blind, big bet, ante, game type (FL, PL, NL), network (useful if integrating with {PokerTracker}), and tournament indicator can be set.
- **Buttons**: Left-clicking on any game-state button on the table will toggle its state from on to off. These include the FCKRA buttons on the bottom right (fold, call, check, raise, allin), and the "play" buttons on the lower left (sit out, leave table, prefold, autopost).

There are some shortcuts built into ManualMode to allow easier setup of game states:
- "P-" and "P+" buttons: These buttons will remove and add players from the table. As players are added, they will Sit Down, Sit In, and will be assigned a name and balance. Players will be added from chair 0 to chair 9, depending on what is open, and removed in reverse order.
- Macro: Right clicking on the "Macro" button will open a window allowing for a string to be entered. Left clicking on the "Macro" button will execute that macro. The format of the macro string is described here: {Macros}
- Betting: Left-clicking on a player's bet rectangle will call the current bet. Shift-left-clicking on a player's bet rectangle will raise. (right click to set a specific SWAG).

Note: When ManualMode detects a betting round change (i.e. 0 to 3, 3 to 4 or 4 to 5 common cards), then all player bets will be collected into the pot, and all player bets will be reset to zero.

ManualMode Options

Clicking on the System Menu (green spades symbol) on the ManualMode window will bring up the system context menu. On that menu is an entry for "Options...". Selecting this entry will bring up the following window:

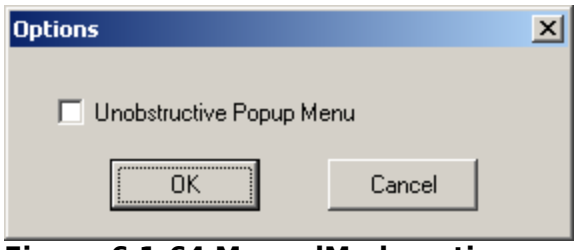


Figure 6.1-64 ManualMode options

On this window is only one option at the moment, to move the right-click popup menu so it does not show up on top of the ManualMode window. The reason for this is that the popup menu will overlay the game state elements, and cause havoc if you have connected an OpenHoldem instance to this ManualMode window. The drawback is that the popup menu appears in a different location from where you right-clicked, and is somewhat disconcerting until you get used to it. Thus the choice it yours – pick whichever method annoys you the least.

Macros

A macro string is a series of characters executed one at a time, left to right. The meaning of each character is as follows:

Character	**Meaning**
R (uppercase)	Reset the game state
P (uppercase)	Sit in a player at this chair, default name, default balance, and cardbacks.
p (lowercase)	Unseat a player at this chair.
b (lowercase)	Set the small blind at this chair
B (uppercase)	Set the big blind at this chair
23456789tjqka	Set the rank of the players cards at this chair, or common cards (player first, common next)
cdhs	See the suit of the players cards at this chair, or common cards (player first, common next)
n	Set the dealer to this chair

The default macro provided (“RPPPPPPPPPPbB”) will reset the table, set 10 players, and set chair 0 as the small blind and chair 1 as the big blind.

A macro of “RPPnAhAsPbPBPPP” will reset the table, seat 7 players, assign the user to chair 1 with the button and deal Ah As as hole cards, assign chair 2 to the small blind and assign chair 3 to the big blind.

6.2 OHReplay

Description

OHReplay is an application for displaying {Replay Frames}. That’s all it does. The nice thing is that OHReplay displays frames exactly as they are displayed by the casino’s poker client. That means that once a frame is displayed by OHReplay, you can connect OpenHoldem to that displayed frame just as if you were connecting to

the native poker client window. This allows for easy debugging of your Table Map and bot logic while offline.

Familiarizing yourself with OHReplay

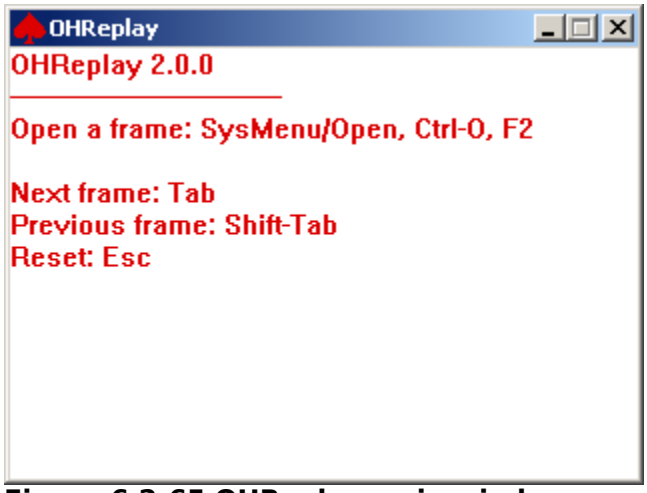


Figure 6.2-65 OHReplay main window

The initial window displayed by OHReplay when it starts up contains all the information you need to know to navigate the program:

- To open a frame, use the System Menu, the Ctrl-O keystroke combination (that is a capital oh, the letter between N and P), or the F2 key. Any of these choices will open the standard Windows file open dialog, from where you can choose the HTML file of the frame to open.
- To navigate between frames, use Tab or Shift-Tab to move to the next or previous numerical frame.
- To return to the initial screen, press Esc.

OHReplay Options

Clicking on the System Menu (red spades symbol) on the OHReplay window will bring up the system context menu. On that menu is an entry for “Open…”. Selecting this entry will bring up the same file open dialog as using F2 or Ctrl-O.

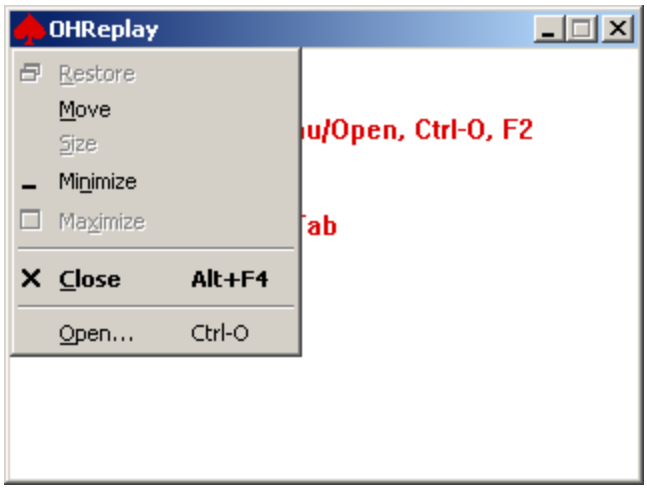


Figure 6.2-66 OHReplay options

Stealth

An important consideration for any botter is that of stealth. Many casinos will attempt to detect the OpenHoldem software on your computer, and will either warn you or seize your account's funds if they find it. There are two general approaches to addressing this problem, a "Single System" solution and a "Dual System" solution. OpenHoldem provides a number of facilities for achieving both of these setups and obscuring the program from detection.

Note that a "System" as discussed in this chapter can refer to a physical machine or a virtual machine. VMware Workstation is a popular choice for consumer system virtualization, but certainly you can make use of the other consumer virtualization products available, such as VirutalPC or VirtualBox. Addtionally, dasta center-class, bare metal hypervisor products such as VMWare ESX or Citrix XEN will work fine as well. See the {VMWare} section for information on configuring VMWare Workstation. Configuration of other virtualization platforms is not discussed in this documentation.

7.1 OpenHoldem Single System Configuration

In this scenario, both the poker client software and OpenHoldem reside on the same machine. In this case, reference the {Registry}, {Obscure Preferences} and {Ninja Folders} sections at a minimum. Review of the {Rootkits} section is recommended.

7.2 OpenHoldem Dual System Configuration

In this scenario, the poker client software resides on "System A", and the OpenHoldem software resides on "System B". As the OpenHoldem software resides on a wholly different machine than the poker software, less thought needs to be given to protecting detection of the OpenHoldem software. OpenHoldem connects to the poker client windows using the {Bring} software as an intermediary. It should be obvious, but in order to make this scenario work, you either need two physical machines or two virtual machines. {Rootkits} still may be needed to obscure the network ports and traffic between the two systems.

