Swarm Unity Tutorial

This tutorial is used in conjunction with the demo code to help new users understand how to use the Swarm Unity Plugin. (Note: The current version of the Swarm Unity Plugin is only for Android and it will not work on other platforms. Also, since the Swarm Unity Plugin relies on deeper parts of Android, it will only run on an Android device and won't run in the Unity Editor.)

Part 1: Initialization and Setup

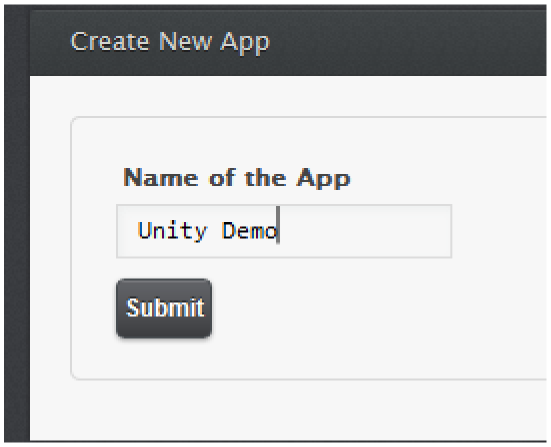
Code file for this section: Demo\Scripts\LoginDemo.cs

When using the Unity Swarm Plugin, the very first thing that you need to do is create your app in the Swarm Admin Panel.

Login in to your Swarm Admin Panel by visiting swarmconnect.com and clicking on the LOGIN button at the top of the screen. You may need to create a new account if you have not already done so. The new account registration process will walk you through setting up your account and creating your first app entry.

Once you login to your Swarm Admin Panel, if you still need to create a new app, you can do so clicking on "Create a New App" from the app list (http://swarmconnect.com/admin/).

Create a new app:



Unity Demo

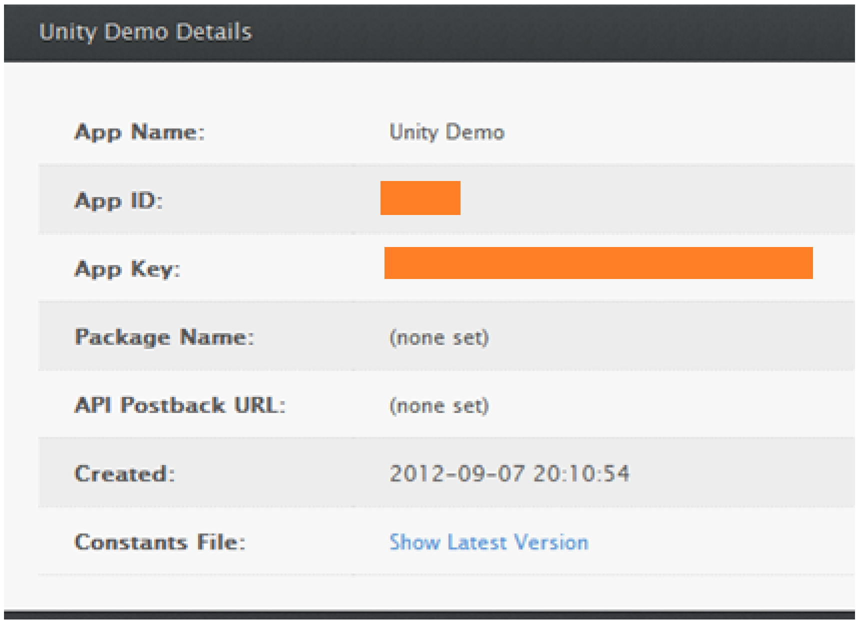
Next you want to initialize Swarm in your code. Typically you should initialize swarm at any entry point into your code:

To initialize Swarm, you need to call init.

Swarm.init (applicationId, applicationKey);

This code initializes Swarm, and to do this we need an application id and an application key.
To find these go back to the Swarm Admin Panel and click on the "App Details" tab.

