Logitech|G Arx Control Applet Development Guide

Contents

Overview

This document is meant to give useful guidelines to Logitech|G Arx Control Applet developers. Reference to the Software Development Kit can be found in the document `LogitechGArxControl.pdf` in this folder.
This documents assumes some basic web development skills from the reader, examples provided are chunks of html, javascript and css code.

Layout design and development

When designing the applet, keep in mind that it will run on different devices with different screen sizes, since Arx Control is available for both iOS and Android.
Here are some useful tips to consider when designing the applet layout.

Relative percentage content size

When defining size of content in your html page, use relative percentage dimensions as much as possible. This will ensure that the applet properly scales on any device.

Example.

APPLET HTML

```
.
.
<div id = ‘characterStats’>
	<div id = ‘healthBar’></div>
	<div id = ‘weaponIndicator’></div>
</div>
.
.
.
```

APPLET CSS

```
#characterStats
{
	height: 50%;
	width: 100%;
	background-color:red;
}
#healthBar
{
	height: 20%;
	width: 100%;
	background-color:blue;
}
#weaponIndicator
{
	margin-top:1%;
	height:20%;
	width:500%;
	background-color:green;
```

```
}
```

Prevent selection of elements on long tap

By default iOS highlights elements and makes them copyable to clipboard on long tap user action. Often this behavior is not desired on Arx Control applets. To prevent this behavior, use this code:

APPLET CSS

```
*
{
	-webkit-touch-callout: none;
	-webkit-user-select: none;
}
```

Prevent applet zoom and scroll

By default the applet is a fully featured webview and therefore it's zoomable and scrollable if the content size exceeds the screen size. If the layout of an applet is well designed it should fit in any screen and it shouldn't need any zoom or pan action. This is strongly advised since Arx Control is designed to be a secondary screen for in-game experience and therefore the applet should serve the gamer needs at a glance. There is different ways to prevent your applet to be zoomable and scrollable, meta tags is a good one:

APPLET HTML

```
<head>
	<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-
scale=1, target-densityDpi=device-dpi, user-scalable=no" />
.
.
.
</head>
```

Default Background

By default the applet background is set to the Logitech|G image that is present in Logitech built-in applets. If the developer decides not to set a background property for the html body, it will be set to the default one to keep the app experience consistent.
To override this behavior, define a background property in the stylesheet as follows:

APPLET CSS

```
body
{
	background-image:url('urlToYourImage');
	//or background-color:black;
}
```

Applet development

In applet notification on tag update

While developing the applet it might be useful to know when an update of content or property has been fired from the game. Add the following Javascript to your code to be notified on any tag update

```
onPropertyUpdate = function () {
	//Do something here
}
```

This technique can be used in many different ways, one of them could be to activate a Javascript function at a given time from the game.

Example.

APPLET

```
<html>
<head></head>
<body>
.
.
<div id = 'health'></div>
.
.
</body>
.
<script>
onPropertyUpdate = function () {
	var health = parseInt(document.getElementById('health').innerHTML);
	if(health < 10)
		flashHealthIndicator(true); //defined somewhere in your script
	else
		stopFlashingHealthDiv(false); //defined somewhere in your script
}
</script>
```

GAME

```
LogiArxSetTagContentById("health","5"); //The applet will start flashing on the health indicator
LogiArxSetTagContentById("health","50"); //The applet will stop flashing on the health indicator
```

Link to remote resources in the applet

The app allows the applet to link to remote resources, for developers to reference to remote assets. This is not advised though, because the user might not be connected to the internet while using the app and that will cause missing assets and therefore bad user experience of the applet. To prevent this, it's preferable to send all the assets using the SDK and link only to local files.
A classic example is the use of the library jQuery. It can be really handy when it comes to more complex applet development and here is an example of how to do so.

