

# Database Control Pro Documentation

By Solution Studios

See the latest documentation at:

<http://www.databasecontrolpro.appspot.com/1/Contents.html>

(it is much more organised than this .pdf)

Links:

Website (New):

<http://www.unitysolutionstudios.weebly.com/>

Unity Forum:

<http://forum.unity3d.com/threads/database-control-pro.415784/>

Youtube Channel:

<https://www.youtube.com/channel/UCD-Yj4CFeLTlwdZWWhfx9v3Q>

# Contents:

Thank you for purchasing Database Control Pro! Hopefully the contents of this documentation will explain everything clearly. If you have any problems, contact us at: [solution\\_studios@outlook.com](mailto:solution_studios@outlook.com)

|   |    |
|---|----|
| - <a href="#">Setting Up Database Control Pro</a>       | 3  |
| - <a href="#">Using the Demos</a>                       | 3  |
| - <a href="#">The Demos</a>                             | 5  |
| - <a href="#">The Login Demo</a>                        | 5  |
| - <a href="#">The Third Person Demo</a>                 | 7  |
| - <a href="#">The City Constructor Demo</a>             | 8  |
| - <a href="#">The Chat Demo</a>                         | 10 |
| - <a href="#">Introduction to the Main Windows</a>      | 12 |
| - <a href="#">Setup Window</a>                          | 13 |
| - <a href="#">Viewer Window</a>                         | 15 |
| - <a href="#">Sequencer Window</a>                      | 16 |
| - <a href="#">Using The Setup Window</a>                | 17 |
| - <a href="#">Setting up a project</a>                  | 17 |
| - <a href="#">Creating a Database</a>                   | 17 |
| - <a href="#">Creating a Command Sequence</a>           | 18 |
| - <a href="#">Publishing a Database</a>                 | 18 |
| - <a href="#">Unpublishing a Database</a>               | 19 |
| - <a href="#">Using The Viewer Window</a>               | 19 |
| - <a href="#">Opening a Database</a>                    | 19 |
| - <a href="#">Editing a Database</a>                    | 20 |
| - <a href="#">Updating a Database</a>                   | 20 |
| - <a href="#">Using the Sequencer Window</a>            | 20 |
| - <a href="#">Opening a Command Sequence</a>            | 21 |
| - <a href="#">Planning a Command Sequence</a>           | 21 |
| - <a href="#">Creating a Command Sequence</a>           | 21 |
| - <a href="#">Variables</a>                             | 22 |
| - <a href="#">Using Variables</a>                       | 22 |
| - <a href="#">Variable Types</a>                        | 22 |
| - <a href="#">Input Variables</a>                       | 23 |
| - <a href="#">Other Variables</a>                       | 23 |
| - <a href="#">Built-in Variables</a>                    | 24 |
| - <a href="#">Commands</a>                              | 24 |
| - <a href="#">Get Data</a>                              | 25 |
| - <a href="#">Set Data</a>                              | 26 |
| - <a href="#">Return</a>                                | 27 |
| - <a href="#">If</a>                                    | 27 |
| - <a href="#">For</a>                                   | 29 |
| - <a href="#">Create Var</a>                            | 30 |
| - <a href="#">Set Var Value</a>                         | 30 |
| - <a href="#">Set Var Value Extra</a>                   | 31 |
| - <a href="#">Clear Cell</a>                            | 32 |
| - <a href="#">Clear Row</a>                             | 32 |
| - <a href="#">Clear Col</a>                             | 33 |
| - <a href="#">Del Row</a>                               | 34 |
| - <a href="#">Del Col</a>                               | 34 |
| - <a href="#">Ins Row</a>                               | 35 |
| - <a href="#">Ins Col</a>                               | 36 |
| - <a href="#">Float Ops</a>                             | 36 |
| - <a href="#">Int Ops</a>                               | 38 |
| - <a href="#">String Ops</a>                            | 39 |
| - <a href="#">Comment</a>                               | 40 |
| - <a href="#">Time Diff</a>                             | 40 |
| - <a href="#">Running a Command Sequence at Runtime</a> | 43 |
| - <a href="#">C#</a>                                    | 43 |
| - <a href="#">UnityScript</a>                           | 44 |
| - <a href="#">The End</a>                               | 45 |

# Setting Up Database Control Pro

---

There is a Quick Setup Video tutorial available here:

<https://www.youtube.com/watch?v=M7oDDOigkJI>

In order to use Database Control Pro in your project, you need to set it up first:

1. Open the Setup Window (Top Menu Bar > Window > Database Control Pro > Setup Window)
2. Find your invoice Id. When you bought Database Control Pro, the Asset Store should have sent you an email with your invoice Id as proof of purchase. The email should have had a .pdf attached containing your invoice Id.
3. Enter and submit your invoice Id into the Setup window in the Unity Editor. (If this is the first time you have used Database Control Pro, our server will create an account for you)
4. That's it. It's ready to go!

Note: Keep you invoice Id handy as you will need it every time you use Database Control Pro in a new project.



## Using the Demos

---

Database Control Pro comes with 4 main demos and a few extra's. The main demo scenes have to be setup in the Setup Window in order to use them. The extra demos are mainly for testing Command Sequences and require you to fill in the fields in the inspector to use them.

