Database Control (Free) Guide

This documentation is for DCF Versions 1.1.x

Note: Upgrading your project from DCF v1.0.x will require you to recreate a database and significant code changes. All database contents will be lost. Make sure you create a backup of your project first.

Contents:

Introduction

What is Database Control (Free)?

Database Control (Free) is a quick and easy solution for an online user account database in your game or app. We give you the ability to create a database and use it for usernames, passwords and another text variable for an unlimited number of accounts in your game/app.

What Platforms does it work on?

It has currently only been tested on Windows Standalone, Unity Web Player, WebGL, and Android builds. We believe it will also work on other platforms (probably all) as long as the user has an internet connection. If you do manage to get it working on other platforms, please let us know, so we can make supported platforms clearer to other users.

Which Language is best to use C# or Js?

Both languages are supported. Once you have setup a database you can look at either of the demo login scenes (there is one for each) and there are further code examples in this PDF if needed.

Can I use one of the demo scenes as a login scene in my published game?

Yes, and feel free to change the UI to suite your game.

Is it Secure?

No. Nothing is encrypted. However, you might take the view 'if somebody wants to cause damage, they will, and encryption won't stop them.' If you want more features and security take a look at Database Control Pro.

Are there any other limitations?

Yes. This is a free package after all. If you want more databases, features and security consider upgrading to Database Control Pro. These are the limitations of Database Control (Free):

- You can only use one database per project.
- It is free, so we cannot provide huge amounts of support, but we will happily answer any questions or help with problems.
- Source code not included.
- No added support for other Asset Store packages/services.
- Highscores not supported.
- Cannot view database contents.

Will it Remain Free?

Yes.

Setup

If you have any problems with registering or logging in please email us and we will try to resolve the problem.

1. Import the package (if you haven't already)
2. Open the editor window. Window > Database Control (Free) Window
3. The editor window should appear and prompt you to create a database for your project. Click 'Create Database' to do this.
4. It should take a few seconds to do this. If it takes longer and the process doesn't seem like it is going to complete, please contact us.
5. Now your database should have been created. You will see the 'Project has been Setup' screen. This is the screen you will see when you next open the window.
6. Then everything should be working correctly. Try the demo scenes and/or the code examples on the next few pages.

Important: When you are testing your game/app and creating accounts, these will be added to your database, so remember to go back into the editor window and empty the database before publishing. You can do this by the following steps:

1. Open the editor window. Window > Database Control (Free) Window
2. You should see the 'Project has been Setup' screen. Click 'Empty Database'.
3. Click 'Yes' on the next screen.
4. Wait for it to complete then press 'Continue' and your database should be empty.

Using C# Code:

Setup your C# Script:

Before you start using Database Control in your script, you will need to import the required DLL files.

To do this, just add the following line to the top of your script.

using DatabaseControl;

Now you can use the code examples over the next few pages within your script for the following Database Control methods:

- DCF.RegisterUser
- DCF.Login
- DCF.GetUserData
- DCF.SetUserData

Each of these represents coroutines in the DCF class of the DatabaseControl.dll.

They have to be coroutines because they use WWW requests to send and receive data. This takes time so they need to use *yield,* which means they have to be coroutines. This means the code to call them and receive the returned value can be a bit messy.

DCF.RegisterUser (C#)

public static IEnumerator RegisterUser (string username, string password, string data)

Parameters:

There are 3 parameters:

- username (string) – The username of the account you are creating
- password (string) – The password of the account you are creating
- data (string) – The default string to be added to the account on creation. E.g. if you are using the data string to represent the level reached, you should use “0” as the user has not passed any levels. It can be left blank as “” if necessary.

Description:

Creates a new account in the database with a username, password and another string, used for storing data.

Returns:

String:	Description:
Success	The account has been successfully created.
UserError	The account has not been created as another account with the same username exits.
UserShort	The submitted username is too short, it should be at least 3 characters.
PassShort	The submitted password is too short, it should be at least 3 characters.
Error	Account not created due to another error. This could be due to a variety of things, such as the database not being setup correctly in the project, or a server-side error, etc. If this happens and you don’t know the cause, please contact us.