Example

APPLET

```
<html>
<head>
<script src=”jquery-1.10.1.min.js”></script>
</head>
<body>
.
<div id=’element’></div>
.
</body>
.
<script>
$(“element”).hide();
</script>
```

GAME

```
//Download the jquery version to your local system and then
LogiArxAddFileAs(“jquery-1.10.1.min.js”,”path/to/downloaded/jquery-1.10.1.min.js”,
“application/javascript”)
```

Update content using special characters

When updating your tags content from the game, make sure to properly escape the special characters. In specific a string containing the double quote " will cause the update to be ignored. Please escape the double quote following this example.

Example

APPLET

```
<html>
<body>
.
<div id=’element’></div>
.
</body>
</body>
```

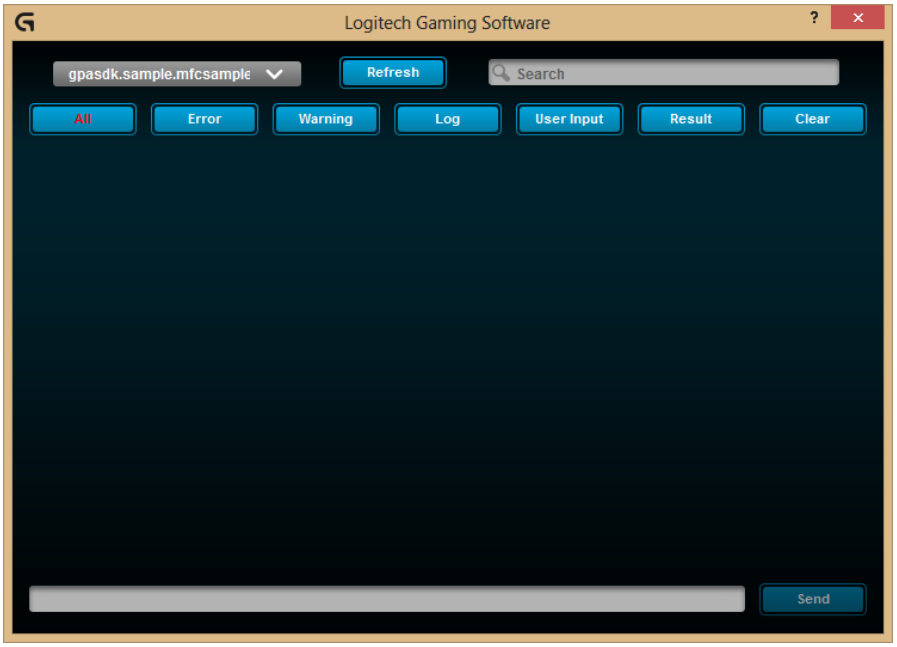
GAME

```
//Download the jquery version to your local system and then
LogiArxSetTagContentById(L"element", L"string containing a \\\"quote\\\” ");
```

Applet Debug

Debug console

Logitech provides a fully functional web debug console, to open this tool, open Logitech Gaming Software, click on the settings icon (bottom right) and then in the Mobile tab, click on the button "launch console". This dialog will open up.



This console is directly connected to one applet at the time, so to debug the app currently being developed, select the corresponding entry in the drop-down menu in the top left corner. If the desired applet is currently active on Arx Control, but it's not in the list, try using the refresh button and it will appear.

The console offers different filters to better debug the applet, any of the blue buttons will filter the debug log against its category, while the search bar in the right top corner will filter against the string being typed. The clear button clears only the output log for the currently selected applet, while if the debug console window is closed, any applet log is dropped.
The line at the bottom is an input line, pressing the send button will inject the typed string in the applet as Javascript code.