7.3 Registry

By default, OpenHoldem will store its preferences and settings in Windows' registry. Unfortunately, a quick registry scan of these known keys will alert the casinos to the presence of OpenHoldem on the machine. It is highly recommended that you make use of the "/INI" command line option to save these preferences and settings in a ".ini" file, that you name this file something clever and store it in a location that is not predictable. See the {Command Line Options} section for more information.

7.4 Obscure Preferences

By default, OpenHoldem uses names for window titles, mutexes, and class names that are easily detectable. It is highly recommended that you change these settings in {Preferences} to something non-obvious.

7.5 Bring

"Bring" is a software program developed by Ray Bornert and can be found here: http://www.winholdem.net/bring/index.html

The sole purpose of Bring is to enable a dual-system configuration, and allow the casino's poker windows from "System A" to be displayed on "System B" where OpenHoldem can connect to them.

7.6 Ninja Folders

Original post here: http://forum.winholdem.net/wbb/viewtopic.php?t=4749

This article explains the basic steps for creating a windows folder that will be as secure as it can possibly be under the NTFS file system.

The general goal is to create a folder that is owned by a single windows user account such that no other user account may access the folder - not even the system or the admin account.

Prerequisites:
1) knowledge of creating/modifying user account
2) knowledge of windows explorer
3) knowledge of NTFS permissions

It is assumed that the reader understands how to create/modify a user account and how to use Windows Explorer to navigate folder space and how to modify folder properties/permissions.

Step 1 - login to your admin account

Step 2 - create/decide target account (from admin account)

This account will own the hyper-safe folder. For the purpose of this example, the account name "**ninja**" will be used. The "**ninja**" account must meet the following requirements:

a) it cannot be the Administrator account
b) it cannot be part of the Administrator group
c) it cannot be part of any group except the default users group.

If you need/want to create a special "**ninja**" account then do that first according to the restrictions above. If the account already exists then make sure it meets the requirements above.

Step 3 - create the target folder (from admin account)

Your goal here is to create/decide the target folder. In an ideal world you would create a top level folder on an empty local (non-shared/not-networked) hard drive where "**ninja**" has full control over that disk. If this is not possible then the next best thing is a top level folder on a non-shared drive. If this is not possible then you must select a location where "**ninja**" will have access to the parent folder so that it can access its private folder.

Step 4 - permissions (from admin account)

Select the properties for the folder and select the security tab add "**ninja**" to the access list and grant "**ninja**" full control. Remove all other users/groups from the access list, which will effectively shut out the entire world. There should be exactly

one account in the access list and that account should be “<host>/**ninja**"

Uncheck/unselect "Allow inheritable permissions from parent to propagate to this object". The reason for this is that you want this folder and it's permissions to be the root node for all children below it. If you do not do this then permissions defined for the parent above your root folder will filter down to your folder and the children in your folder and you do not want that. Note that you cannot give ownership of objects in NTFS, you can only "take" ownership with the account you're currently using. You will take ownership of the folder later when you login to the "**ninja**" account.

Step 5 - logout of the admin account

Step 6 - login to your "**ninja**" account

Step 7 - ownership (from **ninja** account)

Select the properties for your "safe" folder and verify that "**ninja**" is the only account in the access list. Click the advanced button and goto the owner tab and select the "**ninja**" account. Check "Replace owner ..." and click "Apply". You should now be the full owner of your "safe" folder and everything below it.

Step 8 - permissions (from **ninja** account)

Click the permissions tab and check "Reset permissions on all child objects ...". Uncheck "Allow inheritable permissions from parent ..." (note this should already be unchecked since you did that from the admin account), then click apply. This will go quickly if the folder is empty. It can take seconds or minutes or more depending on the children folder tree below you.

Step 9 - verify

Create a test child folder inside your "safe" folder, view the properites and verify that "**ninja**" is the owner with full control and that no other account has access. If this is not the case then you missed a step above (probably the "Allow inheritable permissions ..." in step 4).

Step 10 - logout of the "**ninja**" account

Step 11 - login to the admin account

Step 12 - verify

Try to access the "safe" folder. You should not be able to access the contents of the folder nor modify the folder properties/permissions. As an admin you can still take ownership of the folder but until then the admin access is denied.

Other thoughts
If you have the resources and you want the best possible solution then you should dedicate an entire physical hard drive to your stealth department. That drive should not be the bootable system drive, but should be a secondary drive. You should consume all 4 primary partitions on your stealth drive by creating 4 primary partitions of roughly equal size. Each partition should be formatted with NTFS only (not FAT or FAT32).

You should then make the "**ninja**" account the owner of all 4 partitions and the sole

account with access. There should be no other accounts listed in the access list. The drives should not be networked or shared. If you do this you will notice that your admin account cannot even read the device label and that the 4 hard drives will be seen only as "local-drive". All access from any accounts other than "**ninja**" will be denied.

The benefit of this setup is that once you construct a hard drive like this you dont have to worry about permissions ever again in the context of that drive - you know that anything you do on that drive must be done from the "**ninja**" account and that any folder/files created on that drive are entirely safe.

Casino software
If the casino software demands admin privs to install/update/run, and if you've successfully accomplished all 12 of these steps, then their software running as admin cannot automatically read your stealth drive(s) / folder(s) - the contents are very much private. However, if they are running as admin they could physically take ownership of the folder in order to gain access.

In this context, it must be said that such an event could be viewed as a criminal act in that there is absolutely no need whatsoever for their software to decide to "take" physical ownership of a folder just because it wants to.

An advanced step (not covered in this article) is to install/enable file/folder monitoring/auditing that will allow you to track anything that happens to your "safe" folder. Note that doing this would not stop admin/casino software from taking ownership but you'd have a record of the event which is better than nothing.

7.7 VMWare

Original post by Tammelinn here: http://forum.winholdem.net/wbb/viewtopic.php?t=11894

Guide to using OpenHoldem with VMware in 21 easy steps
by Tammelinn

This is a guide to show you how to get OpenHoldem running securely on a single computer using the VMware Server application as a second, "virtual" computer.

The guide assumes no special knowledge, but you will need some basic computer skills. You will need to know how to install software, how to rename files, how to navigate folders and so on.

I am also assuming that you are using the same operating system as me. If you are using Windows XP Media Centre Edition or Windows XP Professional, you can use this guide. If you are using some other brand of Windows (including Vista), the guide might not be reliable. I have only tested these instructions with XP.

Much of this information is already available elsewhere. In many cases, all I have done is to reword it a little.

Definitions

The "real" computer is the one that is sitting on your desk or on your lap. It has its own operating system. The "virtual" computer is the one we are going to create using VMware Server. We can think of the "virtual" computer as a separate computer

with its own operating system, even though it is physically part of the "real" computer.

Step 1 - Download VMware Server
VMware Server is a free download from http://www.vmware.com (you will need to provide VMware with a valid e-mail address to get a serial number). I used version 1.0.3 while writing this guide.

You will also need a Windows installation disk. I am using a Windows XP Professional installation DVD, which I'm going to install as my "virtual" operating system.

While you wait for the download to finish, you should read the whole of this guide and make sure that you understand it. It would be a good idea to print it out, if you can.

DO NOT INSTALL ANY SOFTWARE until you are told to.

Step 2 - Install VMware Server on your real computer
Double-click on the VMware Server file you downloaded and it will begin installing itself. When you are asked if you want a "complete" or "custom" installation, choose "complete". A big window with some warnings about "IIS" might appear. We don't need IIS, so just click “OK”. When the installer asks for Internet access, grant it. Wait for the installation to complete before moving on to the next step.

Step 3 - Configure VMware Player
Double-click on the "VMware Server" icon that should have appeared on the desktop of your real computer. VMware Server will start for the first time. A Pop-up window asks you to choose between a local host and a remote host. Choose "local host" and click "OK".