Document Version 1.1.0



Unity Demo Details

App Name:	Unity Demo
App ID:	[redacted]
App Key:	[redacted]
Package Name:	(none set)
API Postback URL:	(none set)
Created:	2012-09-07 20:10:54
Constants File:	Show Latest Version

Replace applicationId (line 6) and applicationKey (line 7) with the values from **your** Admin Panel.

int applicationId = ***YOUR APP ID HERE****;*
*string applicationKey = "****YOUR APP KEY HERE****";*

After you get the app up on your Admin Panel and get your app initializing Swarm, you can do a number of things. There are four categories of features you can implement: Achievements, Leaderboards, Cloud Data, and Store. You read more about each feature on the Swarm website (http://swarmconnect.com/achievements , http://swarmconnect.com/leaderboards , http://swarmconnect.com/cloud_data , and http://swarmconnect.com/virtual_store).

For now we will move on to setting up Achievements.

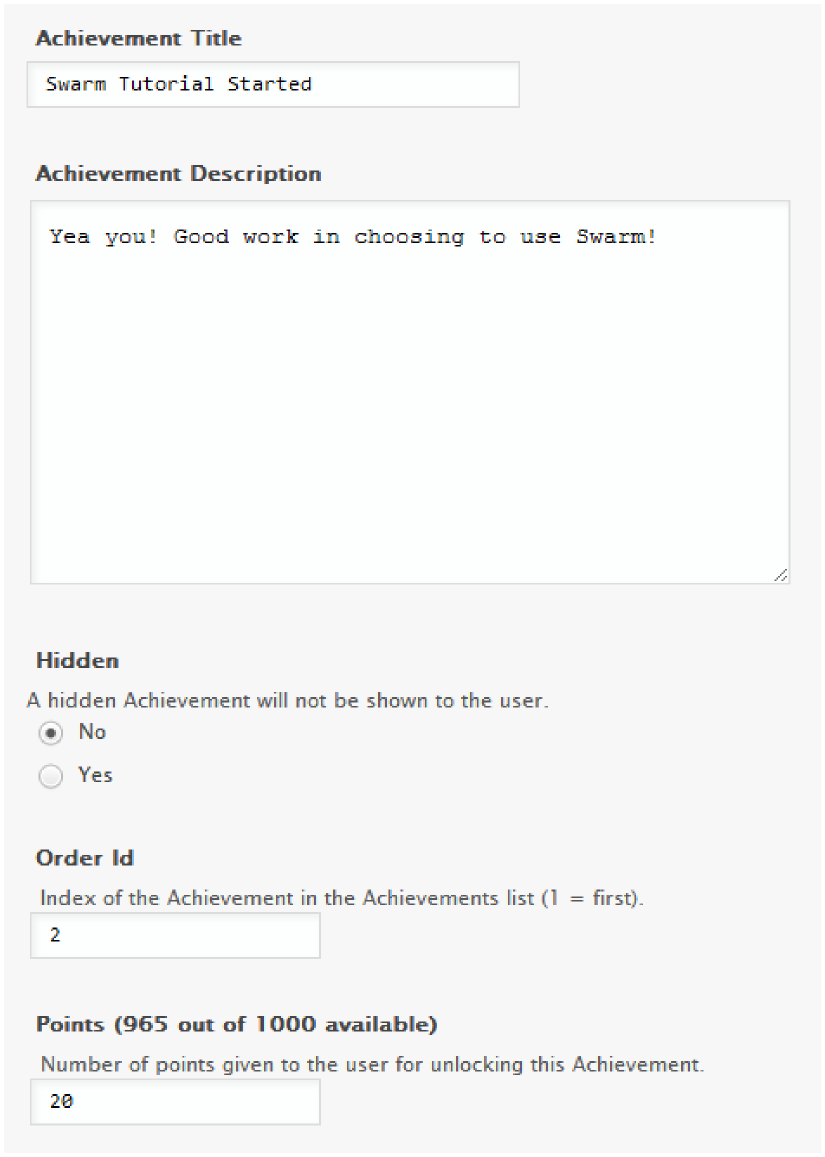
Part 2: Achievements

Code file for this section: Demo\Scripts\AchievmentsDemo.cs

Under the achievements category, there are currently two things you can do: unlock an achievement or show achievements.

