SMITE API / Paladins /Realm API Developer Guide

TABLE OF CONTENTS

DOCUMENT CHANGE HISTORY

DATE	*MODIFIED BY*	*CHANGE DESCRIPTION*
2013-12-12	AP/JC	Initial Version
2014-05-22	AP	Added method "**getmatchidsbyqueue"** (Request: Cyber)
2014-07-07	AP	Added method "**getgodrecommendeditems"**
2014-07-10	AP	Added section for API Access Limits
2014-07-22	AP	Added brief description of data returned by each API.
2014-07-31	AP	Updates to "**getmatchidsbyqueue";** added {hour} parameter
2014-09-04	AP	Added method "**getesportsproleaguedetails"**
2015-02-11	AP	Added methods "**getplayerstatus**" and "**getmatchplayerdetails**"
2015-03-04	AP	Removed method "**getmatchplayerdetails**"
2015-04-03	AP	Re-added method "**getmatchplayerdetails**"
2015-05-20	AP	Added queue for MOTD (465)
2015-09-18	GT	Deprecated "**getteammatchhistory**"
2015-09-22	GT	Added Endpoint, Xbox, and Privacy Sections, "**getplayerachievements**" method
2015-12-07	AP	Added clash (466) and clash challenge (467) as match queue values.
2016-02-11	GT	Added new Ranked Joust (3v3) queue (450) and Ranked Duel Ranked Conquest naming.
2016-04-20	GT	Added PS4 Endpoint
2016-04-26	GT	Added documentation for existing "**getgodskins**" method
2016-09-01	GT	Added documentation for new "**getpatchinfo**" method
2017-04-19	AP	Added method "**getplayerloadouts"** (PALADINS only)
2017-04-27	AP	Added method "**getgodleaderboard"** (SMITE only); Added method "**gethirezserverstatus"**

2017-07-14	AP	Formally documented existing method “**getmatchdetailsbatch**”.
2018-05-21	AP	Indicated that method **searchTeams** applies to SMITE only.
2018-06-04	AP	Indicated the method **getFriends** applies to PC only.
2018-07-03	AP	Added method **getPlayerIDInfoForXboxAndSwitch**. This is meaningful only for the Paladins XBOX API.
2018-10-08	AP	Added method “**getchampionleaderboard”** (PALADINS only);

GETTING STARTED

Introduction

The purpose of this document is to provide SMITE API Developers and PALADINS API Developers with the necessary information to access and utilize the API methods. The JSON data returned from the API methods is designed to be "self-documenting" and therefore detailed descriptions of each data field are not included in this API Developer Guide.

Registration

To register to become developer, go here to register. If your application is accepted you will receive custom credentials to access the API.

Credentials

To access the APIs you'll need your own set of credentials which consist of a developer id (devId) and an authentication key (authKey). The process of obtaining credentials should be a one-time activity.

Here are the credentials for a sample account:

- DevId: (eg, 1004)
- AuthKey: (eg, 23DF3C7E9BD14D84BF892AD206B6755C)

Use your personal credentials to access the api via a *Representational State Transfer* (REST) web service hosted at api.smitegame.com. Note that the same DevId/AuthKey combination should work for both SmiteAPI and PaladinsAPI, across all supported platforms.

Sessions

To begin using the API, you will first need to establish a valid Session. To do so you will start a session (via the **createsession** method) and receive a SessionId. Sessions are used for authentication, security, monitoring, and throttling. Once you obtain a SessionId, you will pass it to other methods for authentication. Each session only lasts for 15 minutes

and must be recreated afterward.

More details regarding Session creation are provided later in this document.

API Access Limits

To throttle API Developer access various limits have been setup to prevent over use of the API (either intentional, more likely unintentional "over use").

