ReplayEngine API

Table of Contents

[1. ReplayEngine Interface Properties 3](#_Toc214684015)

[2. ReplayEngine Interface Methods 4](#_Toc214684016)

[3. Usage 5](#_Toc214684017)

# ReplayEngine Interface Properties

The properties for the ReplayEngine interface are used by the ReplayEngine abstract class to determine which action the current line contains. Currently only a get is defined. There are no planned uses as of this release for the properties.

* string GameIdTag
* string TournamentIdTag
* string TableIdTag
* string HoleCardsTag
* string FlopTag
* string TurnTag
* string RiverTag
* string ShowdownTag
* string SummaryTag
* string FinalBoardTag
* string GameHeaderTag
* string WonPotAmountTag
* string FinalPotAmountTag

# ReplayEngine Interface Methods

The following are the methods defined in the interface. Most take a string and parse it returning a string representing the method call. For example, GetGameId takes a string and parses it to find the GameId. However, it is not known whether all games site use a numeric for their id’s so a string is returned. Additional processing can be done if necessary or desired. The methods, SetHoleCards, SetFlop, SetTurn and SetRiver take an IHandHistoryDictionary and a user name. These methods parse the to find card information for the hole, flop, turn and river cards. It’s possible that game sites print different information so not all methods need to actually do work. However, for the current release most methods are only used within the ReplayEngine implementing class.

The GetHandReplay method is the most important method to implement. This is used in the HandHistoryController class to iterate through the hand history found. The username parameter is the user that is logged in. At the time of this release only PokerStars had a concrete workable implementation of IReplayEngine. The username is used to find winning hands as well as amounts won for display in the statistics tab of the HandHistory section.

* Collection<IHandHistoryDictionary> GetHandReplay( string fileName, string userName );
* ReplayAction GetReplayAction( string lineToCheck );
* string GetGameId( string lineInfo );
* string GetTournamentId( string lineInfo );
* string GetTableId( string lineInfo );
* string GetGameDescription( string lineInfo );
* string GetDateTime( string lineInfo );
* void SetHoleCards( IHandHistoryDictionary history, string lineInfo );
* void SetFlop( IHandHistoryDictionary history, string lineInfo );
* void SetTurn( IHandHistoryDictionary history, string lineInfo );
* void SetRiver( IHandHistoryDictionary history, string lineInfo );
* string GetShowdown( string lineInfo );
* string GetSummary( string lineInfo );
* string GetFinalBoard( string lineInfo );
* string GetGameHeader( string lineInfo );
* string GetWonPotAmount( string userName, string lineInfo );
* string GetTotalPotAmount( string lineInfo );

# Usage

In order to make your engine compatible with the Texas Holdem Calculator you must implement the IReplayEngine interface. This is simply done my importing the Holdem.Interfaces dll to your project references. If you wish to use the default abstract class ReplayEngine, import the Holdem.ReplayEngine dll and use the following class signature:

public class MyReplayEngine : ReplayEngine

The ReplayEngine class defines the GetReplayAction class which essentially takes a line and finds out which action it contains. You can use this action to define how your file is parsed and when to call any methods or properties.

After creating, debugging and testing your ReplayEngine you can simply drop the assembly in: “\Program Files\Texas Holdem Calculator\ReplayEngineAssemblies” or you can add the full path to your assembly as part of the “HandHistoryReplayEngineAsmDirectory” configuration in the Holdem.Configuration.dll.config, which is located in the same directory as the application. The following is an example of the configuration value with two distinct paths added; note the “;” which may also be a “,” if you chose:

value="\Program Files\Texas Holdem Calculator\ReplayEngineAssemblies;\MyConfigKey\SubDirectory\AnotherSub"

You can add as many directories as you wish and the application will search through each one for any types that implement IReplayEngine. Note, however, this is only done on load so any assemblies you add after the application is running will not be picked up until you restart.