Implementing the API (updated June 27th, 2013)

API version .2

This is quick update to enable features required for Geopalz iTunes submissions.
On iOS there are no quit buttons so we have added a new config layer to the API.

At a minimum please add the following to your games.

Check this variable after getting the onAPIReady event back from the wrapper.

if (DGGameAPI.config.showQuitButtons==false) //hide your quit button(s)

Verify the integration is working by running your game in the Sandbox with
_TestGameLandscapeHidingButton.bat
And
_TestGamePortraitHidingButtons.bat

While you are making this change you might add a few more elements which will help us better monetize your game.

if (DGGameAPI.config.showMoreGamesButton==false) //hide your more game button(s)
//This will be used for channels which don't support cross linking and other OEM deals.

if (DGGameAPI.config.showLeaderboardButtons==false) //hide your submit score and showLeaderboard buttons.
//Obviously only relevant if you are calling the other leaderboard related functions.

//Call these commands to let the wrapper know where the user is at within your games.
//This enables us to work with several different 3rd party SDK's by overlaying information over The game at appropriate points. It will also enable greater anayalics support in the future.
DGGameAPI.reportGameState(DGGameAPI.GAMESTATE_MAIN_MENU);
DGGameAPI.reportGameState(DGGameAPI.GAMESTATE_PLAYING);
DGGameAPI.reportGameState(DGGameAPI.GAMESTATE_GAME_END);
DGGameAPI.reportGameState(DGGameAPI.GAMESTATE_LEVEL_SELECT);
DGGameAPI.reportGameState(DGGameAPI.GAMESTATE_LEVEL_END);

*Also check out the show3rdPartyAd information at the end.

The first step is to quickly run through the example games within the testing environment.

Run **"_TestGameLandscape.bat"** and **"_TestGamePortrait.bat"** and click the various menu buttons.

Start a game to see the flow of ads / pop up events.

Click on your desktop to defocus the game. Notice how the music stops.

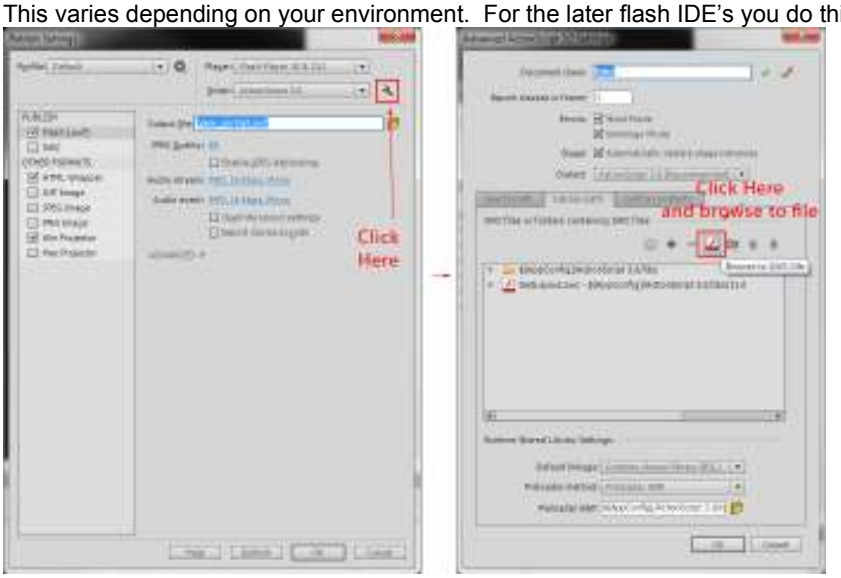
Click back to the title bar of the 'dg game shim' (not the game itself). This will return focus to the game and show a 'tap to resume' overlay. Click on the overlay, the music will resume and the game can continue.

Once you are finished exploring the demo games try to launch the SWF files (game.swf and gameP.swf) directly. Notice how they do not go past the preloader. They are waiting on a 'go' command from the API before they will continue.

Now let's prepare your environement.

Include the SWC Library (DGGameAPI.swc) into your project.

This varies depending on your environment. For the later flash IDE's you do this from Actionscript settings.



Once the SWC is linked you need to initialize at the start of your game. This should be the very first script that runs in your game including a preloader if you are using one. Note: preloaders are not required as the game will already be fully loaded on the user's device before it runs.

Activate the API as soon as your game document class is initialized (as soon as you have access to the .stage property.):

```
//Import the api classes:
import com.differencegames.dggameapi.events.DGGameEvent;
import com.differencegames.dggameapi.DGGameAPI;

// init api once you have access to the stage, set ready listener
DGGameAPI.create(stage,"com.differencegames.yourgamename").
addEventListener(DGGameEvent.CONTAINER_READY,onApiReady);

// start connection.  Send this after your game is ready to be played.
// This would be after any preloader.
DGGameAPI.gameReady();

// Listen to the API ready event.  This event will only be sent when
// we wrap the game in the platform or if you are testing in the
// local Shim environment.   It is very important that your game does
// not run before it receives the ready event.
```

```
function onApiReady(e:Event=null):void {
      //start your game.
}
```

Listen to the onApiReady event. Once the event has fired, you may proceed as normal with your game. Don't forget to the send the gameReady() event as shown above. The API will not send the onApiReady event until after you have called DGGamePI.gameReady().