Here are the default initial limitations for API Developers:

concurrent_sessions: 50
sessions_per_day: 500
session_time_limit: 15 minutes
request_day_limit: 7500

Platform Endpoint Base URLs

As of version 2.11, the SmiteAPI and Paladins API supports 3 platforms: PC, Xbox & PS4. Each platform is supported by a separate endpoint. Applications access each endpoint via a unique base URL:

- SMITE PC: http://api.smitegame.com/smiteapi.svc
- SMITE Xbox: http://api.xbox.smitegame.com/smiteapi.svc
- SMITE PS4: http://api.ps4.smitegame.com/smiteapi.svc
- Paladins PC: http://api.paladins.com/paladinsapi.svc
- Paladins Xbox: http://api.xbox.paladins.com/paladinsapi.svc
- Paladins PS4: http://api.ps4.paladins.com/paladinsapi.svc

The interface for each endpoint is identical. Therefore, a single application code-base should be able to support all supported platforms.

However, since each endpoint is serviced by a unique Web Service, applications must create and maintain separate Sessions for each endpoint. In other words, you cannot use a Session ID created by the SmiteAPI for PC endpoint (via the /createsession method) to make method calls to the SmiteAPI for Xbox endpoint.

Note that, depending on the Hi-Rez development pipeline and platform certification requirements, some SmiteAPI features will not be immediately available in all platforms. For instance, if Xbox does not currently support Clans, then clan/team-related methods should return empty datasets or an error

response.

SmiteAPI for Xbox Special Considerations

As stated earlier, the SmiteAPI for Xbox endpoint supports the same API interface as does SmiteAPI for PC. However, supporting applications must be aware of the following differences that might lead to unexpected responses:

1. Players of SMITE for Xbox are identified by Xbox Live "Gamertags" instead of Smite player names. Therefore, when searching for player information, you should specify Gamertag as a parameter where you would normally specify playername. By the same token, API method results will return Xbox Live Gamertag as "playername", as is expected in SmiteAPI for PC.
2. Xbox Live Gamertags are owned and administered by Microsoft, outside the purview of Hirez Studios. We cache current player GamerTag each time a player logs into SMITE for Xbox and invalidate any other players that might have the same cached GamerTag. In this way, we hope to keep GamerTag searches as current as possible. However, it is possible that stored GamerTag names become "stale" when players change GamerTag through Microsoft, especially if said player does not log into SMITE for Xbox often. By the same token, if an abandoned GamerTag is transferred to another Xbox Live player, it is possible that a SmiteAPI query could return data for the wrong player. Hopefully, this is not a common occurrence, but we wanted to notify you of the possibility. Note that SMITE playerId is guaranteed unique. Therefore, if your application already stores SMITE playerId of customers and specifies that value instead of GamerTag, you eliminate the possibility of name collisions.
3. Caching of GamerTags has only been implemented fairly recently. Therefore, searches of players by GamerTag who have not logged into SMITE for Xbox since caching was implemented will obviously fail.

SMITE Player Privacy Option

SMITE version 2.14 has introduced the option of player opt-in privacy. Players choosing to enable privacy for their account will affect data returned by SmiteAPI for said players:

1. Player queries on individual players marked “Private” will fail. In particular, query methods should return empty dataset, just as if the player did not exist. For example, a /getfriends call on the player “JohnDoe” would return an empty dataset if JohnDoe was marked Private.
2. Methods that return data on multiple players may still return some data for Private players, while other data is obfuscated. In particular, for private players:
    a. Player “Name”: returned as “” (empty string)
    b. PlayerId:: returned as 0 (int value zero) or “0” (string), depending on method
    c. ClanId: returned as 0 (int value zero)

As an example for item #2 above, the /getmatchdetails method for a specified MatchId would return an object for each player in the Match. However, the returned object for each Private player would include a “playerName” of “” and a “playerId” of “0”.

API METHODS & PARAMETERS

APIs - Connectivity

/**ping**[ResponseFormat]
A quick way of validating access to the Hi-Rez API.

/**createsession**[ResponseFormat]/{developerId}/{signature}/{timestamp}
A required step to Authenticate the developerId/signature for further API use.

/**testsession**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}
A means of validating that a session is established.

/gethirezserverstatus[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}
Function returns UP/DOWN status for the primary game/platform environments. Data is cached once a minute.

