Database Control (Free) Guide

This is an early version of Database Control (Free) so if you have any problems please tell us

<u>Contents:</u>

Introduction

What is Database Control?

Database Control 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, and WebGL 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.'

Are there any other limitations?

Yes. This is a free package after all. We may release a paid version in the future with unlimited features and encryption etc.

- You are limited to 2 databases.
- 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.

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. If it is your first time, the window will make a request to create an account with us. If it isn’t, the window will still make a request, but to receive your account data.
4. It may take some time for your account to be set up.
5. Now you will need to create a database. You are limited to two databases, but you can use the same databases in multiple projects.
6. Click the button to create a database.
7. After waiting for this, the database information should appear. Click the button to use the database you just created. You can now close the window.
8. 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.

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:

- DC.RegisterUser
- DC.Login
- DC.GetUserData
- DC.SetUserData

Each of these represents coroutines in the DC 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.

DC.RegisterUser (C#)

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

Parameters:

Important – None of these parameters should include the character – as it is used server-side as a separator

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
usernameInUse	The account has not been created as another account with the same username exits.
ContainsUnsupportedSymbol	One or more of the three parameters provided contains a '-' symbol which cannot be used. Account not created
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 dcRegisterAccount: MonoBehaviour {

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

	IEnumerator Register(string username, string password, string data) {
		IEnumerator e = DC.RegisterUser(username, password, data);
		while(e.MoveNext()) {
			yield return e.Current;
		}
		WWW returned = e.Current as WWW;
		if (returned.text == “Success”) {
			//Account Created Successfully
			//Do Stuff
		}
		if (returned.text == “usernameInUse”) {
			//Account Not Created due to username being used on another account
			//Do Stuff
		}

		if (returned.text == “ContainsUnsupportedSymbol”) {
			//Account Not Created as one of the parameters contained a “-“ symbol
			//this will not be returned in this example as parameters were “a” “b” and “c”
			//Do Stuff
		}

		if (returned.text == “Error”) {
			//Account Not Created, another error occurred
			//Do Stuff
		}

	}

}
```

DC.Login (C#)

public static IEnumerator Login (string username, string password)

Parameters:

Important – None of these parameters should include the character – as it is used server-side as a separator

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.
incorrectUser	The account was not found with the username provided.
incorrectPass	The account was found, but the password was incorrect.
ContainsUnsupportedSymbol	One or more of the two parameters provided contains a '-' symbol which cannot be used. Account not created
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 DC.Login for a player to login to my app when it doesn't return the users data string whereas, DC.GetUserData does return the data as well as requiring the password?

A. DC.GetUserData can be used to login. However, DC.Login returns more information about whether it was the username or password which was incorrect. We recommend using DC.Login to check the password followed by DC.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 dcLogin: MonoBehaviour {

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

	IEnumerator Login(string username, string password) {
		IEnumerator e = DC.Login(username, password);
		while(e.MoveNext()) {
			yield return e.Current;
		}
		WWW returned = e.Current as WWW;
		if (returned.text == “Success”) {
			//Account Created Successfully
			//Do Stuff
		}
		if (returned.text == “incorrectUser”) {
			//Username not found in database
			//Do Stuff
		}
		if (returned.text == “incorrectPass”) {
			//Username found but password incorrect
			//Do Stuff
		}
		if (returned.text == “ContainsUnsupportedSymbol”) {
			//One of the parameters contained a “-“ symbol
			//Do Stuff
		}
		if (returned.text == “Error”) {
			//Should not login as another error occurred
			//Do Stuff
		}

	}

}
```

DC.GetUserData (C#)

public static IEnumerator GetUserData (string username, string password)

Parameters:

Important – None of these parameters should include the character – as it is used server-side as a separator

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
ContainsUnsupportedSymbol	One or more of the two parameters provided contains a '-' symbol which cannot be used. Account not created
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 dcGetAccountData : MonoBehaviour {

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

        IEnumerator getData(string username, string password) {
                IEnumerator e = DC.GetUserData(username, password);
                while(e.MoveNext()) {
                        yield return e.Current;
                }
                WWW returned = e.Current as WWW;
                if (returned.text == “Error”) {
                        //Does not give data as another error occurred
                        //Do Stuff
                } else {
                        if (returned.text == “ContainsUnsupportedSymbol”) {
                                //One of the parameters contained a “-“ symbol
                                //Do Stuff
                        } else {
                                //data received in returned.text variable
                                string dataRecieved = returned.text;
                                //Do Stuff
                        }
                }
        }

}
```

DC.SetUserData (C#)

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

Parameters:

Important – None of these parameters should include the character – as it is used server-side as a separator

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
- dataToSet (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.
ContainsUnsupportedSymbol	One or more of the three parameters provided contains a '-' symbol which cannot be used. Account not created
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 dcSetAccountData : 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 = DC.SetUserData(username, password);
                while(e.MoveNext()) {
                        yield return e.Current;
                }
                WWW returned = e.Current as WWW;
                if (returned.text == “Success”) {
                        //Data set successfully
                        //Do Stuff
                }
                if (returned.text == “ContainsUnsupportedSymbol”) {
                        //One of the parameters contained a “-“ symbol
                        //Do Stuff
                }
                if (returned.text == “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:

- DC.RegisterUser
- DC.Login
- DC.GetUserData
- DC.SetUserData

Each of these represents coroutines in the DC 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.

DC.RegisterUser (Js)

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

Parameters:

Important – None of these parameters should include the character – as it is used server-side as a separator

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
usernameInUse	The account has not been created as another account with the same username exits.
ContainsUnsupportedSymbol	One or more of the three parameters provided contains a '-' symbol which cannot be used. Account not created
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.