Unlock achievements is used to give a player a reward for accomplishing some goal or task in your game.
To do this you need to create some achievements in the Admin Panel to call from the code.

On your Admin Panel click the “Achievements” tab. Create two Achievements, they can be whatever you like, but make sure there are two for this tutorial.



Achievement Title

Swarm Tutorial Started

Achievement Description

Yea you! Good work in choosing to use Swarm!

Hidden

A hidden Achievement will not be shown to the user.

[✓ No] Yes

Order Id

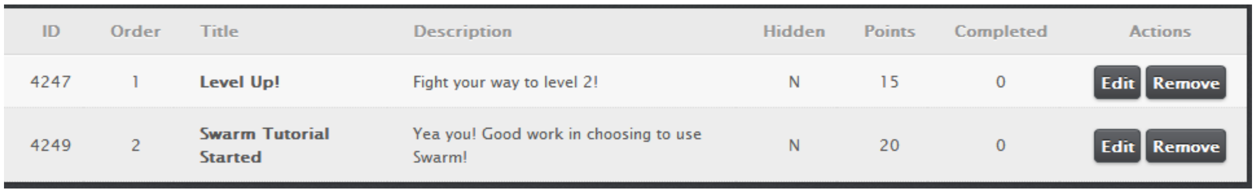
Index of the Achievement in the Achievements list (1 = first).

2

Points (965 out of 1000 available)

Number of points given to the user for unlocking this Achievement.

20



ID	Order	Title	Description	Hidden	Points	Completed	Actions
4247	1	**Level Up!**	Fight your way to level 2!	N	15	0	Edit Remove
4249	2	**Swarm Tutorial Started**	Yea you! Good work in choosing to use Swarm!	N	20	0	Edit Remove

In your code you will now need to call the unlock mechanism on these achievements. The code to do this is:

*SwarmAchievement.unlockAchievement(**YOUR ACHIEVEMENT ID HERE**);*

Do not use achievement ids 4247 and 4249 (they won't work for you), instead **use the ids shown on your Admin Panel**.

Optional step: You can change the names that will display on the Unity GUI buttons, but this is not a necessary step to complete the tutorial.

GUILayout if (GUILayout.Button("Level Up!" ,GUILayout.Height(Screen.height/8))){

Document Version 1.1.0

Upon the first completion of an achievement, a notification will pop up to let the user know they have unlocked an achievement.

You can give many achievements to a player, but you also want them to be able to see the entire list of potential achievements. This is where the show achievements feature comes in.

Call show achievements with this code:

```
Swarm.showAchievments();
```

Great, now you know how to give players achievements and let them view their achievements anytime.

For the next part of the tutorial we will cover Leaderboards.