APIs

/**getdataused**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}
Returns API Developer daily usage limits and the current status against those limits.

/**getdemodetails**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{match_id}
Returns information regarding a particular match. Rarely used in lieu of getmatchdetails().

/**getesportsproleaguedetails**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}
Returns the matchup information for each matchup for the current eSports Pro League season. An important return value is “match_status” which represents a match being scheduled (1), in-progress (2), or complete (3)

/**getfriends**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{player}
Returns the Smite User names of each of the player’s friends. *[PC only]*

/**getgodranks**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{player}
Returns the Rank and Worshippers value for each God a player has played.

/**getchampionranks**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{player}
Returns the Rank and Worshippers value for each Champion a player has played. *[PaladinsAPI only]*

/**getgods**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{languageCode}
Returns all Gods and their various attributes.

/**getchampions**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{languageCode}

Returns all Champions and their various attributes. *[PaladinsAPI only]*

/**getgodleaderboard**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{godId}/{queue}
Returns the current season's leaderboard for a god/queue combination. *[SmiteAPI; only queues 440, 450, 451]*

/**getchampionleaderboard**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{godId}/{queue}
Returns the current season's leaderboard for a champion/queue combination. *[PaladinsAPI; only queue 428]*

/**getgodskins**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{godId}/{languageCode}
Returns all available skins for a particular God.

/**getchampionskins**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{godId}/{languageCode}
Returns all available skins for a particular Champion. *[PaladinsAPI only]*

/**getgodrecommendeditems**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{godid}/{languageCode}
Returns the Recommended Items for a particular God. *[SmiteAPI only]*

/**getchampionecommendeditems**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{godid}/{languageCode}
Returns the Recommended Items for a particular Champion. *[PaladinsAPI only; Osbsolete - no data returned]*

/**getitems**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{languagecode}
Returns all Items and their various attributes.

/**getmatchdetails**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{match_id}
Returns the statistics for a particular completed match.

/**getmatchdetailsbatch**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{match_id,match_id,match_id,...match_id}
Returns the statistics for a particular set of completed matches. NOTE: There is a byte limit to the amount of data returned; please limit the CSV parameter to 5 to 10 matches because of this and for Hi-Rez DB Performance reasons.

/**getmatchplayerdetails**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{match_id}
Returns player information for a live match.

/**getmatchidsbyqueue**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{queue}/{date}/{hour}
Lists all Match IDs for a particular Match Queue; useful for API developers interested in constructing data by Queue. To limit the data returned, an {hour} parameter was added (valid values: 0 - 23). An {hour} parameter of -1 represents the entire day, but be warned that this may be more data than we can return for certain queues. Also, a returned "active_flag" means that there is no match information/stats for the corresponding match. Usually due to a match being in-progress, though there could be other reasons.

- **NOTE** - To avoid HTTP timeouts in the GetMatchIdsByQueue() method, you can now specify a 10-minute window within the specified {hour} field to lessen the size of data returned by appending a ",mm" value to the end of {hour}. For example, to get the match Ids for the first 10 minutes of hour 3, you would specify {hour} as "3,00". This would only return the Ids between the time 3:00 to 3:09. Rules below:
    - Only valid values for mm are "00", "10", "20", "30", "40", "50"
    - To get the entire third hour worth of Match Ids, call GetMatchIdsByQueue() 6 times, specifying the following values for {hour}: "3,00", "3,10", "3,20", "3,30", "3,40", "3,50".

- The standard, full hour format of {hour} = "hh" is still supported.

/**getleagueleaderboard**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{queue}/{tier}/{season}
Returns the top players for a particular league (as indicated by the queue/tier/season parameters).

/**getleagueseasons**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{queue}
Provides a list of seasons (including the single active season) for a match queue.

/**getmatchhistory**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{player}
Gets recent matches and high level match statistics for a particular player.

/**getmotd**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}
Returns information about the 20 most recent Match-of-the-Days.