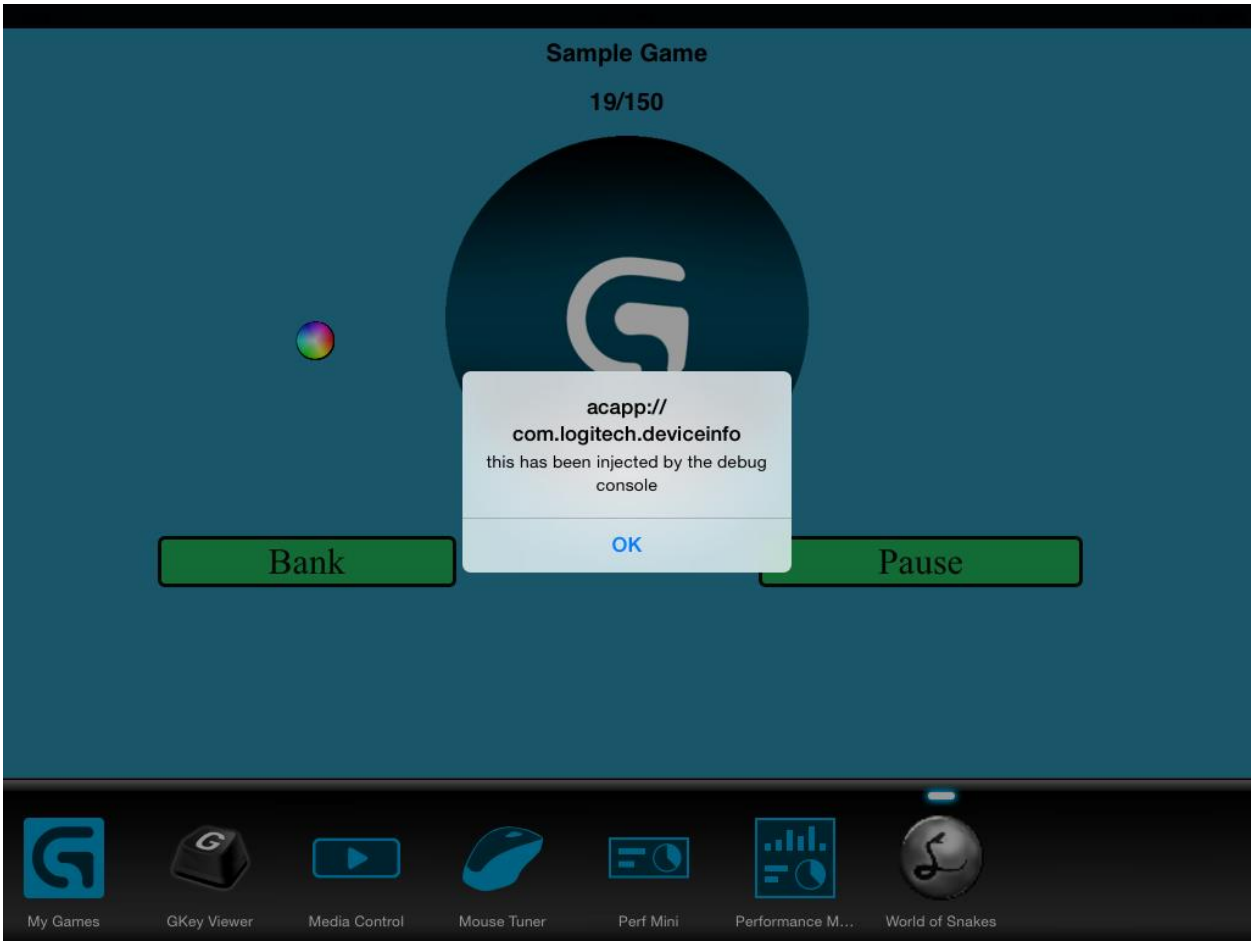
Example

DEBUG CONSOLE
For example: If you type in the following code in to the bottom text box:

alert("this has been injected by the debug console")

After pressing the send button, the applet will execute the Javascript and the result will be the following

APPLET



Content caching

By default the content sent to Arx Control is cached by the app webview. This can be an obstacle while developing. To avoid this behavior just keep the Debug Console open while developing and the App won't cache any content.
Please note that due to Apple's restrictions, at this point the caching removal only works on Android devices.