Example:

Creates an account with username ‘a’, password ‘b’ and data string ‘c’, when the void CreateAccount is called.

```
using UnityEngine;
using System.Collections;
using DatabaseControl;

public class dcfRegisterAccount: MonoBehaviour {

	public void CreateAccount () {
		StartCoroutine(Register(“a”,”b”,”c”));
	}

	IEnumerator Register(string username, string password, string data) {
		IEnumerator e = DCF.RegisterUser(username, password, data);
		while(e.MoveNext()) {
			yield return e.Current;
		}
		string returned = e.Current as string;
		if (returned == “Success”) {
			//Account Created Successfully
			//Do Stuff
		}
		if (returned == “UserError”) {
			//Account Not Created due to username being used on another account
			//Do Stuff
		}
		if (returned == “UserShort”) {
			//Account Not Created due to username being too short
			//Do Stuff
		}
		if (returned == “PassShort”) {
			//Account Not Created due to password being too short
			//Do Stuff
		}
		if (returned.text == “Error”) {
			//Account Not Created, another error occurred
			//Do Stuff
		}

	}

}
```

DCF.Login (C#)

public static IEnumerator Login (string username, string password)

Parameters:

There are 2 parameters:

- username (string) – The username of the account you are logging in to
- password (string) – The password of the account you are logging in to

Description:

Returns whether a password is correct for logging into the account with the username provided.

Returns:

String	Description
Success	The password was correct.
UserError	The account was not found with the username provided.
PassError	The account was found, but the password was incorrect.
Error	Should not login due to another error. This could be due to a variety of things, such as the database not being setup correctly in the project, or a server-side error, etc. If this happens and you don't know the cause, please contact us.

Common Question:

Q. Why should I use DCF.Login for a player to login to my app when it doesn't return the users data string whereas, DCF.GetUserData does return the data as well as requiring the password?

A. DCF.GetUserData can be used to login. However, DCF.Login returns more information about whether it was the username or password which was incorrect. We recommend using DCF.Login to check the password followed by DCF.GetUserData to receive the player's data.

Example:

Attempts to login to account when loginToAccount is called with submitted username and password as parameters.

```
using UnityEngine;
using System.Collections;
using DatabaseControl;

public class dcfLogin: MonoBehaviour {

	public void loginToAccount (string username, string password) {
		StartCoroutine(Login(username, password));
	}

	IEnumerator Login(string username, string password) {
		IEnumerator e = DCF.Login(username, password);
		while(e.MoveNext()) {
			yield return e.Current;
		}
		string returned = e.Current as string;
		if (returned == "Success") {
			//Account Created Successfully
			//Do Stuff
		}
		if (returned == "UserError") {
			//Username not found in database
			//Do Stuff
		}
		if (returned == "PassError") {
			//Username found but password incorrect
			//Do Stuff
		}
		if (returned == "Error") {
			//Should not login as another error occurred
			//Do Stuff
		}

	}

}
```

DCF.GetUserData (C#)

public static IEnumerator GetUserData (string username, string password)

Parameters:

There are 2 parameters:

- username (string) – The username of the account you are getting data from
- password (string) – The password of the account you are getting data from

Description:

Finds account with username provided, checks the password of the account and if correct returns the player's data string.

Returns:

String	Description
---	Player's data string is returned if username is found and password is correct
Error	Username not found or password incorrect. Or Data not returned due to another error. This could be due to a variety of things, such as the database not being setup correctly in the project, or a server-side error, etc. If this happens and you don't know the cause, please contact us.

Example:

Attempts to receive account data when getAccountData is called.

```
using UnityEngine;
using System.Collections;
using DatabaseControl;

public class dcfGetAccountData : MonoBehaviour {

	public void getAccountData(string username, string password) {
		StartCoroutine(getData(username, password));
	}

	IEnumerator getData(string username, string password) {
		IEnumerator e = DCF.GetUserData(username, password);
		while(e.MoveNext()) {
			yield return e.Current;
		}
		string returned = e.Current as string;
		if (returned == "Error") {
			//Does not give data as another error occurred
			//Do Stuff
		} else {
			//data received in 'returned' variable
			string dataRecieved = returned;
			//Do Stuff
		}
	}

}
```

DCF.SetUserData (C#)

public static IEnumerator SetUserData (string username, string password, string data)

Parameters:

There are 3 parameters:

- username (string) – The username of the account you are logging in to
- password (string) – The password of the account you are logging in to
- data (string) – The data string to be stored on the account (overwrites all previous strings)

Description:

Finds account with username provided, checks the password of the account and if correct sets the data string with the one provided.

Returns:

String	Description
Success	The password was correct and data string was stored.
Error	Username not found or password incorrect. Or Did not set data due to another error. This could be due to a variety of things, such as the database not being setup correctly in the project, or a server-side error, etc. If this happens and you don't know the cause, please contact us.