/**getplayeridinfoforxboxandswitch**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{playerName}
Meaningful only for the Paladins Xbox API. Paladins Xbox data and Paladins Switch data is stored in the same DB. Therefore a Paladins Gamer Tag value could be the same as a Paladins Switch Gamer Tag value. Additionally, there could be multiple identical Paladins Switch Gamer Tag values. The purpose of this method is to return all Player ID data associated with the playerName (gamer tag) parameter. The expectation is that the unique player_id returned could then be used in subsequent method calls. *[PaladinsAPI only]*

/**getplayer**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{player}
Returns league and other high level data for a particular player.

/**getplayerloadouts**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/playerId}/{languageCode}
Returns deck loadouts per Champion. *[PaladinsAPI only]*

/**getplayerstatus**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{player}
Returns player status as follows:

0 - Offline
1 - In Lobby (basically anywhere except god selection or in game)
2 - god Selection (player has accepted match and is selecting god before start of game)
3 - In Game (match has started)
4 - Online (player is logged in, but may be blocking broadcast of player state)
5 - Unknown (player not found)

/**getqueuestats**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{player}/{queue}
Returns match summary statistics for a (player, queue) combination grouped by gods played.

/**getteamdetails**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{clanid}
Lists the number of players and other high level details for a particular clan.

/**getteammatchhistory**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{clanid}
Gets recent matches and high level match statistics for a particular clan/team.

***DEPRECATED* - As of 2.14 Patch, /getteammatchhistory is no longer supported and will return a NULL dataset.**

/**getteamplayers**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{clanid}
Lists the players for a particular clan.

/**gettopmatches**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}
Lists the 50 most watched / most recent recorded matches.

/**searchteams**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{searchTeam}
Returns high level information for Clan names containing the "searchTeam" string. *[SmiteAPI only]*

/**getplayerachievements**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}/{playerId}
Returns select achievement totals (Double kills, Tower Kills, First Bloods, etc) for the specified playerId.
[SMITEAPI only]

/**getpatchinfo**[ResponseFormat]/{developerId}/{signature}/{session}/{timestamp}
Function returns information about current deployed patch. Currently, this information only includes patch version.

API Parameter Details

- *date* - a string in the format "20171231" (for Dec 31, 2017, as an example)
- *queue* - the id of the game mode (*last updated 2018-03-01, ordered by most played*):

most_played_queue_ranking	smite_match_queue_id	smite_match_queue_name	paladins_match_queue_id	paladins_match_queue_name
1	435	Arena Queue	424	LIVE Casual
2	448	Joust Queued (3v3)	469	LIVE Team Deathmatch
3	426	Conquest	452	LIVE Onslaught
4	445	Assault	428	LIVE Competitive
5	466	Clash	465	Classic Siege
6	495	Adventures - Horde	425	LIVE Practice Siege
7	451	Conquest Ranked	453	LIVE Onslaught Practice
8	434	MOTD	470	LIVE Team Deathmatch Practice
9	459	Siege 4v4	445	LIVE Test Maps
10	450	Joust 3v3 Ranked	474	LIVE Battlegrounds Solo
11	440	Joust Ranked (1v1)	475	LIVE Battlegrounds Duo
12	441	Joust Challenge	476	LIVE Battlegrounds Quad
13	457	Arena (vs AI) (Easy)	472	Custom T_Magistrate's Archives
14	468	Arena (vs AI) (Medium)	468	Custom T_Trade District
15	436	Basic Tutorial	423	Custom S_StoneKeep
16	438	Arena Challenge	471	Custom T_Foreman's Rise
17	429	Conquest Challenge	433	Custom S_FrogIsle
18	482	Joust 3v3 Training	431	Custom S_FishMarket
19	462	Arena Tutorial	458	Custom S_Brightmarsh
20	483	Arena Training	430	Custom S_TimberMill
21	474	Joust (vs AI) (Easy)	440	Custom S_SerpentBeach
22	456	Joust (vs AI) (Medium)	438	Custom S_JaguarFalls
23	478	Clash (vs AI) (Easy)	459	Custom S_SplitstoneQuarry
24	469	Clash (vs AI) (Medium)	464	Custom O_Magistrate's Archives
25	472	Arena Practice (Medium)	432	Custom S_FrozenGuard
26	481	Assault (vs AI) (Easy)	462	Custom O_Foreman's Rise
27	476	Conquest (vs AI) (Easy)	439	Custom S_IceMines
28	471	Clash Tutorial	455	Custom O_PrimalCourt
29	473	Joust Practice (Medium)	454	Custom O_SnowfallJunction
30	446	Assault Challenge	NULL	NULL
31	454	Assault (vs AI) (Medium)	NULL	NULL
32	443	Arena Practice (Easy)	NULL	NULL
33	467	Clash Challenge	NULL	NULL
34	461	Conquest (vs AI) (Medium)	NULL	NULL
35	464	Joust Practice (Easy)	NULL	NULL
36	477	Clash Practice (Medium)	NULL	NULL
37	460	Siege Challenge	NULL	NULL
38	480	Assault Practice (Medium)	NULL	NULL
39	463	Conquest Tutorial	NULL	NULL
40	470	Clash Practice (Easy)	NULL	NULL
41	479	Assault Practice (Easy)	NULL	NULL
42	475	Conquest Practice (Medi...	NULL	NULL
43	458	Conquest Practice (Easy)	NULL	NULL
44	444	Jungle Practice	NULL	NULL
45	496	Jungle Practice (Presele...	NULL	NULL