ACBridge

ACBridge is a javascript class that Arx Control injects in each applet to enhance its development and debugging capabilities. The class exposes the following public methods:

- ACBridge.click(buttonName)
- ACBridge.tr(string, tag)
- ACBridge.trAll()
- ACBridge.trLoadStrings()

In addition, ACBridge will call the following functions at the correct time if they exist:

- onPropertyUpdate()
- onACBridgeLoad()

Following is a detailed explanation of each method.

Send a programmatic click event

When ACBridge loads, every element with an ID will be assigned a default click event. When tapped, it will send a click event with its ID back to the game.
Sometimes it can be useful to send a programmatic click event from the applet to the game to send some data back. This can be achieved through ACBridge class, using the click method:

ACBridge.click()

A very common use of this functionality would be to add to the click event argument the data to send back to the game using string separator character.

Example
Let's assume that in this applet there is a loadout selector of some sort (`selectLoadOut`) and a button to apply the selected loadout to the game (`selectLoadOutButton`)
APPLET

```
<html>
<head></head>
<body>
.
.
<div id = 'selectLoadOut'>
       <div id='currentLoadout'>Assault</div>
</div>
<div id = 'selectLoadOutButton'></div>
.
.
</body>
.
<script>
document.getElementById('selectLoadOutButton').addEventListener('touchend', function (event) {
          var currentLoadout = document.getElelementById('currentLoadout).innerHTML;
    ACBridge.click("selectLoadout|" + currentLoadout);
  }, false);
</script>
```

GAME

```
//Function that will be called from the registered callback on event
//LOGI_ARX_EVENT_TAP_ON_TAG
void onAppletTap(String tagId)
{
       ...
       If(tagId.startsWith("selectLoadout"))
       {
              selectActiveLoadout(tagId.stringAfter("|"));
       }
```

```
}
```

Prevent ACBridge.click events

Sometimes it might come handy to prevent some tags to fire the ACBridge.click event. In order to do so, just add the class 'ignoreTapEvents' to the tag that we don't want to listen to.

Example
A classic case where this can be useful is when we have a button nested in a parent tag. In this case we only want the Arx applet to callback our game only for the button and not for its parent.

APPLET

```
<html>
<head></head>
<body>
.
.
<div id = 'parentDiv' class = 'ignoreTapEvents'>
      <div id='button'>Button</div>
</div>
.
</body>
.
```

GAME

The ingame function registered through callback during initialization will only be called for 'button' and not for 'parentDiv'. Without this solution, the game will receive two callbacks any time.

Log to Logitech debug console

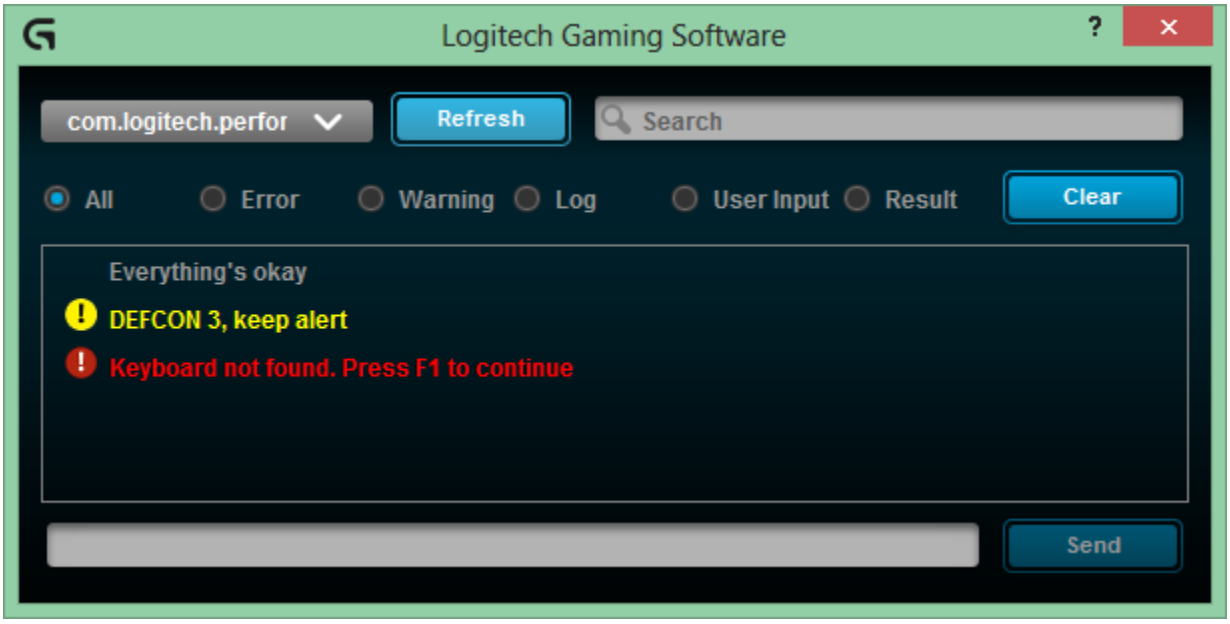
All standard console output streams will be piped to the Logitech Debug console.