Click the "New Virtual Machine" icon and a window will appear. Choose "typical" and click "next". Now choose which operating system you want to install on your virtual computer. Here, I selected the "Microsoft Windows" option, and the version I selected was "Windows XP, Professional".

The next window gives you two boxes to fill in. The first box gives you a chance to name your new virtual computer. Choose any name you like.

The second box ("Location") allows you to choose where the Virtual computer will be saved on your hard drive. You probably don't need to change the contents of this box. (The virtual computer is actually a single file which exists on the hard drive of your real computer. If you want this file to be in a different folder or even on a different partition, you can tell VMware now.)

Next you must choose the type of network. Choose "Use bridged networking".

The next window asks you to decide how big the virtual computer's virtual hard drive is going to be. The suggested 8GB is quite enough for the purposes of playing poker. Click "Finish" when you are ready.

VMware will spend quite a few minutes creating the virtual computer.

Step 4 - Install an operating system on the virtual computer
When VMware is ready, you'll see a new window with some details about the virtual computer. Find "Start this virtual machine" on the left and click it. VMware will now look for the Windows installation disk in your CD/DVD drive. Put the installation disk

in the drive and Windows will begin installing onto your virtual computer.

Soon, Windows will ask you if you want to format your hard drive. Only the virtual hard drive of the virtual computer is going to get formatted, so you don't need to worry about the data on your real hard drive. It is perfectly safe. If you asked how you want to format the virtual hard drive, make sure you choose one of the NTFS options. (Do not choose any option with FAT16 or FAT32).

Eventually, Windows will ask you to enter your full name. We are going to create three seperate users, so choose a name for the first user ("Jessica", for example) and enter it here. This user will have permanent administrator privileges.

Next, enter an administrator password for this user. Write down the name and password now.

ADMINISTRATOR NAME AND PASSWORD _____

Later, Windows will ask you to enter up to five user names. The first user - the box labelled "your name" - should be the same as the user you entered above ("Jessica"). Don't fill in the other boxes yet.

Wait until the installation is complete. You should be looking at a brand-new installation of Windows XP. If you can run Solitaire (click "Start - All Programs - Games - Solitaire") then you are ready to move on to the next step.

Step 5 - Install VMTools, Firewall and Antivirus on virtual computer
VMTools is a package which, among other things, gets the mouse to move smoothly between the real and the virtual computer. You need to install it now, so press CTRL+ALT to restore mouse control to the real computer. Then, on the VMware Server window, click on the menu option VM, and then choose "Install VMTools..." VMTools will now install itself inside the virtual computer. The virtual computer will need to reboot.

After the reboot, find the VMTools icon in the system tray, next to the clock. It looks like three small interlinked squares. When you have found it, double-click on it.

A window appears and the "Options" tab is already showing. Find the line "Show VMware Tools in the taskbar" and click on it, so that there is no longer a tick next to it. Click “OK” to save your changes. This measure prevents the casino from seeing that you're using VMware - just in case they take a screenshot of your desktop.

Now, your virtual computer is connected to the Internet, so it needs firewall software and antivirus software, just as your real computer does. Install the software (connect to the Internet to download it, if you have to), and then run a scan to check that your virtual computer is free from viruses.

(You could use AVG Antivirus from http://free.grisoft.com, and Zonealarm Firewall from http://www.zonealarm.com)

Step 6 - Check that the real and virtual computers can communicate
If the real computer and the virtual computer can't communicate with each other, all your efforts in securing them will be wasted. Let's check the communications now.

The traditional way to test a network is to have one computer "ping" the other. A ping is a short signal which one computer sends to another. The signal will get

bounced back if the network is running as it should.

Before pinging anything, you need to turn off the firewalls on both computers. If you have installed a firewall such as Zonealarm, switch it off. Now click "Start - Control Panel - Security Centre". At the bottom of the window is an option to "Manage security settings for: Windows Firewall". Click on "Windows Firewall" and make sure it is off. Do this on both computers.

Now you need to find out the IP address for both computers. (An IP address is a numerical address which computers on a network use to identify each other.)

On your real computer, click "Start - Run". Type "cmd" in the box and press return.

A new window appears. Type “ipconfig" and press return. You will see a series of numbers, some of which are marked "IP address". Find the one that corresponds to the type of Internet connection you have - for example, the one labeled "wireless connection" if you connect to the Internet through a wireless router.

The IP address of my real computer is 192.168.1.4 - yours probably looks almost the same. Write it down now.

REAL COMPUTER IP ADDRESS _____

Now do the same thing on your virtual computer. Find the IP address and write it down. The IP address of my virtual computer is 192.168.1.7 - yours will be similar.

VIRTUAL COMPUTER IP ADDRESS _____

On your real computer, find the new window again and type "ping x.x.x.x", then press return. Replace x.x.x.x with the IP address of your virtual computer. You should see something like this...

```
Pinging 192.168.1.7 with 32 bytes of data:

Reply from 192.168.1.7: bytes=32 time<1ms TTL=128
Reply from 192.168.1.7: bytes=32 time<1ms TTL=128
Reply from 192.168.1.7: bytes=32 time<1ms TTL=128
Reply from 192.168.1.7: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.7:
```
(and so on)

This dialogue means that the real computer sent a signal to the virtual computer four times, and each time it got a response.

If the network isn't set up properly, you will see something like this...

```
Pinging 192.168.1.7 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 192.168.1.7:
(and so on)
```

This dialogue means that the real computer sent a signal to the virtual computer four times, but received no response.

Go to your virtual computer and do the same thing. This time, type "ping x.x.x.x" replacing x.x.x.x with the IP address of your real computer. If you get a response, your network is set up properly and you can continue to the next step. If the network isn't set up properly, you should seek help from someone who understands networks.

Don't forget to TURN YOUR FIREWALLS BACK ON AGAIN when you finish this step. If you have a good firewall such as Zonealarm, you can leave the less powerful Windows firewall turned off permanently (on your real computer as well as your virtual computer).

Step 7 - Set up user accounts on virtual computer
Click "Start - Control Panel - User Accounts".

You had already set up one user while installing Windows. This user already has admin privileges. We'll call it the "admin" user. We need to give this user a password. It can be the same password you used in step 4. Click on the user, then click on "create a password". Enter the password into the boxes and click on "create password". The next window asks you if you want to make your files and folders private. You certainly do, so click "Yes, make private". Click on "Back" to return to the user accounts window.

Now we'll set up two new users with fewer privileges. You will be logged in as the second user whenever you connect to a casino. We'll call this the "casino" user. The third user will be created to hide things from the casino. We'll call this the "stealth" user.

Choose names and passwords for these users, and write them all down now.

ADMIN USER _____
CASINO USER _____
STEALTH USER _____

The casino user will have restricted privileges initially. Click "create a new account", enter the name of the casino user and click "next". Click on "limited" and then click on "create account". The casino user will appear next to the admin user. Click on the casino user and give it a password.

The stealth user will have restricted privileges permanently. Click "create a new account" and set up the stealth user in the same way you set up the casino user.

Click "create a new account". Choose a name and then click on "limited account".

That completes this step. If there is a "guest" user shown, you can ignore it. Close the "user accounts" window.

Step 8 - Create a protected folder on virtual computer
You should still be logged in as the admin user. Stay logged in as this user throughout step 8.

We need to create a folder to hide some sensitive information. The best place for

this folder is inside the root folder (click Start - My Computer - Local Disk C:. Then click on "show contents of this folder").

Create a new folder in this window and give it an innocent name. ("Files" would be a good name.) From now on, this folder will be referred to as the "Ninja" folder.

Click "Start - Control Panel - (Appearances and Themes) - Folder Options". Click the "View" tab. In the "Advanced Settings" window, find the line "Use simple file sharing (recommended)". If there is a green tick on this line, click on it to remove it. Click on the big "OK" button to save your changes.

Now, right click on the Ninja folder and choose "Properties". Click on the "Security" tab. In the "Group or user names" box you will see a list of users, but the stealth user should not be one of them.