There is sample code for Flash IDE and Actionscript 3 environments included in the folder.

Adding a More Games Button

When you call for the More Games display, your game will be obscured by a popover screen until the user dismisses it. You may want to save state at this time as there is no guarantee the user will return to your game before it is shutdown:

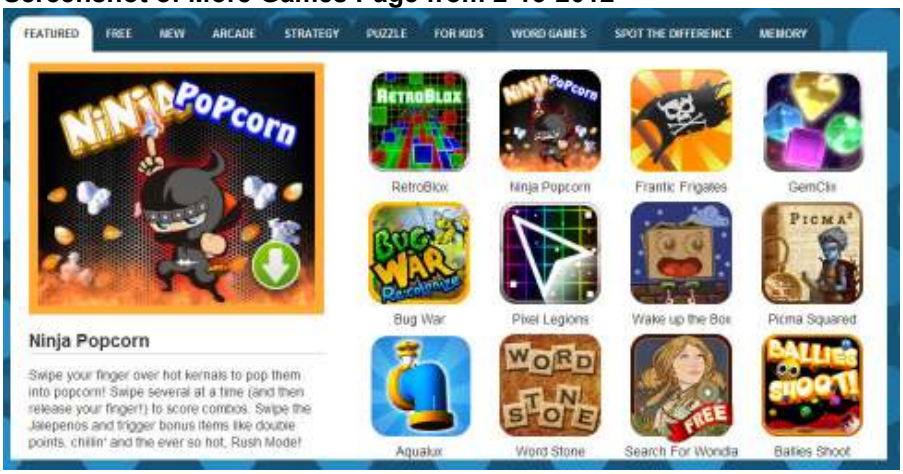
```
DGGameAPI.showMoreGames();
```

Optionally you can listen for two events- one when the moreGames page is about to display, and another when it's dismissed and control is returned to your App:

```
DGGameAPI.addEventListener(DGGameEvent.MORE_GAMES_SHOWING,onMoreGamesShowing);
DGGameAPI.addEventListener(DGGameEvent.MORE_GAMES_DISMISSED,onMoreGamesDismissed);
```

The More Games overlay is a cross promotion system for promoting all the apps within the network. Clicking on the icons within it will directly open up the app purchase page on the user's device. You will be able to see how this looks when you run the game within the shim.

Screenshot of More Games Page from 2-13-2012



Lifecycle Events

At any time, the wrapper, the user, or the OS may cause your game to be paused, or quit. In either case, there is no guarantee the game will be resumed before being shutdown by the OS; it is a good idea to save your game state at this time. Also, when you receive a pause event, you should suspend all audio playback immediately (**otherwise the user may be listening to your music while running other apps and the game will get rejected when the sales channel QA teams reviews it)**
You may resume audio when the resume event is fired.

Listening for Quitting, Pausing, and resume events:

```
DGGameAPI.addEventListener(DGGameEvent.GAME_PAUSING,onGamePausing);
DGGameAPI.addEventListener(DGGameEvent.GAME_RESUMED,onGameResumed);
DGGameAPI.addEventListener(DGGameEvent.GAME_QUITTING,onGameQuitting);
```

NOTE: If you're already listening for Event.ACTIVATE or Event.DEACTIVATE on the stage, remove these listeners in favor of the ones above; they may not fire at the appropriate times within the wrapper.

Quit Button

All games should include a 'quit' option on the main menu. Use the following API call to quit the game.

```
DGGameAPI.requestQuit();
```

Things your game shouldn't do:

Do not listen to activate and deactivate events. Instead listen to the pause and resume events from the API.

Do not set the stage scale mode. The game should run at 1024x600 and leave all scaling to our wrapper. If your game is capable of smartly scaling to any screen resolution then contact us for details. We do have a special mode for this but it isn't activated within the default testing shim.

Do not set the stage quality. We adjust the quality setting for individual devices in order to achieve the best combination of performance and visual quality.

Do not listen and react to orientation change events.

Don't use the right click contextMenu. This will work in the testing environment but crash once it's published to mobile,

Testing With the Shim

Build your project name it 'game.swf' and place it in testing directory. Run **_TestGameLandscape.bat** if it should run in landscape mode and **_TestGamePortrait.bat** if it should run in portrait mode. Note: you can edit the bat file to make it automatically rename a swf in the folder into 'game.swf'. Add these lines to the top if you want to streamline testing a little further.
del game.swf
copy yourGameNameExportFromFlash.swf game.swf
[portrait is the same except use gameP.swf]

The console window will show you when events are successfully fired. The CONTAINER_READY event will be dispatched to your application after load. You can use the 'simQuit', 'simPause', 'simResume', 'nook', 'amazon' and 'simMoreGames' buttons to send these events to your program and test reactions.

