# StealthBot

Developer Guide

# CONFIDENTIAL

Authored September 24, 2007
Revision: 2007.09.24 (1)
Andy Trevino / Stealth@USEast / andytrevino@gmail.com / andy@stealthbot.net

# Table of Contents

## Introduction

So, it's finally time to let other people into the crazy, mixed-up world that is StealthBot's development. It's been a wonderful pet project for almost five years now, but the inclusion of more people writing code is long overdue. I have always been exceptionally impressed by the dedication of many members of the community, in particular those of you selected to be developers and those considered for future positions as developers, and I cannot thank you enough.

You've been chosen as a StealthBot developer, which means I really like you, and you've shown quality code-writing and community skills. Welcome to the team. I won't waste any more words.

## Things to Note about StealthBot's Code

When taking in StealthBot's code, you'll notice some aspects of it really truly suck. The command system is a 1,200+-line IF statement. (There are some merits to this, by the way.) There is code (particularly in the Tic-Tac-Toe form) that dates back to 2003, sloppy and sometimes ridiculous, but it all for the most part works. Short of blowing the project away and starting over again, this is just kind of the way things are.

I have stuck to a few good tenets of code design and software engineering throughout the bot's development, ingrained in me by my sophomore Intro to Programming teacher Aaron Pavao. The aspect I'm most proud of is the consistent, effective tabbing of every flow control statement in the program. You will be hard-pressed to find an IF statement or FOR loop that's not properly tabbed-out and clearly-readable.

Comments are another thing, which I address in a later section in more detail; as such, I commented some revisions but not others, some code but not other code, et cetera. You will find that comments, when not containing blocks of unused code, will at least provide minor insight into my thought process at the time of construction.

I mostly know my way innately around StealthBot's code, so at times things aren't where they would logically be, and that's what I'd like to address next.

## Code Object Contents and Description

The following is a table of all the modules in StealthBot, and a brief description of what they do, as of this document's last date of revision.

## *Table of StealthBot Forms*

| FORM NAME | DESCRIPTION | COMMENTS |
|---|---|---|
| **frmAbout** | Contains the program's About dialog and related code. | Some very old code lies here… |
| **frmCatch** | Catch Phrase Editor UI | |
| **frmCCEditor** | Custom Command Editor UI | |
| **frmChat** | Main form of the program. | Contains major RichTextBox utility functions like AddChat, and many objects with events like the BNCS/BNLS connection classes are anchored on this form. |
| **frmClanInvite** | Clan invitation accept/decline dialog | |
| **frmCustomInputBox** | Used for Step-by-Step Config | Initially designed to be adaptable to other situations requiring custom input. It still could be, but at this time is only used for step-by-step config. (Not very well-engineered.) |
| **frmEMailReg** | E-mail registration dialog | |
| **frmFilters** | In/outbound chat filter UI | |
| **frmManageKeys** | CD-key manager | Could be easily duplicated or adapted to allow an expansion key manager. |
| **frmProfile** | Profile viewer | See frmWriteProfile for the profile editor (these could be merged into one form.) |
| **frmProfileManager** | Unused bot config profile manager UI | Not finished! |
| **frmQuickChannel** | Quick channel editor UI | |
| **frmRealm** | Realm logon UI | |
| **frmSettings** | Settings UI | Contains some of what I think is my most beautiful code in the code that draws the TreeView. Note the comment at the top if you want to make UI control changes to the settings panel. |
| **frmSplash** | Splash screen | |
| **frmTTT** | Tic Tac Toe | This was once a major StealthBot selling point. "It has Tic Tac Toe!" Hah. |
| **frmUserManager** | Userlist manager UI | |
| **frmWhisperWindow** | The Individual Whisper Window UI | More-IM-style conversation was the goal with these. |
| **frmWriteProfile** | Editable profile UI | |

## *Table of StealthBot Modules*

| MODULE NAME | DESCRIPTION | COMMENTS |
|---|---|---|
| **BNCSutil** | BNCSutil GPL library | |
| **modAPI** | Consolidated API methods used by StealthBot code | I tried to bring all the Windows API used anywhere together in this module, to the point of breaking the code-portability tenet of Object-Oriented Design. |
| **modBNLSCode** | Contains some utilities for connecting to BNLS | |
| **modColors** | Code related to the Color List files (.sclf) for SB skins | |
| **modCommandCode** | Command-related code | Includes several KEY functions like ProcessCommand(), ValidateAccess() and Commands(). Commands() works on an authenticate-first basis, calling ValidateAccess() with the speaker's credentials and desired command to authenticate the user, then letting them move into the command code IF if they're OK. **If you are adding a command, you MUST update ValidateAccess() to return something for that command, or your command will be ignored.** |
| **modCRC32Checksum** | CRC32 checksum calculation code | |
| **modEnum** | Consolidation of custom Types used in the program | |
| **modEvents** | Contains all the event handlers | |
| **modGlobals** | Contains all global-scope variables used within the program | |
| **modITunes** | Contains ITunes handler code | |
| **modLogging** | Contains unused code for the HTML logging system | |
| **modMail** | Handlers for the botmail system | |
| **modMenu** | Code for the dynamic scripting menus | Very cool in my opinion! I had to do a decent amount of tweaking to adapt this. |
| **modNews** | Handles bot news | |
| **modOSversion** | Retrieves OS version information | |