*For Smite, queue_id's 435, 448, 445, 426, 451, 459, 450, & 440 are the only ones considered for player win/loss stats

from **/getplayer**.

- *languageCode* - the language Id that you want results returned in. Default is 1.
  - 1 - English
  - 2 - German
  - 3 - French
  - 5 - Chinese
  - 7 - Spanish
  - 9 - Spanish (Latin America)
  - 10 - Portuguese
  - 11 - Russian
  - 12 - Polish
  - 13 - Turkish

- *match_id* - The id of the match. Can be obtained from /getmatchHistory, /gettopmatches & /getmatchidsbyqueue.

- *season* - The season of a league. Starts at 1 and increases by 1 for each calendar year. As of 2017-02-01 we are currently on season 4.

- *tier* - League tier
  - Bronze V = 1, Bronze IV = 2, Bronze III = 3, Bronze II = 4, Bronze I = 5
  - Silver V = 6, Silver IV = 7, Silver III = 8, Silver II = 9, Silver I = 10
  - Gold V = 11, Gold IV = 12, Gold III = 13, Gold II = 14, Gold I = 15
  - Platinum V = 16, Platinum IV = 17, Platinum III = 18, Platinum II = 19, Platinum I = 20
  - Diamond V = 21, Diamond IV = 22, Diamond III = 23, Diamond II = 24, Diamond I = 25
  - Masters I = 26, Grandmaster = 27

- *Player* - This may either be a,) the Player Name, or b.) the Hirez internally stored player_id (available to API developers via the /getplayer API method).

- *PlayerId* - This is the Player ID.

- *Player Name* - This is the Player Name.

- *clanId* - id of the clan. Can be obtained from searchteams

- *searchTeam* - name of clan for whom to search

CREATING A SESSION

The url format for calling a method from the api is http://api.smitegame.com/smiteapi.svc/ + the pattern for the method above, where [ResponseFormat] is replaced by the formatting that you want returned (either XML or JSON).

To create a session with a JSON response, call the **createsession** method as follows:

http://api.smitegame.com/smiteapi.svc/createsessionJson/1004/8f53249be0922c94720834771ad43f0f/20120927183145

which would return JSON data such as:


```
{
	"ret_msg": "Approved",
	"session_id": "0ECDF26BC1F04EE4BA4AF10EC3604E04",
	"timestamp": "2/14/2013 7:50:20 PM"
}
```


The sessionId is contained in an element called “session_id”. This parameter is needed to call the other methods.

You'll see that we passed in a few other variables besides our **devId** and **authKey**.