Example:

Attempts to set account data when setAccountData is called.

```
using UnityEngine;
using System.Collections;
using DatabaseControl;

public class dcfSetAccountData : MonoBehaviour {

	public void setAccountData (string username, string password, string data) {
		StartCoroutine(setData(username, password, data));
	}

	IEnumerator setData(string username, string password, string data) {
		IEnumerator e = DCF.SetUserData(username, password);
		while(e.MoveNext()) {
			yield return e.Current;
		}
		string returned = e.Current as string;
		if (returned == “Success”) {
			//Data set successfully
			//Do Stuff
		}
		if (returned == “Error”) {
			//Did not set data as another error occurred
			//Do Stuff
		}

	}

}
```

Using Js Code:

Setup your Js Script:

Before you start using Database Control in your script, you will need to import the required DLL files.

To do this, just add the following line to the top of your script.

import DatabaseControl;

Now you can use the code examples over the next few pages within your script for the following Database Control methods:

- DCF.RegisterUser
- DCF.Login
- DCF.GetUserData
- DCF.SetUserData

Each of these represents coroutines in the DCF class of the DatabaseControl.dll.

They have to be coroutines because they use WWW requests to send and receive data. This takes time so they need to use *yield,* which means they have to be coroutines. This means the code to call them and receive the returned value can be a bit messy.

DCF.RegisterUser (Js)

public static IEnumerator RegisterUser (username : String, password : String, data : String)

Parameters:

There are 3 parameters:

- username (string) – The username of the account you are creating
- password (string) – The password of the account you are creating
- data (string) – The default string to be added to the account on creation. E.g. if you are using the data string to represent the level reached, you should use “0” as the user has not passed any levels. It can be left blank as “” if necessary.

Description:

Creates a new account in the database with a username, password and another string, used for storing data.

Returns:

String	Description
Success	The account has been successfully created
UserError	The account has not been created as another account with the same username exits.
UserShort	The submitted username is too short, it should be at least 3 characters.
PassShort	The submitted password is too short, it should be at least 3 characters.
Error	Account not created due to another error. This could be due to a variety of things, such as the database not being setup correctly in the project, or a server-side error, etc. If this happens and you don’t know the cause, please contact us.

Example:

Creates an account with username ‘a’, password ‘b’ and data string ‘c’, when the function CreateAccount is called.

```
import DatabaseControl;

#pragma strict

public function CreateAccount () {

	var e = DCF.RegisterUser(“a”, “b”, “c”);
	while(e.MoveNext()) {
		yield e.Current;
	}
	var returned = e.Current;
	if (returned == “Success”) {
		//Account Created Successfully
		//Do Stuff
	}
	if (returned == “UserError”) {
		//Account Not Created due to username being used on another account
		//Do Stuff
	}
	if (returned == “UserShort”) {
		//Account Not Created due to username being too short
		//Do Stuff
	}
	if (returned == “PassShort”) {
		//Account Not Created due to password being too short
		//Do Stuff
	}
	if (returned == “Error”) {
		//Account Not Created, another error occurred
		//Do Stuff
	}
}
```

DCF.Login (Js)

public static IEnumerator Login (username : String, password : String)

Parameters:

There are 2 parameters:

- username (string) – The username of the account you are logging in to
- password (string) – The password of the account you are logging in to

Description:

Returns whether a password is correct for logging into the account with the username provided.

Returns:

String	Description
Success	The password was correct.
UserError	The account was not found with the username provided.
PassError	The account was found, but the password was incorrect.
Error	Should not login due to another error. This could be due to a variety of things, such as the database not being setup correctly in the project, or a server-side error, etc. If this happens and you don't know the cause, please contact us.

Common Question:

Q. Why should I use DCF.Login for a player to login to my app when it doesn't return the users data string whereas, DCF.GetUserData does return the data as well as requiring the password?

A. DCF.GetUserData can be used to login. However, DCF.Login returns more information about whether it was the username or password which was incorrect. We recommend using DCF.Login to check the password followed by DCF.GetUserData to receive the player's data.