Example:

APPLET

```
<script>
      console.log("Everything's okay");
      console.warn("DEFCON 3, keep alert");
      console.error("Keyboard not found. Press F1 to continue");
</script>
```

DEBUG CONSOLE



Translation module

The ACBridge has basic support for translation through the funcitons `tr(), trAll(), and trLoadStrings()`.

The `trLoadStrings()` function will load up a file called `localizedStrings.json`, if it exists.

`The trAll()` function will replace the text inside tags that have `data-trstring` attribute set, if the corresponding string exists in the `localizedStrings.json` file. Otherwise, the tag is left alone.

The `tr()` funciton takes in a string to translate, and optionally an HTML tag object to assign the translated string to. It will return the translation if it exists, or the original string if it does not. The tag, if it is passed in, will have its innerHTML overwritten if the translation is successful.

Example
APPLET

```
<html>
<head></head>
<body>
.
.
<div data-trstring=”hello”>
      Hello world!
</div>
.
</body>
.
<script>
var onACBridgeLoad = function()
{
```

```
        trLoadStrings();
        trAll();
        document.body.appendChild(document.createTextNode(“>” + tr(‘hello’) + “<”));
}
</script>
```

RESULT:

Event triggers

The ACBridge will call the `onACBridgeLoad()` function once the document and the ACBridge has finished loading. Consider this as a timing-safe alternative to the JQuery event `$(document).load()` or `.ready()` that makes sure that both the DOM and the ACBridge are ready to be used.

Example

APPLET

```
<html>
<head>
<script>
	var onACBridgeLoad = function()
	{
		var splineCount = reticulateSplines(document.getElementById(“params”));
		ACBridge.click(“splines|” + splineCount);
	}
</script>
</head>
<body>
.
<div id=”params”>
	7
</div>
.
</body>
.
</html>
```

GAME

```
//Function that will be called from the registered callback on event
//LOGI_ARX_EVENT_TAP_ON_TAG
void onAppletTap(String tagId)
{
	...
	If(tagId.startsWith(“splines”))
	{
		matchSplines(tagId.stringAfter(“|”));
	}
}
```

Orientation lock/unlock

The ACBridge allows the applet to lock itself to the current orientation and overwrite the mobile OS autorotate. To do so, use the functions ACBridge.lockOrientation() and ACBridge.unlockOrientation().

Example

APPLET

```
<html>
<head>
<script>
```

```
	if(orientationLockWanted)
	{
		ACBridge.lockOrientation();
	}
	else
	{
		ACBridge.unlockOrientation();

	}
</script>
</head>
<body>
.
.
.
</body>
.
</html>
```

Saving images to gallery

The ACBridge allows applets to save images to camera roll. To use this functionality, call the function ACBridge.save("filename.png"). On Android pictures will be saved in a specific album called "Arx Control", while on iOS the pictures will be stored in the camera roll.

For questions/comments, email devtechsupport@logitech.com