We want to change the folder's settings so that the stealth user is the only user who can see inside it. (Not even users with admin privileges should be able to see inside.) Click the “Add" button. In the "Enter the object names to select" box, type in the name of the stealth user (for example, my stealth user is called James). Click the "OK" button. There should now be a new entry in the list. My stealth user shows up as “James (XPPRO/James)" in that list.

Now we need to remove every user except the stealth user from the list. Click the "Advanced" button. Find the line which says "Inherit from parent the permission entries..." and click on it. A warning pop-up appears; click on the "Remove" button. The line which says "Inherit from parent the permission entries..." should no longer have a tick next to it. Click the “OK” button. Another warning pop-up might appear. If it does, click "Remove", then “OK”. The list of users should now have only one user in it, the stealth user.

Find the "Full control" line. In the "allow" column, there is a small box. Click on it, and a column full of ticks will appear. If everything has gone to plan, there will be one user with full control over the Ninja folder. If you're happy, click "OK". If something seems to be wrong, click "Cancel" and go through the last two paragraphs again.

Step 9 - Take charge of the protected folder

Log out from the admin user account and login as the stealth user (click Start - Log off - Switch User). Go to "My Computer" again and find the Ninja folder. Right click on the folder and choose "Properties". Click on the "Security" tab. Check that the stealth user is still the only user in the list.

Click the "Advanced" button and click the "Owner" tab. Left click once on the stealth user so that it changes colour to blue. Find the "Replace owner..." line and click on it, so that a green tick appears next to it. Then click on "Apply".

The user in the box "Current owner of this item" should have changed to the stealth user.

Now click on the "Permissions" tab. Find the "Inherit from the parent permission entries..." line; there should still be no tick next to it. Just below, find the "Replace permission entries on all child objects..." line; click on it so that there is a green tick next to it. Click the "Apply" button. A warning pop-up appears; click the "Yes" button. Click on the “OK” buttons twice to save your changes.

Let's check that everything is set up correctly. The stealth user is the only one who should be able to see the Ninja folder and ANYTHING it contains.

Open the Ninja folder (by double clicking on it) and create a folder called "test" inside it. Right click on the "test" folder, choose "Properties" and click on the "Security" tab. If the stealth user is the only person in the list, everything is as it should be.

Close all the windows and log out from the stealth user account.

Step 10 - One more check
Log in as the admin user. Find the Ninja folder, and try to look inside it. Your attempt to see inside this protected folder should be denied. When you run casino software, it will not be able to see inside this folder (unless it tries to take ownership of the folder - but this would be a CRIMINAL act.)

This setup is not 100% secure, but it is probably secure enough.

Step 11 - Install Bring on your virtual computer
You now have a real computer and a virtual computer. OpenHoldem is going to run on the real computer. The casino software is going to run on the virtual computer.

Now you need to install some software that will allow OpenHoldem (on the real computer) to see what the casino software (on the virtual computer) is doing. There are several ways of doing this. We are going to use a piece of software called "Bring".

On the virtual computer, log in as the stealth user. Now go to the official WinHoldem site (See {Bring}) and download "bring.exe" (from the "stealth" page). Download it directly into the Ninja folder.

The casino software can't see inside the Ninja folder, but it will be able to see Bring being run as a process. (If you want to see all the processes your computer is running, press CTRL+ALT+DEL and then click on the "processes" tab.)

The solution is to rename Bring to something more innocent. Then, if the casino software looks at the processes you are running, it won't be able to find out anything incriminating about Bring.

Rename Bring now. For example, you could rename it to "notepad" or “paint".

Step 12 - Install software on your real computer
You can now install Bring on your real computer. You don't need to rename it; the casino cannot see what is happening on your real computer.

Open a folder on the desktop, call it "Bring", and download into it the files "bring.exe" and “bring.ini".

Click "Start - Control Panel - Folder Options". Click the "View" tab. In the box labeled "Advanced Settings", scroll down and find the line "Hide extensions for known file types". If there is a green tick next to this line, click on it to make the tick disappear. Click "OK” to save your changes.

You should also install OpenHoldem on your real computer now if you haven't already. DO NOT INSTALL OH ON YOUR VIRTUAL COMPUTER.

Step 13 - Configure Bring
Do you remember the IP address of your virtual computer? You're going to need it again now.

On your real computer, go into the "Bring" folder you just created and open the file "bring.ini" in a text editor. (Right-click on “bring.ini”, choose "Open with..." and select "Notepad".)

Find the following lines:

```
# remotehost port password
--------------------------------
a 192.168.x.x ???? prompt
b 192.168.x.x ???? prompt
--------------------------------
```

Change them to this:

```
# remotehost port password
--------------------------------
a x.x.x.x eeee secret
--------------------------------
```

x.x.x.x should be the IP address of your virtual computer.

eeee is the "port" number. It can be a number of your own choice. 80 would be a good choice, and so would 800, 5000 or 8080.

"secret" is a password. It's not good password, but it is fine for the purposes of testing Bring.

Here are the changes I made to my bring.ini file:

```
# remotehost port password
--------------------------------
a 192.168.1.7 80 secret
--------------------------------
```

Step 14 - Start Bring
Now we're going to test Bring by playing a game of solitaire on the virtual computer, and by attempting to control it from the real computer.

Go to the virtual computer. You should still be logged in as the stealth user. Start the Solitaire programme ("Start - All Programs - Games - Solitaire").

Now open another command prompt (click "Start - Run", type "cmd" and press enter).

The command prompt window is currently looking at the "My Documents" folder. You need to move into the Ninja folder. If you followed the earlier suggestion, the location of your Ninja folder will be "C:\files". To move into this folder, simply type “CD C:\files" and press return. (If this doesn't work you will need to learn how to use the command prompt. Do an Internet search for "DOS command prompt".)

Now type the following command and press return. Instead of bring, you should type

the name of your renamed file. Instead of 80, you should type the port number you chose in the previous step, and instead of "secret" you should type the password you chose.

```
bring -s 80 secret
```

A small pop-up will appear to show you that Bring has started on your virtual computer. Click "OK".

Now go to your real computer. Go into the Bring folder and double click on "bring.exe".

Another pop-up appears to show you that Bring has started on your real computer. Click “OK”.

Another window appears labeled "Select remote connection". You should see a line that looks exactly like the line you entered in the "bring.ini" file.

```
a 192.168.1.7 80 secret
```

Click on this line so that it turns blue, and click the "OK" button.

Another window appears with a list of programs that are currently running on your virtual computer. One of them should be solitaire, so click on it so that it turns blue and then click the "OK" button. Solitaire will launch on your real computer, and if everything is set up correctly, you will be able to play it and watch your mouse clicks affecting both computers at once!

If you are unable to control solitaire from your real computer you definitely won't be able to control a poker table. Before you continue, seek help from someone who understands networks.

Step 15 - Stop running Bring
When you have had enough of playing solitaire, you should shut down Bring.

On the real computer, just close the solitaire window.

On the virtual computer, there are three ways to stop Bring.

1. Log off

2. Stop the Bring process. Press CTRL+ALT+DEL. Two task manager windows will appear, one for the real computer and one for the virtual computer. Find the task manager on the virtual computer and click the "Processes" tab, find “bring.exe" (however you renamed it), click on it to turn it blue, and then click on "End Process".

3. Lock the desktop. This doesn't actually close Bring, but it does halt it and prevent your real computer from communicating with it.

Stop Bring on both computers before you move on to the next step.

Step 16 - Set up the "run as a different user" facility
When you connect to the casino's Internet site you should always be logged into your virtual computer as the casino user. However, you will be running Bring AS IF you will logged on as the stealth user. In this way, the casino won't be able to detect that

you are using Bring. Here's how to do it.

On the virtual computer, log in as the casino user. Right-click on the desktop and choose "New", then choose "Shortcut". A window will appear. In the box, type "cmd". Click "Next" and then click "Finish".