Actually the authKey is not passed directly, but instead embedded and hashed in another parameter (**signature**).

CREATING A SIGNATURE

A distinct signature is required for each API method called.

The signature is created by concatenating several fields and then hashing the result with an MD5 algorithm. The components of this hash are (in order):

1. your devId
2. the method name being called (eg, "createsession")
    a. This will not include the ResponseType, just the name of the method.
3. your authKey
4. current utc timestamp (formatted yyyyMMddHHmmss)

Sample C# Code to Create a Signature:

```
var signature = GetMD5Hash("1004" + "createsession" +
"23DF3C7E9BD14D84BF892AD206B6755C" + "20120927183145");

private static string GetMD5Hash(string input) {
      var md5 = new System.Security.Cryptography.MD5CryptoServiceProvider();
      var bytes = System.Text.Encoding.UTF8.GetBytes(input);
      bytes = md5.ComputeHash(bytes);
      var sb = new System.Text.StringBuilder();
      foreach (byte b in bytes) {
            sb.Append(b.ToString("x2").ToLower());
      }
      return sb.ToString();
}
```

EXAMPLE API CALL

The uri to use for all API calls starts with *http://api.smitegame.com/smiteapi.svc/* followed by a slash + the method + any parameters to complete the call.

For example, to get stats for a given player, call **getplayer** in the following manner:

> http://api.smitegame.com/smiteapi.svc/getplayerjson/1004/0abd990b4ca9f86817e087ad684515db/83B082E576584DA8B1DB073DECA9E819/20120927193800/HirezPlayer

Again, the complete pattern for this call is:

> getplayer[ResponseFormat]/{devId}/{signature}/{sessionId}/{timestamp}/{playerName}

A JSON [ResponseFormat] for this call would provide JSON results as follows:


```
[
	{
	"Created_Datetime": "5/30/2012 2:34:40 PM",
	"Last_Login_Datetime": "8/24/2013 12:02:20 AM",
	"Leaves": 0,
	"Level": 30,
	"Losses": 10,
	"MasteryLevel": 0,
	"Name": "HirezPlayer",
	"Rank_Stat": 0,
	"TeamId": 0,
	"Team_Name": "",
	"Wins": 40,
	"ret_msg": null
	}
]
```


The MatchId Parameter

The pattern for getmatchstats is:

getmatchstats[ResponseFormat]/{devId}/{signature}/{sessionId}/{timestamp}/{matchId}

The {matchId} parm is a unique id for each map that's created by the server for a set of players. One place you can get this value from will be getmatchhistory.

GRAPHICS

You can find any graphics that we've published for use here.

APPENDIX A - COMPREHENSIVE CODE EXAMPLE

To assist with development & debugging efforts, the following sample Microsoft Visual Studio 2012 WindowsFormApplication is provided. Note that you may have to add a few Assembly References to your project if some of the System.* classes can't initially be found.

The application simply consists of:

a.) a button to call the **createsession** API method, and
b.) a button to call the **getgods** API method and some logic to display all gods in a MessageBox.

The values for devKey and authKey are removed for security purposes.

Note that the API calls are synchronous and may take a few seconds before generating a response. You will probably call the API asynchronously, but for purposes of this exercise (to quickly understand how to work with the API methods) the synchronous method was used.

The application consists of two primary forms (Form1.cs & Form1.Designer.cs) listed on the following pages.