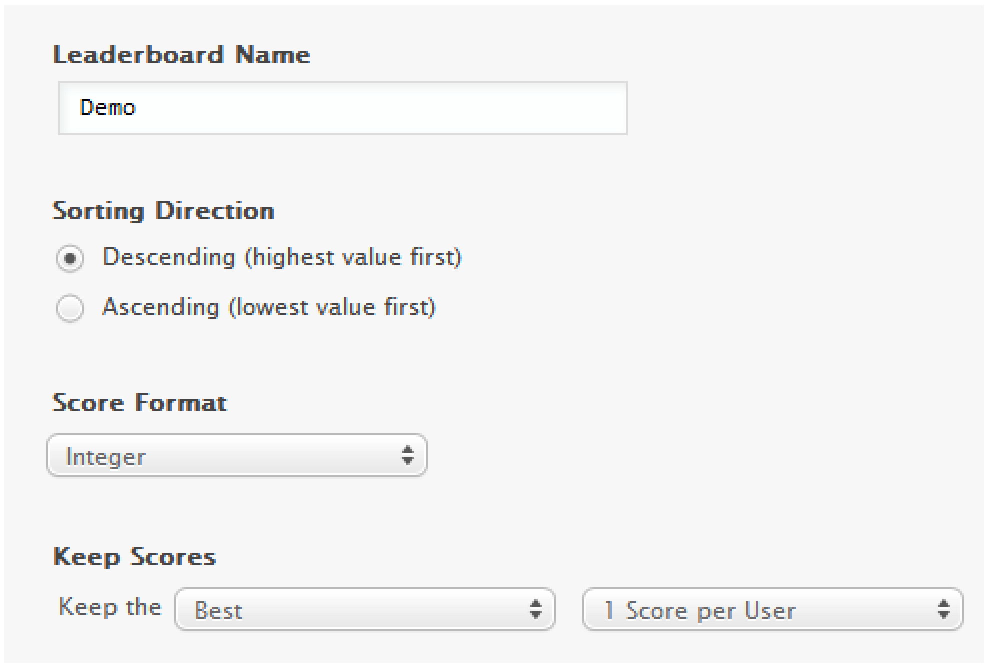
Part 3: Leaderboards

Code file for this section: Demo\Scripts\LeaderboardsDemo.cs

For the leaderboards there are three functions to take note of: submit score, show leaderboard, and show leaderboards.

First, we will go over submitting a score to a leaderboard. Of course, if you haven't already figured it out, we need to create a leaderboard in the Swarm Admin Panel.

Click the "Leaderboards" tab in the admin panel and create a leaderboard.



Leaderboard Name
Demo

Sorting Direction
● Descending (highest value first)
○ Ascending (lowest value first)

Score Format
Integer

Keep Scores
Keep the Best | 1 Score per User

Normally, you would keep track of a score all game and then send it to the leaderboard at the end of a game. That is beyond the scope of this tutorial so we will instead setup two buttons with two different scores.

Get the leaderboard id from the Admin Panel and slap it in to the appropriate variable. Again, **don't use our leaderboard id value of 2169** as it won't work for you. You have to **use the leaderboard id number in your Admin Panel.**

*int leaderboardId = **YOUR LEADERBOARD ID HERE**;*

See that submit score is called twice at line 22 and line 27:

SwarmLeaderboard.submitScore(leaderboardId, 5);
SwarmLeaderboard.submitScore(leaderboardId, 10);

Show leaderboards works in the same fashion as show achievements.

showLeaderboards();

There is also a show Leaderboard (singular) method that will show a specific leaderboard by id rather than all leaderboards available. If you only have a single leaderboard, both show functions will take you to the same screen.

showLeaderboard(leaderboardId);

Yay you! Now you have the ability to setup a leaderboard for your game! Let's move on to the Store functions.