A new icon, “cmd.exe", will have appeared on your desktop. It's actually a shortcut to the command prompt that you have used a few times before. Right-click on this shortcut and choose "Properties". A window appears and the "Shortcut" tab should already be showing. Click the "Advanced" button. Find the line which reads "Run with different credentials". Click on it so that a tick appears next to it. Click the "OK" buttons twice. "Run with different credentials" actually means "run this program as a different user".

Start solitaire again. Double click the “cmd.exe" shortcut and a window will appear, asking you which user you want to use to run the program. Click on the line "The following user". Then, enter the stealth user's name and password, and click “OK".

The command prompt window appears. It is running as if you were logged in as the stealth user. We can safely run Bring from this window. Go into the Ninja folder (perhaps by typing "CD C:\files”). Type the following command, changing the words "bring", "80" and "secret" exactly as you did in step 14.

```
bring -s 80 secret
```

Bring's pop-up window appears, so click the “OK” button.

Now go to the real computer and start Bring there. Try to open the solitaire window, just as you did before.

Go back to the virtual computer. Press CTRL+ALT+DEL to make the task manager appear in both windows. Find the virtual computer's task manager and select the "Properties" tab. Check that "bring.exe" (however you renamed it) is running under the stealth user, even though you are logged in as the casino user. Check that "sol.exe" - the solitaire process - is running under the casino user.

Step 17 - Check security with Gaze

We're going to make one more security check using the program Gaze. It can be downloaded from the forums at the WinHoldem official site. (http://forum.winholdem.net/wbb/viewtopic.php?t=1114)

Still on the virtual computer and still logged in as the casino user, go to the WinHoldem site. There should be two files - "gaze.cpp" and “gaze.exe". Download both of them to the virtual computer's desktop.

Double click on “gaze.exe” to run it. A window will appear, and shortly afterwards a pop-up labeled "Windows - no disk" will appear. Click the "Cancel" button, and close the Gaze window.

The results of the test have been saved to a file on the desktop called "gaze”. Confusingly, there are now two text files on the desktop with the same name. Double click one of them to open it in a text editor. If the first few lines don't look like this, you've got the wrong file.

```
Sat Jun 23 17:40:42 2007
```

```
==
PROCESSES
==

==
PROCESS NAME: [System Process]
--
WARNING: OpenProcess failed with error 87 (The parameter is incorrect)
process ID = 0x00000000
thread count = 1
parent process ID = 0x00000000
Priority Base = 0
```

You want to read the report on “bring.exe" (however you renamed it). Press CTRL+F to do a search for this file. When you find it, it should look like this...

```
==
PROCESS NAME: bring.exe
--
WARNING: OpenProcess failed with error 5 (Access is denied)
process ID = 0x00000EE8
thread count = 1
parent process ID = 0x00000EDC
Priority Base = 8
WARNING: CreateToolhelp32Snapshot (of modules) failed with error 5
(Access is denied)

THREAD ID = 0x00000EEC
base priority = 8
delta priority = 0
```

If you can see the message "OpenProcess failed with error 5 (Access is denied)", then you will know for sure that the casino user can't find out anything incriminating about Bring. Your security checks are now complete.

Close Notepad, solitaire and the “cmd.exe" shortcut. Delete Gaze from your desktop and empty the recycle bin. Terminate the Bring process. Reboot the virtual computer.

Step 18 - Install the casino software
You can now install the casino software. On the virtual computer, log in as the casino user. Create a new folder on the desktop called "poker". It's a good idea to install all your casino software in a single folder.

The casino user currently has restricted privileges. Some casinos insist that you install their software when logged in as a user with administrative privileges. Some casinos also insist that you run their software when logged in as a user with administrative privileges. Preferably, we would like to install and run the software while logged in as a user has restricted privileges.

Here is how to set up a typical casino's poker software.

Still logged in as the casino user, go to the casino web site. Download their software directly into the poker folder you created moments ago. Go into the folder and double click on the software. It will begin installing. When the install asks where you

would like to install the software, choose the "poker" folder you created moments ago. Don't let the install the use the default folder.

If the installation completes without complaints, all is well. However, if the installation complaints that it needs administrative privileges, this is what you must do:

Log out from the casino user's account and log in to the admin user's account. Open the "User Accounts" window just as you did before. Click on the casino user act and then click on "Change the account type". Click on "Computer Administrator" and then click the "Change Account Type" button. Log out from the admin user's account and log back into the casino user's account.

Try to install the software again. This time it should install without complaints. Now log out of the casino user's account and go back to the admin user account. Change the casino user's account type back to "limited". Log out of the admin user account and log back into the casino user's account.

Now try running the software. If the software complaints that it needs administrative privileges to run, change the casino user's account to "Computer Administrator" just as you did above. It's unfortunate that you have to run the software with administrative privileges, but it's not the end of the world.

Step 19 - Change your video settings
OpenHoldem is quite fussy about which desktop theme you are using. Currently, only the Windows XP default style and the Windows 2000 classic style are supported. You must change your desktop theme to one of these styles on both the real computer and the virtual computer. Click "Start - Control Panel - (Appearances and Themes) - Change the computer's theme" to do this.

Step 20 - Start playing poker
On the virtual computer, start Bring by using the “cmd.exe" shortcut. Start the casino software and choose the table. Make sure the table's window is not obscured by any other Windows.

Now go to your real computer, start Bring and try to get the table to appear in a window. If everything has been set up properly, you should be able to control the table from your real computer.

Now start OpenHoldem. Press OpenHoldem’s “Green Circle Button” button and you're ready to make money!

Step 21 - How to play in the future
Every time you use OpenHoldem to play poker for you, you must follow this simple checklist.

1. Boot up your real computer. Start OpenHoldem.
2. Boot up your virtual computer and log in as the casino user.
3. On the virtual computer, use the “cmd.exe” shortcut to run Bring.
4. Start the casino software. Select a table. Make sure this window isn't obscured by any other windows.
5. On the real computer, start Bring. Find the window with the poker table.

7.8 Rootkits

Rootkits are generally considered to be the subject of viruses, malware and other anti-social computer topics. (http://en.wikipedia.org/wiki/Rootkit) In the case of poker botting, however, we can use rootkits to our advantage. Specifically, since rootkits can hide such things as viruses, they can also hide OpenHoldem programs, registry entries, network ports (Bring traffic), files, etc.

This document will not give you specific steps for configuring and using a rootkit. That is up to you. However, a Google of “hacker defender” may or may not turn up some interesting results.

> *Note: Your anti-virus software will go nuts if you try to run a rootkit while your anti-virus software is operating. If you can configure your anti-virus software with a white list, then this may help the situation. Otherwise, you may need to uninstall your anti-virus software to use a rootkit.*

Automation

Alright, so you have made it through the previous sections, and have a working Table Map, a set of bot logic for your target game, and a fully-stealthed out configuration. You may have just started to realize what a pain it is to have to manually start the software, select a table, sit at the table, connect OpenHoldem, leave the table, start a new table, etc. This becomes even more oppressive if you want to run multiple simultaneous tables.

The answer is automation. Automation will do these opening, selecting, moving, and clicking activities for you, and allow you to ultimately sleep while your poker bot plays for you. Nothing better than making money while you sleep!

The general tool of choice for automation is AutoIt (http://autoitscript.com). AutoIt is a free (as in beer – this is not an open source program) Windows scripting language and execution engine. From the web site: "AutoIt is a freeware Windows automation language. It can be used to script most simple Windows-based tasks (great for PC rollouts or home automation). AutoIt has been in popular use since 1999 and continues to provide users and administrators with an easy way to script the Windows GUI. In February 2004 the latest version of AutoIt - known as AutoIt v3 - was released and added powerful scripting features."

Automation of poker botting is one of those "simple Windows-based tasks" that AutoIt is great for. The challenge is that now you have a new language to learn (does it ever end?), but the support for AutoIt on their forums is astounding, and you can also get help on the OpenHoldem forums.

As a starting point, the "Scout" hopper (hopper is the name generally given to AutoIt scripts that automatically start and connect poker windows for you) can be found here: http://scout-poker-bot-hopper.googlecode.com/files/Scout2.68.zip

Scout will not work out of the box, guaranteed, but it can serve as a possible starting point for your script for your casino. This is an important point – hoppers, like Table Maps, are very specific to a given casino. Think about the actions you take to join a table on different casinos – the actions to do that all differ ever so slightly!