The *Form1.Designer.cs* file (for the buttons):

```
namespace WindowsFormsApplication1
{
   partial class Form1
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

      #region Windows Form Designer generated code

      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
         this.buttonCreateSession = new System.Windows.Forms.Button();
         this.buttonGetGods = new System.Windows.Forms.Button();
         this.label1 = new System.Windows.Forms.Label();
         this.label2 = new System.Windows.Forms.Label();
         this.SuspendLayout();
         //
         // buttonCreateSession
         //
         this.buttonCreateSession.Location = new System.Drawing.Point(280, 168);
         this.buttonCreateSession.Name = "buttonCreateSession";
         this.buttonCreateSession.Size = new System.Drawing.Size(205, 23);
         this.buttonCreateSession.TabIndex = 1;
         this.buttonCreateSession.Text = "Create Session\r\n";
         this.buttonCreateSession.UseVisualStyleBackColor = true;
         this.buttonCreateSession.Click += new System.EventHandler(this.buttonCreateSession_Click);
         //
         // buttonGetGods
         //
         this.buttonGetGods.Location = new System.Drawing.Point(280, 211);
         this.buttonGetGods.Name = "buttonGetGods";
         this.buttonGetGods.Size = new System.Drawing.Size(205, 23);
         this.buttonGetGods.TabIndex = 2;
         this.buttonGetGods.Text = "Call GetGods() API Method\r\n";
```

```
            this.buttonGetGods.UseVisualStyleBackColor = true;
            this.buttonGetGods.Click += new System.EventHandler(this.buttonGetGods_Click);
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(229, 173);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(38, 13);
            this.label1.TabIndex = 4;
            this.label1.Text = "Step 1";
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(229, 216);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(38, 13);
            this.label2.TabIndex = 5;
            this.label2.Text = "Step 2";
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(802, 414);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.buttonGetGods);
            this.Controls.Add(this.buttonCreateSession);
            this.Name = "Form1";
            this.Text = "Form1";
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.Button buttonCreateSession;
        private System.Windows.Forms.Button buttonGetGods;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label2;
    }
}
```

The *Form1.cs* File (For the API Method Calls)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.IO;
using System.Threading.Tasks;
using System.Windows.Forms;

using System.Security.Cryptography;
using System.Net;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Json;
using System.Web.Script.Serialization;

namespace WindowsFormsApplication1
{

    public partial class Form1 : Form
    {
        string devKey = "XXXX"; // devKey goes here
        string authKey = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";  // authKey goes here
        string timestamp = DateTime.UtcNow.ToString("yyyyMMddHHmmss");
        string urlPrefix = "http://api.smitegame.com/smiteapi.svc/";


        string signature = "";
        string session = "";

        public Form1()
        {
            InitializeComponent();
        }

        private static string GetMD5Hash(string input)
        {
            var md5 = new System.Security.Cryptography.MD5CryptoServiceProvider();
            var bytes = System.Text.Encoding.UTF8.GetBytes(input);
            bytes = md5.ComputeHash(bytes);
            var sb = new System.Text.StringBuilder();
            foreach (byte b in bytes)
            {
                sb.Append(b.ToString("x2").ToLower());
            }
            return sb.ToString();
        }

        private void buttonCreateSession_Click(object sender, EventArgs e)
        {
            // Get Signature that is specific to "createsession"
            //
```

```
        signature = GetMD5Hash(devKey + "createsession" + authKey + timestamp);

        // Call the "createsession" API method & wait for synchronous response
        //
        WebRequest request = WebRequest.Create(urlPrefix + "createsessionjson/" + devKey + "/" + signature + "/" + timestamp);
        WebResponse response = request.GetResponse();

        Stream dataStream = response.GetResponseStream();
        StreamReader reader = new StreamReader(dataStream);

        string responseFromServer = reader.ReadToEnd();

        reader.Close();
        response.Close();

        // Parse returned JSON into "session" data
        //
        using (var web = new WebClient())
        {
            web.Encoding = System.Text.Encoding.UTF8;
            var jsonString = responseFromServer;
            var jss = new JavaScriptSerializer();
            var g = jss.Deserialize<SessionInfo>(jsonString);

            session = g.session_id;

            MessageBox.Show(session);
        }
    }

    private void buttonGetGods_Click(object sender, EventArgs e)
    {
        // Get Signature that is specific to "getgods"
        //
        signature = GetMD5Hash(devKey + "getgods" + authKey + timestamp);


        // Call the "getgods" API method & wait for synchronous response
        //
        string languageCode = "1";

        WebRequest request = WebRequest.Create(urlPrefix + "getgodsjson/" + devKey + "/" + signature + "/" + session + "/" + timestamp + "/"
+ languageCode);
        WebResponse response = request.GetResponse();