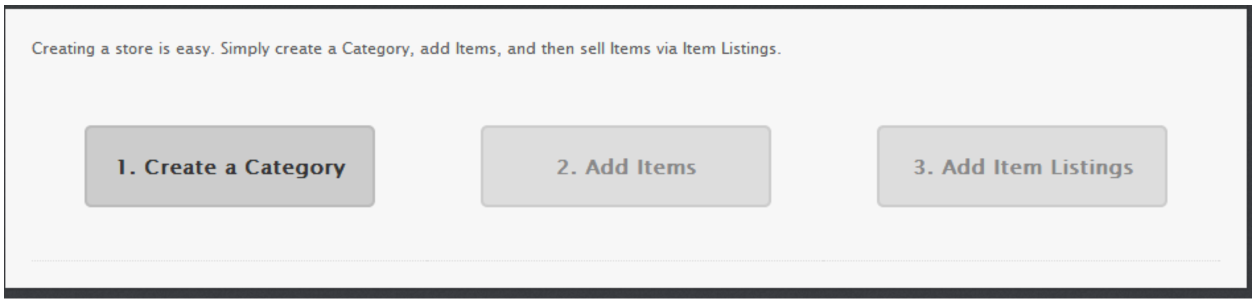
Part 4: Store

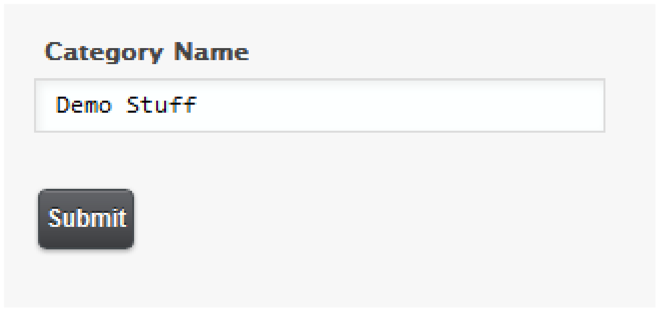
Code file for this section: Demo\Scripts\StoreDemo.cs

For any app these days, sometimes the best way to make money is through micro-transactions. Swarm has built in functionality to make this easier for you.

The features we will use in this section are: get item quantity, purchase item, consume item, and show store.

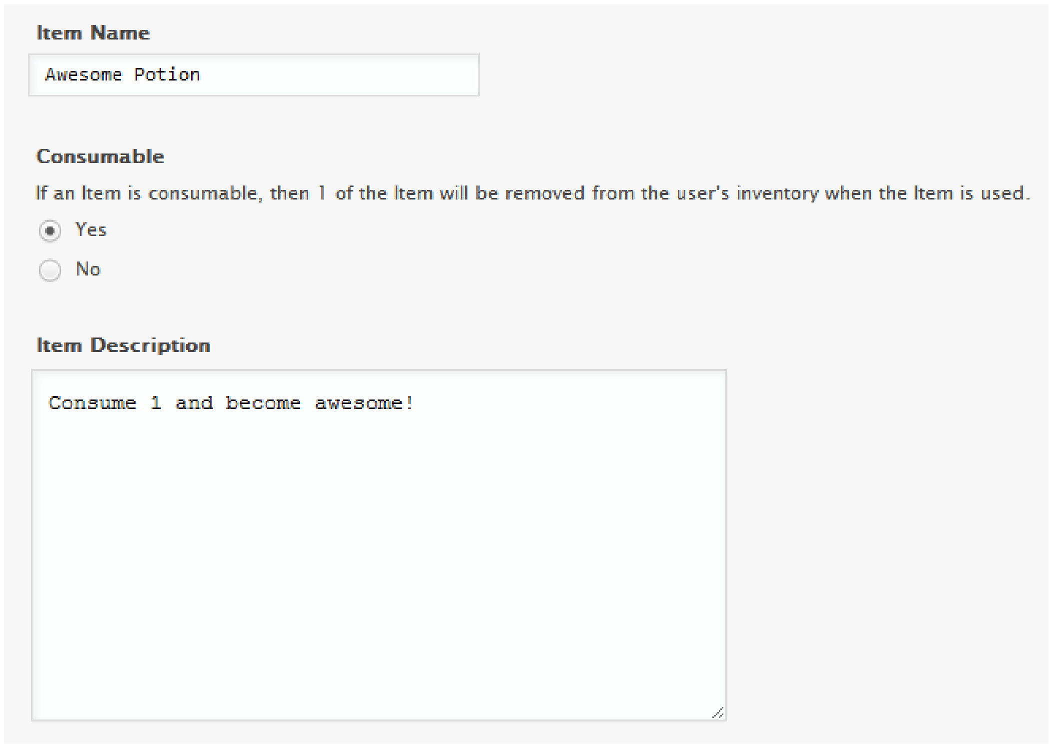
Click the "Store" tab and create a Category (Weapons, Armor, Potions, etc.)


