Bridge : Data - Advanced: More Functions

SHARED DATA / DEVELOPER FUNCTIONS

Al the functions listed outside this document always operated only on the currently logged in player's data.
In this document we explore additional functionalities…

Firstly, you are able to store data that can be accessed by all players. For example: creating game settings. As soon as you make the change it is immediately live for all players so if you make the difficulty for level 3 extra hard, everybody gets that same value the next time your game fetches that value. They don't need to download the binary again to receive your game updates!

The functions are identical to the ones described before so if you have become familiar with their use then the use of these will feel very natural and come with no learning curve. There are only 2 things that set the shared functions apart from the functions that operate on a user's own data:

1. There is no FetchGlobalInfo or FetchEverything functions since that makes no sense
2. The function names have the word "Shared" in them to make their use clear.

- FetchSharedField
- FetchSharedCategory
- FetchSharedCategoryLike
- FetchAllSharedInfo
- RemoveSharedField
- RemoveSharedCategory
- DeleteSharedImage
- UpdateSharedCategory
- StoreSharedImage

AUTHORITATIVE NETWORK

Since it is intended to be used primarily or even exclusively during networked multiplayer games I have dubbed these functions the Authoritative Network functions.

Just like before, the functions are nearly identical to those players use on their own data except not all functions exist. For example, after a match has ended and you need to upload a score to a user… there is no reason why you need to fetch all the info of all his games and all his personal info just to assign him that score… so that function was omitted.

Just like you used "Shared" in the developer functions you now use "User" instead.
Also, add an extra parameter at the end of the function for the id of whoever's data you are updating.

- FetchUserField
- FetchUserCategory
- FetchUserCategoryLike
- FetchUserGameInfo
- FetchUserGlobalInfo
- RemoveUserField
- RemoveUserCategory
- DeleteUserImage
- UpdateUserCategory
- StoreUserImage

IMAGE HANDLING

Images are uploaded individually, not as a collection like categories, but can still be placed under categories just like any other data. Every image that gets uploaded stores 2 things:

1. It saves the image (JPG / PNG) under the user's upload folder under the WordPress default uploads folder
2. It saves an entry into the Data tables listing the absolute path to the image

To fetch back your image inside your game is a two step process:

1. Get the URL using the normal FetchField or FetchCategory functions like normal
2. Start a Coroutine and fetch the image using Unity WebRequest class. Alternatively, use the MBS String extension to fetchit as a Task

   Example:
   Texture2D image;
   async Task FetchImage(string url) => image = await url.DownloadTextureTask();

Because the image is stored on your website as an image, not as encrypted data, it means you can display the image on your website like any other image.