Extensions

9.1 User DLL

A User DLL provides the ability to extend the functionality of OpenHoldem. More details on User DLLs can be found here: {User DLL}

9.2 Mouse DLL

The mouse DLL extension provides for a user-specified handling of mouse actions. The standard, reference mouse.dll that is shipped with OpenHoldem performs very un-human-like mouse activities. For example, instantly warping the mouse pointer to a location on the button and clicking is unlikely to be human behavior. A human is more likely to start the mouse moving, accelerate to the target, then decelerate as the pointer nears the position of the button. Simulating mouse movements that are more human-like may help to avoid detection by casions.

The mouse DLL extension is available if you would like to implement custom mouse action behavior. The source code for the reference mouse.dll implementation can be found on Google Code in the source code repository.

Two methods must be exposed in your mouse.dll in order to interface properly with OpenHoldem.

“MouseClick” is called when OpenHoldem wants to click a mouse button on a specific location on the screen. The prototype for the “MouseClick” method is:

```
MOUSEDLL_API int MouseClick(const HWND hwnd, const RECT rect,
const MouseButton button, const int clicks, const HWND
restore_focus, const POINT restore_cursor)
```

“MouseClickDrag” is called when OpenHoldem wants to click a mouse button at a specific location on the screen, hold it, drag the cursor to another location, and then release it. Dragging is from rect.left to rect.right, halfway between rect.top and rect.bottom. The prototype for the “MouseClickDrag” method is:

```
MOUSEDLL_API int MouseClickDrag(const HWND hwnd, const RECT rect,
const HWND restore_focus, const POINT restore_cursor)
```

Parameters:

- hwnd (in) – the HWND of the window to take the mouse action on
- rect (in) – the bounding rectangle of the area to take action on – this generally corresponds to a Table Map Region record rectangle, like those that define button click areas; this rectangle is in relative client context
- button (in) – the mouse button to be clicked, this is an enum: “enum MouseButton { MouseLeft, MouseMiddle, MouseRight }”
- clicks (in) – the number of clicks of the specified button
- restore_focus (in) – the HWND of the window to return focus, active and foreground to, after the action has been taken; if this is NULL, no focus resotre is requested
- restore_cursor (in) – a POINT to return the cursor to, after the action has been taken; if POINT.x and POINT.y are set to -1, then no cursor return is requested

9.3 Keyboard DLL

The keyboard DLL extension provides for a user-specified handling of keyboard actions. The standard, reference keybaord.dll that is shipped with OpenHoldem performs very un-human-like keyboard actions. For example, there is no delay between keystrokes. When SWAGing, a human is likely to have somewhat random delays between keypresses as the SWAG amount is entered into the SWAG box. Simulating keyboard actions that are more human-like may help to avoid detection by casions.

The keyboard DLL extension is available if you would like to implement custom keyboard action behavior. The source code for the reference keyboard.dll implementation can be found on Google Code in the source code repository.

Two methods must be exposed in your keyboard.dll in order to interface properly with OpenHoldem.

"SendString" is called when OpenHoldem wants to send a string to the poker client. Generally, this method is expected to click in the specified rectangle first, to activate it. The prototype for the "SendString" method is:

```
KEYBOARDDLL_API int SendString(const HWND hwnd, const RECT rect,
const CString s, const bool use_comma, const HWND restore_focus,
const POINT restore_cursor)
```

"SendKey" is called when OpenHoldem wants to send a single key to the poker client. Generally, this method is expected to click in the specified rectangle first, to activate it. The prototype for the "SendKey" method is:

```
KEYBOARDDLL_API int SendKey(const HWND hwnd, const RECT rect,
UINT vkey, const HWND restore_focus, const POINT restore_cursor);
```

Parameters:

- hwnd (in) – the HWND of the window to take the mouse action on
- rect (in) – the bounding rectangle of the area to click in to activate – this generally corresponds to a Table Map Region record rectangle, like the chat box or the SWAG field; this rectangle is in relative client context; if the members of the RECT struct are {-1,-1,-1,-1} then do not click in the rectangle first
- s (in) – the string to be sent to the poker client
- use_comma (in) – if true, then use a "comma" instead of a "dot" as the decimal separator
- vkey (in) – the VKEY code of the key to send to the poker client
- restore_focus (in) – the HWND of the window to return focus, active and foreground to, after the action has been taken; if this is NULL, no focus resotre is requested
- restore_cursor (in) – a POINT to return the cursor to, after the action has been taken; if POINT.x and POINT.y are set to -1, then no cursor return is requested

9.4 Scraper Override DLL

The scraper override DLL extension provides for a user-specified override of the output of the screen scraper engine. Sometimes, better game state information can be retrieved from poker client controls that are not scrapable by the pixel-driven scraper engine in OpenHoldem, such as by parsing the chat box. Parsing the chat box would require direct memory reading of the poker client, or DLL injection. The

approach for such a chat box parsing is beyond the scope of this document, but a good discussion on the topic can be found here:
http://www.codingthewheel.com/archives/how-i-built-a-working-online-poker-bot-7

Immediately after the screen scrape is complete, OpenHoldem will pass a pointer to a structure containing the results of the scrape to your override DLL. This is not a const pointer reference, so any of the members of the struct can be changed directly. The prototype for the method called by OpenHoldem is quite straightforward:

```
SCRAPERDLL_API void OverrideScraper(SScraperState *state)
```

No casting required. The members of the SScraperState struct are as follows:

```
struct SScraperState
{
	char			title[512];
	unsigned int		card_common[5];
	unsigned int		card_player[10][2];
	unsigned int		card_player_for_display[2];
	bool			dealer[10];
	bool			sitting_out[10];
	CString			seated[10];
	CString			active[10];
	CString			name[10];
	double			balance[10];
	bool			name_good_scrape[10];
	bool			balance_good_scrape[10];
	double			bet[10];
	double			pot[10];
	CString			button_state[10];
	CString			i86X_button_state[10];
	CString			i86_button_state;
	CString			button_label[10];
	double			sblind;
	double			bblind;
	double			bbet;
	double			ante;
	LimitType		limit;
	double			handnumber;
	bool			istournament;
};
```

Any of these can be changed like so (which will give you four-of-a-kind Aces)

```
SCRAPERDLL_API void OverrideScraper(SScraperState *state)
{
	state->card_common[0] = StdDeck_MAKE_CARD(StdDeck_Rank_ACE,
StdDeck_Suit_HEARTS);
	state->card_common[1] = StdDeck_MAKE_CARD(StdDeck_Rank_ACE,
StdDeck_Suit_DIAMONDS);
	state->card_common[2] = StdDeck_MAKE_CARD(StdDeck_Rank_ACE,
StdDeck_Suit_SPADES);
	state->card_common[3] = StdDeck_MAKE_CARD(StdDeck_Rank_ACE,
StdDeck_Suit_CLUBS);
}
```

And on return, OpenHoldem will instantly use the changed values instead of the scraped values.

Appendix A – OpenScrape Font collection example

This example on collecting fonts was reproduced from here:
http://www.maxinmontreal.com/wiki/index.php5?title=Collecting_text

We start with nothing except a table that we need to extract fonts from, here I am using a Notepad doc for simplicity.

Figure 9.4-67 Font collection example - notepad

Next we fire up OpenScrape, and attach it to our "Table". Then we build a region around the area we are working on. "New" / "Region" / name it characters / Use the coordinate editor to enclose the Region. You can also "shift-drag" the rectangle around to place it.

Figure 9.4-68 Font collection example - region

Set the region to the correct "transform", Type 0 in this case. Use the eyedropper to highlight the foreground pixels until we have nice seperation in the "Pixel seperation" field. Press "Create Font". We are presented with the "Add font characters" window.

Highlight each "?" in turn and enter the correct character. "OK" to save. "Sort" when done here, then "save" on main GUI.