Creating a store is easy. Simply create a Category, add Items, and then sell Items via Item Listings.


Category Name

Demo Stuff

Click the “Store” tab again and create an item.


Item Name

Awesome Potion

Consumable

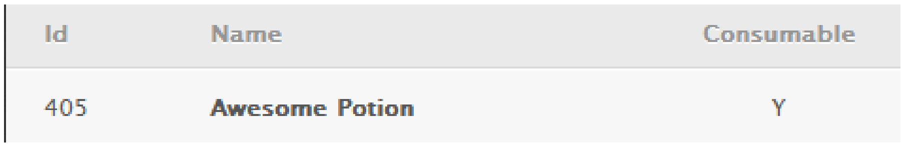
If an Item is consumable, then 1 of the Item will be removed from the user's inventory when the Item is used.

[✓ Yes] No

Item Description

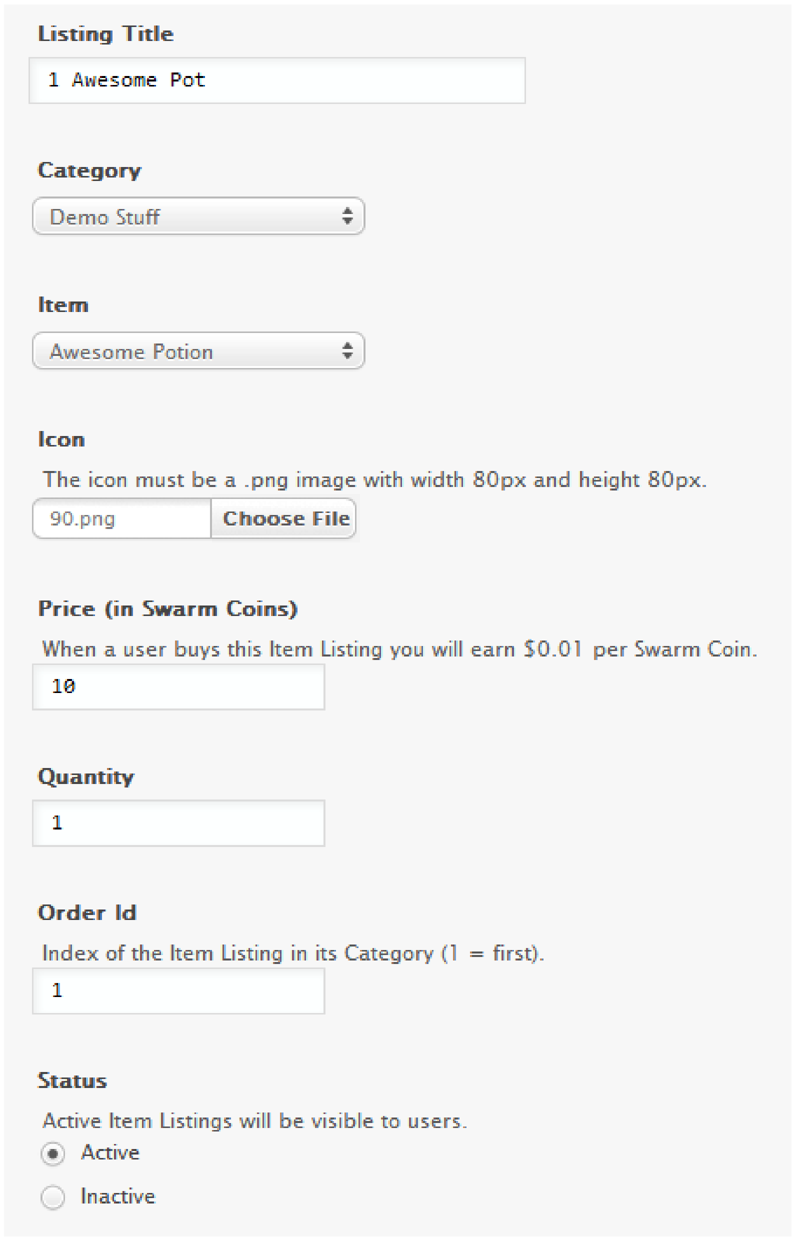
Consume 1 and become awesome!

