# DiceBot v4 Programmer Mode Documentation

# Process

Like previous versions, DiceBot 4s programmer mode runs in a loop. When the users starts the bot, the bot executes the Reset function (detailed below) for the applicable game. The NextBet parameter from the reset function will be used as the first bet. Once the bet result for the first bet is received from the site, the bot will do its internal processing (like logging it to the database, sending it to the GUI etc). Once all internal processing is completed, the global variables for the programmer mode is updated (this includes the stats, sitedetails and sitestats objects). The DoDiceBet function is then called using three paramters:

* PreviousBet: result of the previous bet, includes betamount, profit, roll, betid. Full UML for class available in this document.
* Win: a boolean called that indicates whether the previous bet is seen as a win internally or not,
* NextBet: An Instance of the PlaceDiceBet class, with the amount, chance and high values copied from the previous bet. This object is used to place the next bet, so set the properties of this class to manipulate the next bet.

# Required functions

For a script to be valid, it MUST implement the following functions:

* Void DoDiceBet(DiceBet PreviousBet, bool Win, PlaceDiceBet NextBet)
* ResetDice(PlaceDiceBet NextBet)

DoDiceBet is the function that handles the core of your logic. It gets called when a bet result has been received and the next bet needs to be calculated. The NextBet parameter is used to send the bet to the site, and must thus be used to specify the bet details needed.

ResetDice is called when the script starts and the result is used for the first bet. It can be used in DoDiceBet to reset your script easily, and will be called from DiceBot if internal triggers or error handling is used and set to reset. It is recommended that any variables that might influence the functionality of your script be reset to their definition/instantiation values in this function. The NextBet parameter is used to send the bet to the site, and must thus be used to specify the bet details needed.

# Optional Functions

DiceBot has some optional functions that might be called from DiceBot, and can be used in the script to extend functionality. These are not yet implemented but will include custom error handling for errors received from the site or while betting.

# Available variables

Unlike version 3 of DiceBot, the amount of internal variables has been reduced, but the amount of information available to the script has been greatly increased. There are now only 4 global values defined in DiceBots programmer mode

* Stats: Instance of SessionStats, provides statistics of the current betting session, including wins, losses, profit,streak profits etc.
* SiteDetails: Instance of SiteDetails, Provides details of the site being used
* SiteStatsL Instance of SiteStats, provides site level stats of the current user including amount of bets at the site, wins and losses at the site, wagered, profit and balance.
* Balance: Users balance for the currency being used at the site being used.

These variables are updated after each bet and are read only. While the programmer mode will technically allow a script to assign a value to any of these variables and their properties, the changes will be overwritten with the next bet. The instances received in the programmer mode are disposable copies of the instances used inside of the bot and is disposed of soon after the result of the next bet is received.

# Available functions

There are internal functions available for the programmer modes, to access features of the bot and sites being used.

* Void Withdraw(string Address, decimal Amount)
* Void Invest(decimal Amount)
* Void Tip(string User, decimal Amount)
* Void ResetSeed()
* Void Print(string Message)
* Void RunSim(decimal StartingBalance, long NumberOfBets)
* Void ResetStats()
* Object Read(string Prompt, int ReturnType)
* Object Readadv(string Prompt, string AccetText, string CancelText, int ReturnType)
* Void Alarm()
* Void Ching()
* Void ResetBuiltIn()
* Void ExportSim(string FileName)