If you hit the problem of openscrape seeing two letters as one because they have touching pixels then you will need to collect them seperately. Make a new temporary region that only encloses one character (call it char), capture the problem characters individualy and all should be good.

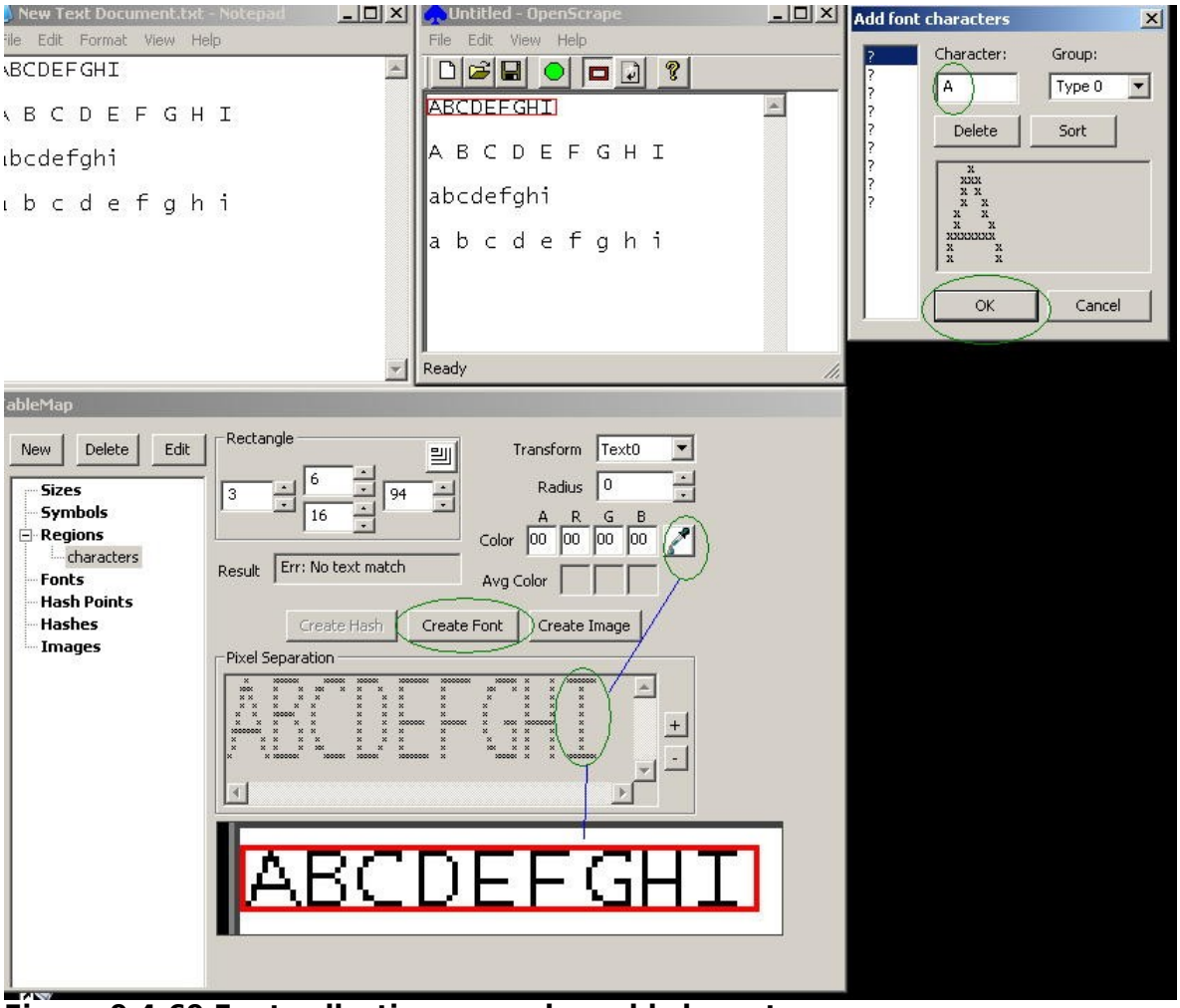


Figure 9.4-69 Font collection example - add characters

Here is the result after collecting only "A", you can see this by clicking inside the "characters" Region.

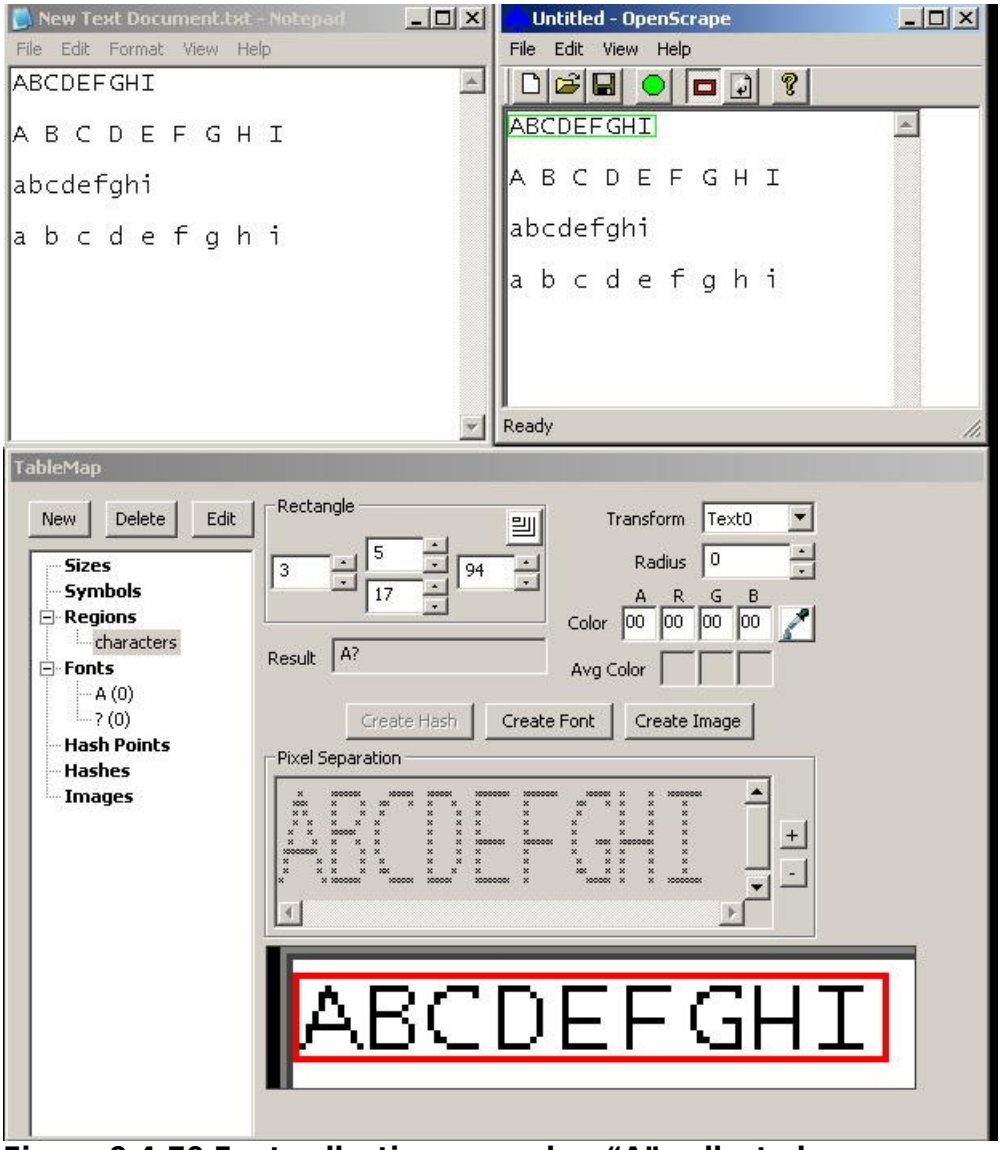


Figure 9.4-70 Font collection example – “A” collected

Note "A,B,C" are already present in the Table Map.