Example:

Attempts to login to account when loginToAccount is called with submitted username and password as parameters.

```
import DatabaseControl;

#pragma strict

public function loginToAccount (username : String, password : String) {

	var e = DCF.Login(username, password);
	while(e.MoveNext()) {
		yield e.Current;
	}
	var returned = e.Current;
	if (returned == "Success") {
		//Account Created Successfully
		//Do Stuff
	}
	if (returned == "UserError") {
		//Username not found in database
		//Do Stuff
	}
	if (returned == "PassError") {
		//Username found but password incorrect
		//Do Stuff
	}
	if (returned == "Error") {
		//Should not login as another error occurred
		//Do Stuff
	}
}
```

DCF.GetUserData (Js

public static IEnumerator GetUserData (username : String, password : String)

Parameters:

There are 2 parameters:

- username (string) – The username of the account you are getting data from
- password (string) – The password of the account you are getting data from

Description:

Finds account with username provided, checks the password of the account and if correct returns the player's data string.

Returns:

String	Description
---	Player's data string is returned if username is found and password is correct
Error	Username not found or password incorrect. Or Data not returned due to another error. This could be due to a variety of things, such as the database not being setup correctly in the project, or a server-side error, etc. If this happens and you don't know the cause, please contact us.

Example:

Attempts to receive account data when getAccountData is called.

```
import DatabaseControl;

#pragma strict

public function getAccountData (username : String, password : String) {

	var e = DCF.GetUserData(username, password);
	while(e.MoveNext()) {
		yield e.Current;
	}
	var returned = e.Current;
	if (returned == "Error") {
		//Does not give data as another error occurred
		//Do Stuff
	} else {
		//data received in 'returned' variable
		var dataRecieved = returned;
		//Do Stuff
	}

}
```

DCF.SetUserData (Js)

public static IEnumerator SetUserData (username : String, password : String, data : String)

Parameters:

There are 3 parameters:

- username (string) – The username of the account you are logging in to
- password (string) – The password of the account you are logging in to
- data (string) – The data string to be stored on the account (overwrites all previous strings)

Description:

Finds account with username provided, checks the password of the account and if correct sets the data string with the one provided.

Returns:

String	Description
Success	The password was correct and data string was stored.
Error	Username not found or password incorrect. Or Did not set data due to another error. This could be due to a variety of things, such as the database not being setup correctly in the project, or a server-side error, etc. If this happens and you don't know the cause, please contact us.

Example:

Attempts to set account data when setAccountData is called.

```
import DatabaseControl;

#pragma strict

public function setAccountData (username : String, password : String, data : String) {

	var e = DCF.SetUserData(username, password, data);
	while(e.MoveNext()) {
		yield e.Current;
	}
	var returned = e.Current;
	if (returned == "Success") {
		//Data set successfully
		//Do Stuff
	}
	if (returned == "Error") {
		//Did not set data as another error occurred
		//Do Stuff
	}

}
```

Contact Us:

If you have any problems or questions, the best way to contact us is by emailing us.

You can email us at:
solution_studios@outlook.com

Or you can use the contact form on our website:
http://unitysolutionstudios.weebly.com/contact-us.html

Or send us a message on the forums:
https://forum.unity3d.com/threads/database-control-free-released.334560/

Things to include in your email:

- Detailed information about the problem. Are you receiving any messages, errors or warnings in the console? If yes, please quote them.
- Circumstances, what were you trying to do? Use demo scene? Login? Register? etc.
- Have you moved any folders around?
- What code have you been using if any?
- **Most Importantly:** If you are receiving an error or delay from our server, please copy and paste the contents of the following two files into the email. The files can be found at:
  Assets>Database Control Free>Resources>DCF_RuntimeData.txt
  Assets>Editor Default Resources>Database Control Free>DCF_EditorData.txt
- If you have any platform specific problems or problems from updating Unity please tell us the Unity version you are using when the problems occur.

Thank you.

Other Links:

Asset Store Page:
https://www.assetstore.unity3d.com/en/#%21/content/41337

Unity Forum Post:
https://forum.unity3d.com/threads/database-control-free-released.334560/

Youtube Channel (Contains Setup Video Tutorial):
https://www.youtube.com/channel/UCD-Yj4CFeLTIwdZWhfx9v3Q

Website:
http://unitysolutionstudios.weebly.com/

Database Control Pro Links:

Asset Store Page:
https://www.assetstore.unity3d.com/en/#!/content/68574

Unity Forum Post:
https://forum.unity3d.com/threads/released-database-control-pro.415784/

Website Page:
http://unitysolutionstudios.weebly.com/database-control.html

Online Documentation:
http://www.databasecontrolpro.appspot.com/1/Contents.html

Thank you for using Database Control (Free)

Please take some time to leave a review on the Asset Store Page to help others find and use the Asset.