```
import DatabaseControl;

#pragma strict

public function CreateAccount () {

        var e = DC.RegisterUser(“a”, “b”, “c”);
        while(e.MoveNext()) {
                yield e.Current;
        }
        var returned = e.Current as WWW;
        if (returned.text == “Success”) {
                //Account Created Successfully
                //Do Stuff
        }
        if (returned.text == “usernameInUse”) {
                //Account Not Created due to username being used on another account
                //Do Stuff
        }
        if (returned.text == “ContainsUnsupportedSymbol”) {
                //Account Not Created as one of the parameters contained a “-“ symbol
                //this will not be returned in this example as parameters were “a” “b” and “c”
                //Do Stuff
        }
        if (returned.text == “Error”) {
                //Account Not Created, another error occurred
                //Do Stuff
        }
}
```

DC.Login (Js)

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

Parameters:

Important – None of these parameters should include the character – as it is used server-side as a separator

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.
incorrectUser	The account was not found with the username provided.
incorrectPass	The account was found, but the password was incorrect.
ContainsUnsupportedSymbol	One or more of the two parameters provided contains a '-' symbol which cannot be used. Account not created
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 DC.Login for a player to login to my app when it doesn't return the users data string whereas, DC.GetUserData does return the data as well as requiring the password?

A. DC.GetUserData can be used to login. However, DC.Login returns more information about whether it was the username or password which was incorrect. We recommend using DC.Login to check the password followed by DC.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 = DC.Login(username, password);
        while(e.MoveNext()) {
                yield e.Current;
        }
        var returned = e.Current as WWW;
        if (returned.text == “Success”) {
                //Account Created Successfully
                //Do Stuff
        }
        if (returned.text == “incorrectUser”) {
                //Username not found in database
                //Do Stuff
        }
        if (returned.text == “incorrectPass”) {
                //Username found but password incorrect
                //Do Stuff
        }
        if (returned.text == “ContainsUnsupportedSymbol”) {
                //One of the parameters contained a “-“ symbol
                //Do Stuff
        }
        if (returned.text == “Error”) {
                //Should not login as another error occurred
                //Do Stuff
        }
}
```

DC.GetUserData (Js

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

Parameters:

Important – None of these parameters should include the character – as it is used server-side as a separator

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
ContainsUnsupportedSymbol	One or more of the two parameters provided contains a '-' symbol which cannot be used. Account not created
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 = DC.GetUserData(username, password);
        while(e.MoveNext()) {
                yield e.Current;
        }
        var returned = e.Current as WWW;
        if (returned.text == “Error”) {
                //Does not give data as another error occurred
                //Do Stuff
        } else {
                if (returned.text == “ContainsUnsupportedSymbol”) {
                        //One of the parameters contained a “-“ symbol
                        //Do Stuff
                } else {
                        //data received in returned.text variable
                        var dataRecieved = returned.text;
                        //Do Stuff
                }
        }

}
```

DC.SetUserData (Js)

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

Parameters:

Important – None of these parameters should include the character – as it is used server-side as a separator

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
- dataToSet (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.
ContainsUnsupportedSymbol	One or more of the three parameters provided contains a '-' symbol which cannot be used. Account not created
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 = DC.SetUserData(username, password, data);
        while(e.MoveNext()) {
                yield e.Current;
        }
        var returned = e.Current as WWW;
        if (returned.text == “Success”) {
                //Data set successfully
                //Do Stuff
        }
        if (returned.text == “ContainsUnsupportedSymbol”) {
                //One of the parameters contained a “-“ symbol
                //Do Stuff
        }
        if (returned.text == “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 also email us at:
solution_studios@outlook.com

Or you can use the contact form on our website:
http://solution-studios-for-unity.moonfruit.com/contact-us/4583955951

Things to include in your email:

- The database Id of the database you are using. (This should be displayed in the Editor Window)
- 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 data.txt into your email. The file can be found in: Assets>DatabaseControl>Data>Resources>data.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:

Unity Forum Post:
http://forum.unity3d.com/threads/database-control-free-beta-testing.334560

Quick Setup Video:
https://www.youtube.com/watch?v=STEfELsnYkk

Website:
http://solution-studios-for-unity.moonfruit.com/home/4583955575

Thank you for using Database Control (Free)