| | | |
|---|---|---|
| **modOutboundPackets** | Contains several outbound packet crafting methods, like Send0x50() | |
| **modParsing** | BNCS packet and statstring parsing code | |
| **modProcessPriority** | Allows SB to run at high priority during efp | |
| **modQueueCode** | Unused experimental queue code | |
| **modQuickChannels** | Quick channel loading/saving code | |
| **modSafelist** | Safelist loading/saving code | |
| **modSubclassing** | Subclassed handlers for things like the system tray callback | |
| **modSystray** | Constants etc related to the systray icon | |
| **modTimerProcs** | Timer procedure callback functions | |
| **modURLDetection** | Handles URL detection on the main RTB | |
| **modUTF8** | Handles UTF-8 enc/dec | |
| **modWar3Clan** | Warcraft III clan methods | |
| **modWhisperWindows** | IWW handler code | Moderately-cool |
| **modWinampControl** | Winamp control code | Also moderately-cool |
| **modWinampTitle** | Ported code to get Winamp's title | Publicly-available and thus very well-commented |
| **UnsignedConversions** | VB unsigned conversion module | |

## *Table of StealthBot Class Objects*

| CLASS NAME | DESCRIPTION | COMMENTS |
|---|---|---|
| **cBNCSRecvBuffer** | BNCS packet receive buffer | Was once the source of many problems |
| **cBNLS** | BNLS outbound packet methods | Ignore the comments at the top. Networkz wrote nothing but the method signatures. |

| | | |
|---|---|---|
| **cBNLSRecvBuffer** | BNCS packet receive buffer | |
| **clsBitwiseOperator** | Bitwise operator methods | Written by sneakcharm (David Fritts) – very nice code |
| **clsBotVars** | Stores important settings | |
| **clsClanPacketHandler** | Handles Warcraft III clan packets | |
| **clsCRC32** | Performs CRC32 calculations | May not be used anymore. Also by sneakcharm. |
| **clsErrorHandler** | A better error handler | Adaptable to provide more-detailed error messages for foggy Winsock errors |
| **clsFriend** | Friend object | |
| **clsFriendlistHandler** | "Packet friend list" handler | |
| **clsMCPHandler** | Realm packet handler | |
| **clsPacketDebuffer** | Packet debuffer class used in many places | |
| **clsQueueObj** | Queue container object | Created when the queue was converted from a dynamic array of strings to a Collection |
| **clsSafelistEntry** | Safelist entry container object | |
| **clsUserInfo** | User container object | The main in-channel user tracking collection uses these |
| **CToolTip** | Tooltip drawing code | I don't remember where it's from, but it's publicly-available |
| **DataStorage** | Holds connection-specific data | Used only during connections |
| **PacketBuffer** | Outbound packet buffer | |
| **ScriptSupportClass** | Exposed to scripting | Publicly-available, minus the bulk of the addchat code |

## Naming Conventions

I've tried to maintain consistent conventions throughout StealthBot's development. Obviously this wasn't completely possible, but you should be able to always get the gist of a variable's purpose by its name. I didn't do anything stupid like naming variables after girls or rock bands, both real horror stories courtesy Mr. Pavao.

As long as you all continue the tradition of logical naming, you'll help keep things in order in the wonderful world of StealthBot.

### *Variables*

Variables are named with their purpose. I don't use any sort of special notation except occasionally a single letter depicting their type. Single-letter variables are temp variables, and the letter indicates their type: i is an int, l is a long, c and n are int or long counters, s and s2 are strings, and so on. You will not find meaningful global variables with single-letter names.

### *Code Objects*

Code objects follow strict naming conventions, the logic behind this being that they aren't created very often and as such can have good identifiers on them. If I wrote it, it has 3 lowercase letters denoting the type of object (frm, mod, cls) and then the name of the class in proper case. A few class objects begin with a single lowercase C which is also OK in my book.

### *UI Objects*

UI objects follow the strictest naming conventions. The vast majority of the time, and certainly in forms with multiple objects, these objects will be named with three lowercase letters denoting their object type (cbo is a combobox, txt is a text box, fra is a frame, lbl is a label and so on) followed by a descriptive name in proper case. I would ask that you adhere to this when creating UI controls that have any sort of reference in code.

## Comment Conventions

StealthBot was not really coded with an eye towards group development. Now that group development is taking place, I ask that everyone participating writes a comment to identify who writes particular hunks of code. For example:

```
' This method checks whether or not a user is authenticated to use
'   their desired command. -andy
```

should be considered an acceptable comment to start a method you're writing.

## Compiler Flags

The bot has three compiler flags built in that you will find useful. You can set these flags in the Make tab of Project Properties.

**BETA = 1** will tell VB to compile with the beta authentication system.

**COMPILE_DEBUG = 1** will run the bot with subclassing turned off. This is important because if subclassing is ON and the bot encounters an error, the **entire VB IDE will crash.** Use when debugging at all times -- it will also prevent any subclassing-dependent stuff, like minimization to the system tray, from screwing you over.

**COMPILE_RELEASE = 1** will enable the CRC32 hash checking for a release version of StealthBot. Be sure to use ApplyCRC32 in docs to apply a CRC32 hash to the end of the program binary before distributing.

## Final Words

Welcome to StealthBot, guys, and thank you for everything you've done and will do.