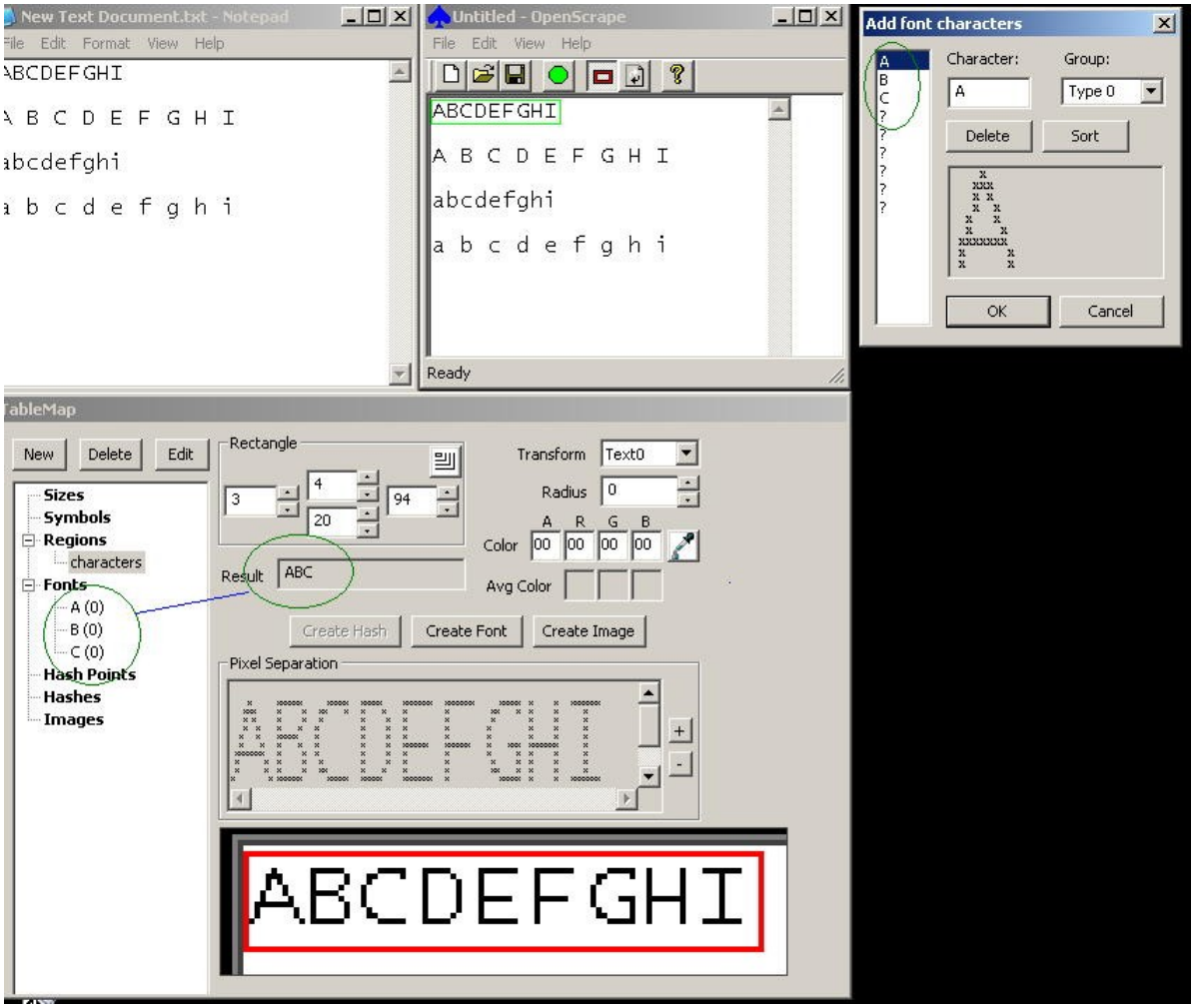


Figure 9.4-71 Font collection example - "ABC" collected

So on we go collecting fonts, here we collect "D" .

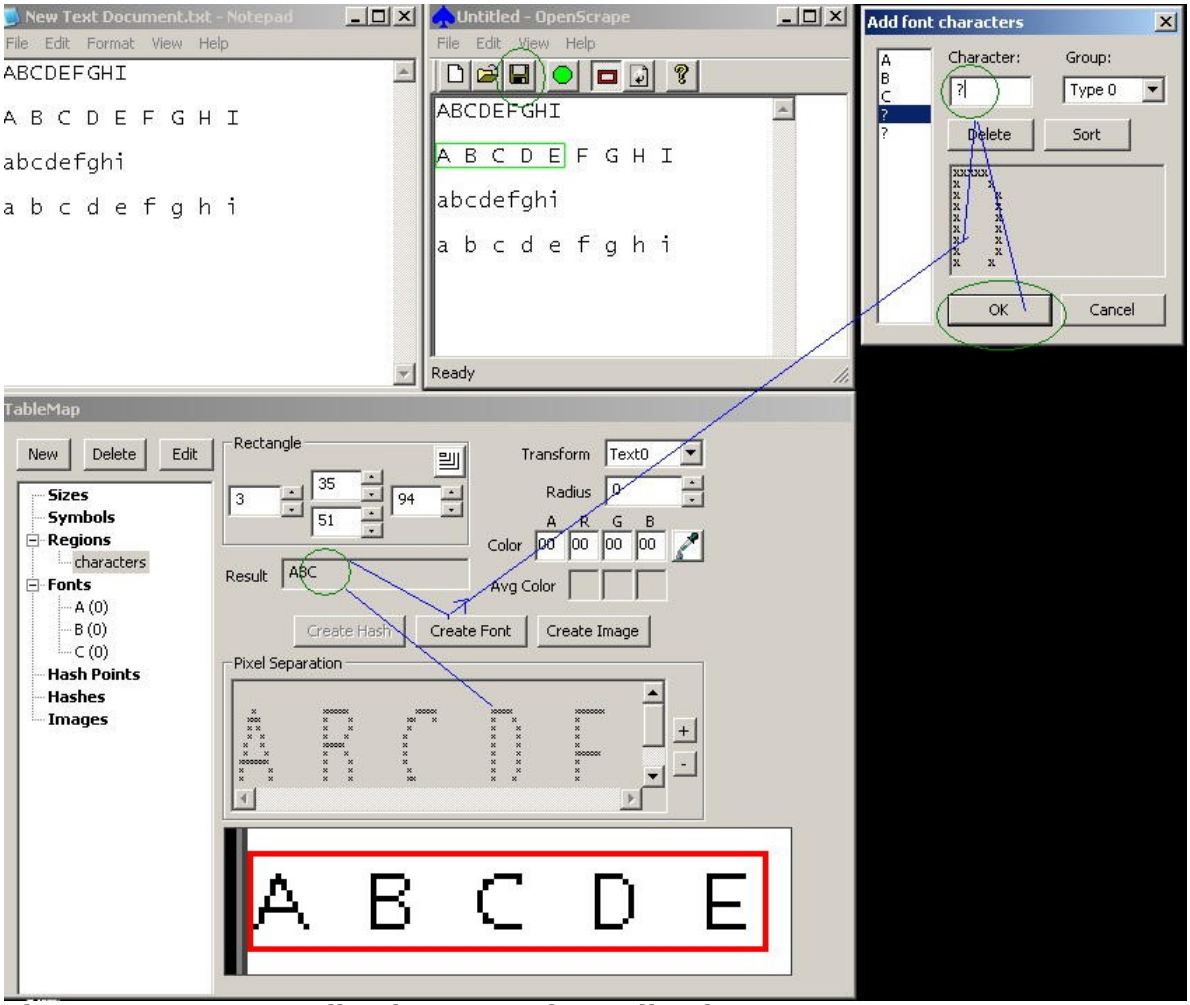


Figure 9.4-72 Font collection example - collecting "D"

Appendix B – Future TODOs

- Move Edit/Preferences/Use Comma or Dot setting to the Table Map.