The image below shows an Item and its id number (**405**).



Id	Name	Consumable
405	**Awesome Potion**	Y

Click the “Store” tab again and create a listing.



Listing Title

1 Awesome Pot

Category

Demo Stuff

Item

Awesome Potion

Icon

The icon must be a .png image with width 80px and height 80px.

90.png | Choose File

Price (in Swarm Coins)

When a user buys this Item Listing you will earn $0.01 per Swarm Coin.

10

Quantity

1

Order Id

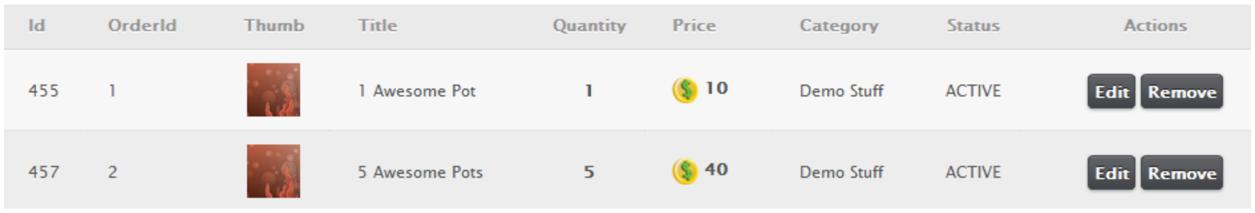
Index of the Item Listing in its Category (1 = first).

1

Status

Active Item Listings will be visible to users.

[✓ Active]
Inactive



Id	OrderId	Thumb	Title	Quantity	Price	Category	Status	Actions
455	1		1 Awesome Pot	1	10	Demo Stuff	ACTIVE	Edit Remove
457	2		5 Awesome Pots	5	40	Demo Stuff	ACTIVE	Edit Remove

As shown, listings can be multiples of items. It is important to **note that both Item Listings and Items have different, unique ID numbers and you’ll have to pay attention to which one you’re using**.

With are store properly set up, let’s move onto some code.

Get item quantity will return the number of a certain item based on the id that you give it. Note that this is NOT the listing id, but the item id. Hover over the store tab and you will see a menu appear. Click on the "Items" choice and find the item id. Stick this id in the code (**don't use item id number 405 as it won't work for you, instead use the value shown on your screen**):

```
SwarmUserInventory.getItemQuantity(YOUR ITEM ID HERE, delegate(int quantity) {
        // Insert your code here
});
```

You may have noticed that, unlike anything we have done to this point there is a delegate parameter. Here's where things get a little more interesting.

This parameter exists because a callback is required in order to use this feature. Essentially, callbacks allow your app to keep running while Swarm does its thing. This means Swarm will not interfere with your game and cause it to stall. Refer to the callback section of the documentation (http://swarmconnect.com/admin/docs/callbacks) for more information on callbacks. Don't worry; it is not as complex as it sounds.

You can do whatever you want within the delegate, the important point is that you MUST have the delegate to gain any information back from the callback! The code inside of the delegate is what gets called when the data returns from the server.

Purchase item, obviously, is used to give the logged in user an item. This allows you to let users buy items in game directly (without having to display the store screen). This feature also requires a callback. **Remember to use your own item listing id number** (instead of 455).

```
SwarmStore.purchaseItemListing(YOUR ITEM LISTING ID HERE, delegate(int statusCode) {
        Debug.Log("Got back in purchaseItem, data: "+ statusCode);
});
```

Again, do what you want inside the method; the tutorial has a Debug.log for simplicity.