That is all that is required for the minimum integration. On the following pages are optional advanced integrations you are welcome to explore.

Optional Integration

In-Game Ads

The in-game 300x250 ads will exclusively promote your own titles and once you have some paid titles listed it can be great to release a free one with the ad system to get additional exposure. It is possible to implement the ads on paid games as well but it should be done very tastefully in order to not degrade the player experience.

Show In-Game Ads

```
DGGameAPI.showAd(xPos:Number,yPos:Number)
//shows the advertisement and positions it on the screen at the x,y position within your game. This will be overlaid on top of your game (ie, not in your display list). You must call remove ad when the user would leave the screen where the ad is displayed.
```

Show the next Ad (optional)

```
DGGameAPI.showNextAd();
//Cycles through the different ads. Because the ads are all games within our network a lot of players do click buttons to show the next game and cycle through the promotional ads. The example game included in this package implements this functionality.
```

Remove the Ad

```
DGGameAPI.removeAd()
//removes the advertisement. You must call this when you leave the screen where the ad should be shown
```

Hungry Critters (a free game we released to promote the network) with an in-game ad on main menu.

Implementing the device back button
The default behavior is for the wrapper to handle this event and it will simply exit out of your game when it happens. In many cases this is actually the preferred outcome since it can let the user return to the same state in the game in the future.

The alternative route is to move the user back 1-level within your game at each press of this button.

For example (user presses the back button three times):
User Playing a Level -> Level Select Screen -> Game Main Menu -> Exit Game

If you want to listen to back button events you must do two things.
```
DGGameAPI.gameHandlesBackButton=true;  //defaults to false
```

Second listen to back button event. This event will not be sent unless you set the above variable to true.
```
DGGameAPI.addEventListener(DGGameEvent.USER_PRESSED_BACK_BUTTON,onBackButton);

private function onBackButton(e:Event=null):void {
      //back your game up or exit if there is nowhere else to go.
}
```

It is CRITICAL that your game will move backwards (ie from game play to main menu) each time the event is received. If there is nowhere else to back up then you must quit your game. Failure to do this could leave the user stuck in the game which will cause the application to get rejected or bad comments and ratings from the players

Note: Again, it is not required to implement the button and in some cases it can be preferable not to use it. The default behavior will suspend the application with a back button press. That will let the player resume it at their existing state if they return quickly. If they take too long then the device will shut down the game to conserve resources.

Debugging – Trace Style Messages
```
//lets you output messages to the testing shim console and can be very helpful for debugging.
DGGameAPI.log("This will appear in the console!");
```

Scores, leaderboards, achievements

*You will not see these in the testing shim, but there is a log event to verify receipt of the call.

We are working on a wider system that will integrate across many services and also leverage our own GamerSafe platform depending on the sales channel we are targeting. Currently we using Amazon, HeyZap and Gramble for leaderboards depending on the opportunity available. The leaderboard will only show up when you call displyScoreboard. If you want to submit and show score at same time send both calls.

```
DGGameAPI.submitScore(leaderBoardID,score,"");
//DGGameAPI.submitScore("HighScores",5000,""), DGGameAPI.submitScore("HardMode",3000,"")

DGGameAPI.displayScoreboard(leaderBoardIdName);
//DGGameAPI.displayScoreboard("HighScores"), DGGameAPI.displayScoreboard("HardMode")
```

Example of how it is displayed on-device.

Showing 3rd Party Ads

Updated (June 27th) Currently we have integration with 6 difference ad providers and are adding more all the time. We can control which networks run through our server so will be optimizing the inventory to get the best possible return. Currently these ad formats are getting $5-$15 eCPM across GooglePlay and Amazon.

The ads are player friendly and can be immediately dismissed vs. forcing them to watch a video or other type of forced delay.

Tiny Monsters ad running within ChartBoost, presented on main menu of game.

Almost everything is paid on a per install basis. The ads are interesting so they always end up catching some people but when you trigger them has a big impact. One of the best spots is to show the ad right as the player would be finished playing your game. At that point they are at the most likely state to try something new and install the advertised app.

```
//The default call is very simple.
DGGameAPI.show3rdPartyAd()

//Optionally you can add one or two parameters.
DGGameAPI.show3rdPartyAd(location:String);
// DGGameAPI.show3rdPartyAd("gameMenu");
// DGGameAPI.show3rdPartyAd("end level screen");
// This can allow more granular control over the ads in the future


DGGameAPI.show3rdPartyAd(location:String,extraData:String);
//The extraData parameter is not used currently.  It might be needed for different ad
networks in the future.
```