        Stream dataStream = response.GetResponseStream();
        StreamReader reader = new StreamReader(dataStream);

        string responseFromServer = reader.ReadToEnd();

        reader.Close();
        response.Close();

        // Parse returned JSON into "gods" data
        //
        using (var web = new WebClient())
        {
```

```
        web.Encoding = System.Text.Encoding.UTF8;
        var jsonString = responseFromServer;
        var jss = new JavaScriptSerializer();
        var GodsList = jss.Deserialize<List<Gods>>(jsonString);
        string GodsListStr = "";

        foreach (Gods x in GodsList)
            GodsListStr = GodsListStr + ", " + x.Name;

        MessageBox.Show("Here are the Gods: " + GodsListStr);
    }

}

public class SessionInfo
{
    public string ret_msg { get; set; }
    public string session_id { get; set; }
    public string timestamp { get; set; }

}

public class Menuitem
{
    public string description { get; set; }
    public string value { get; set; }
}

public class Rankitem
{
    public string description { get; set; }
    public string value { get; set; }
}

public class AbilityDescription
{
    public string description { get; set; }
    public string secondaryDescription { get; set; }
    public List<Menuitem> menuitems { get; set; }
    public List<Rankitem> rankitems { get; set; }
    public string cooldown { get; set; }
    public string cost { get; set; }
}

public class AbilityRoot
{
    public AbilityDescription itemDescription { get; set; }
}

public class Gods
{
    public int abilityId1 { get; set; }
    public int abilityId2 { get; set; }
    public int abilityId3 { get; set; }
    public int abilityId4 { get; set; }
    public int abilityId5 { get; set; }
    public AbilityRoot abilityDescription1 { get; set; }
```

```
        public AbilityRoot abilityDescription2 { get; set; }
        public AbilityRoot abilityDescription3 { get; set; }
        public AbilityRoot abilityDescription4 { get; set; }
        public AbilityRoot abilityDescription5 { get; set; }
        public int id { get; set; }
        public string Pros { get; set; }
        public string Type { get; set; }
        public string Roles { get; set; }
        public string Name { get; set; }
        public string Title { get; set; }
        public string OnFreeRotation { get; set; }
        public string Lore { get; set; }
        public int Health { get; set; }
        public Double HealthPerLevel { get; set; }
        public Double Speed { get; set; }
        public Double HealthPerFive { get; set; }
        public Double HP5PerLevel { get; set; }
        public Double Mana { get; set; }
        public Double ManaPerLevel { get; set; }
        public Double ManaPerFive { get; set; }
        public Double MP5PerLevel { get; set; }
        public Double PhysicalProtection { get; set; }
        public Double PhysicalProtectionPerLevel { get; set; }
        public Double MagicProtection { get; set; }
        public Double MagicProtectionPerLevel { get; set; }
        public Double PhysicalPower { get; set; }
        public Double PhysicalPowerPerLevel { get; set; }
        public Double AttackSpeed { get; set; }
        public Double AttackSpeedPerLevel { get; set; }
        public string Pantheon { get; set; }
        public string Ability1 { get; set; }
        public string Ability2 { get; set; }
        public string Ability3 { get; set; }
        public string Ability4 { get; set; }
        public string Ability5 { get; set; }
        public string Item1 { get; set; }
        public string Item2 { get; set; }
        public string Item3 { get; set; }
        public string Item4 { get; set; }
        public string Item5 { get; set; }
        public string Item6 { get; set; }
        public string Item7 { get; set; }
        public string Item8 { get; set; }
        public string Item9 { get; set; }
        public int ItemId1 { get; set; }
        public int ItemId2 { get; set; }
        public int ItemId3 { get; set; }
        public int ItemId4 { get; set; }
        public int ItemId5 { get; set; }
        public int ItemId6 { get; set; }
        public int ItemId7 { get; set; }
        public int ItemId8 { get; set; }
        public int ItemId9 { get; set; }
        public string ret_msg { get; set; }
    }
  }
}
```