Note that the id number this time is the LISTING id. When using purchaseItemListing, always use the listing id, not the item id. Purchasing item listings is convenient, because a single item listing can contain a large quantity of a single item.

Consume item will consume 1 of the items of the given item id. Consuming an item reduces the quantity in the user's inventory by 1. This one takes the item id number. **Remember to use your own item id number** (instead of 405)**.**

```
SwarmUserInventory.consumeItem(YOUR ITEM ID HERE);
```

Finally, we have the usual show method to display the store screen:

```
Swarm.showStore();
```

With the first three of four categories covered, we move on to the final one, Cloud Data.

Part 5: Cloud Data

Code file for this section: Demo\Scripts\CloudDataDemo.cs

For this section there are a couple of key features: getUserData and saveUserData.

The saveUserData method the data to the server in key-value pairs, and getUserData retrieves data from the server based on the key you provide.

Cloud Data for Swarm is basically whatever a storage space to keep data for your users. What data you store is up to you.

There is no setup needed on the Swarm Admin Panel to use the Cloud Data feature.

In the CloudDataDemo.cs file, you'll see code like this:

```
string dataKey = "myKey";
string data = "Data 1";
string data2 = "Data 2"
```

saveUserData is pretty straight forward and is used like this:

```
SwarmActiveUser.saveUserData(dataKey, data);
```

getUserData requires a delegate to handle the callback:

```
SwarmActiveUser.getUserData (dataKey, delegate(string responseData) {
        // Insert your code here
        // responseData is the data string returned to you from Swarm's servers
});
```

Okay, we've covered ever category of feature, we are done right? Not quite, there are still a few miscellaneous functions that we need to go over.

Part 6: Miscellaneous

Code file for this section: Demo\Scripts\LoginDemo.cs and Demo\Scripts\HomeDemo.cs

The last few features we need to cover are: is enabled, is logged in, the login listener, get username, and show dashboard. These do not require any setup in the Admin Panel.

Calling Swarm.isEnabled() will return true upon a user login and false otherwise. Its use would be for checking if Swarm has been enabled (Swarm is enabled if a user has logged in at least one time before (has accepted terms and conditions).

```
if (Swarm.isEnabled() == true) {
        Debug.Log ("Is enabled");
} else {
        Debug.Log ("Is not enabled");
    }
```

Here we just output the result to the debug log because this is not something that the user needs to see, but is useful information for a developer.

Login listener allows you to listen for various types of login and logout steps. This will give you the freedom to decide what to do when one of the following login actions occurs: a login is started, a login is canceled, a user is successful logged in, and a user is logged out. Adding a login listener requires adding a delegate to handle the callback. The status value returned tells you information about what happened during the login.

```
    SwarmLoginManager.addLoginListener(delegate(int status) {

        if (status == SwarmLoginManager.USER_LOGGED_IN) {
            Application.LoadLevel("HomeDemo");
        } else if (status == SwarmLoginManager.LOGIN_STARTED) {
            Debug.Log ("login started!");
        } else if (status == SwarmLoginManager.LOGIN_CANCELED) {
            Debug.Log ("login canceled!");
        } else if (status == SwarmLoginManager.USER_LOGGED_OUT) {
            Debug.Log ("user logged out!");
        }
    });
```

The getUsername method is used in HomeDemo.cs. It returns the username of the currently logged in user.

```
    SwarmActiveUser.getUsername();
```

And finally we arrive at the last method showDashboard. The showDashboard method is used in HomeDemo.cs. It is a standard show function, but will bring you to the full Swarm dashboard which contains all other Swarm sections.

```
    Swarm.showDashboard();
```

And we are finally done with the coding and setup of Swarm in Unity!

Now go build and run the demo app on your Android device and test the features out for yourself.