Setting up the Main Demos: (all screenshots use the Login Demo as an example)

1. Open one of the 4 demo scenes at the path: Assets/Database Control Pro/Main Demo/ ... the demo you want to setup ... / ... the scene file
2. Open the setup window by going to the top menu bar Window>Database Control Pro>Setup Window. Make sure your project is setup and go to the 'Demos' tab:



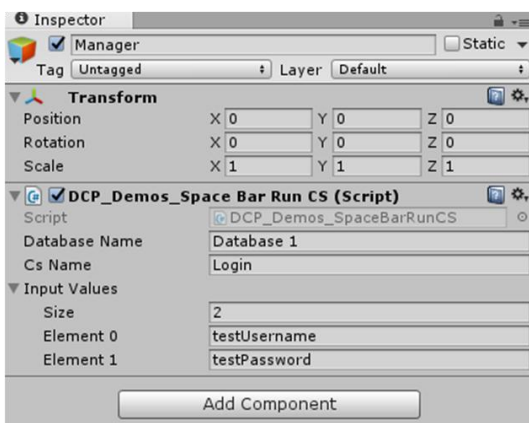
Using the Demos in your Game:

You can use the demo Command Sequences in your game if you wish. You can even modify the Command Sequences in the Sequencer Window. If you want to use a login system in your game we would strongly advise using the Basic Login Demo's Command Sequences and building up from there rather than trying to use the City Constructor Demo's Command Sequences and building down as they are much simpler and easier to follow.

Using the Extra demo scenes:

The extra demo scenes are mainly for testing your Command Sequences. This means they will not work out of the box and need setting up. However with these demos you can use any of your own Databases and Command Sequences so you don't need to use the Setup Window. To use them, setup a Database with one or more Command Sequences and find the 'Manager' GameObject in the scene and enter the Database name and Command Sequence name.

E.g. In the Run Command Sequence on Space bar Demo Scene



If your Command Sequence has Input Variables (e.g. it requires a username and password to be sent to it), fill these in as the string array 'Input Values'. Make sure you fill them in in the same order as they are recieved by the Command Sequence.

# The Demos

---

Database Control Pro comes with 4 main demos and 2 extra demos.

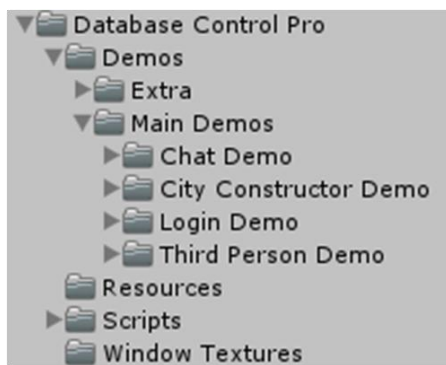
The 4 Main Demos:

- The Login Demo
- The Third Person Demo
- The Chat Demo
- The City Constructor Demo

These demos need to be setup before they can be used.

These Demos can be found in:

Assets>Database Control Pro>Demos>Main Demos



The Final two demos can be used for testing your Command Sequences.

They are both the same demo, except one is C# and the other is Unityscript

More information on how to find and use the extra demos in the Running a Command Sequence at Runtime section.

## The Login Demo

---

The Login Demo is one of the best places to start when using Database Control Pro. It shows a simple Login/Register system where the player can save and load a data string which is saved in the database.

If you want to use a more complex setup with Database Control Pro which is still based on user accounts then we would suggest starting with the Login Demo and building on it.

Make sure you have [setup the demo](#) before you try using it.

Once you have setup the demo, you can delete the 'DeleteMe' object from the scene. (It is a child of the canvas)

The Login Demo can be found at:

Assets>Database Control Pro>Demos>Main Demos>Login Demo>DCP Login Demo

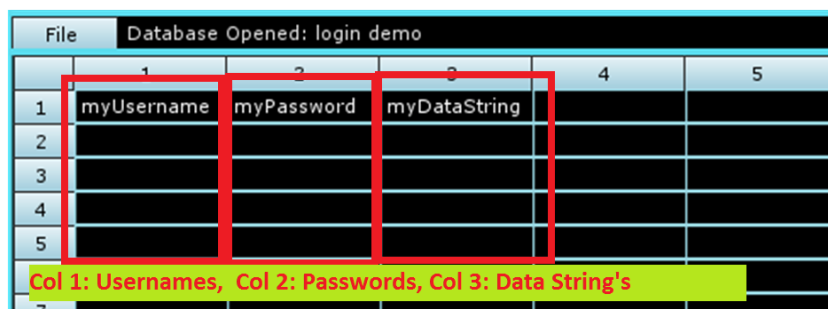
The login demo uses the Smooth UI animation system to fade the UI elements in and out.

All of the code is in a single script called 'LoginDemo\_Accounts' which is attached to the 'AccountsManager' GameObject.

## The Structure of the Database:

First play the demo and register at least one account. Then Open the Setup Window and View the Database which you set the demo up with.

The Database should look something like this:



|   | 1          | 2          | 3            | 4 | 5 |
|---|------------|------------|--------------|---|---|
| 1 | myUsername | myPassword | myDataString |   |   |
| 2 |            |            |              |   |   |
| 3 |            |            |              |   |   |
| 4 |            |            |              |   |   |
| 5 |            |            |              |   |   |
| 6 |            |            |              |   |   |
| 7 |            |            |              |   |   |

Col 1: Usernames, Col 2: Passwords, Col 3: Data String's

The first column of the database stores the player's usernames.

The second column of the database stores the player's passwords.

The third column of the database stores the player's data string.

This means every row of the database represents a single player's account.

## The Command Sequences:

The Login Demo comes with 4 Command Sequences:

1. Login: This sequence looks down the first column of the database (through all of the usernames), and looks for the submitted player username. If it isn't found, 'UserError' is returned. If it is found the 2nd column of the player's row is looked at (the player password), if this is the same as the submitted password 'Success' is returned, otherwise 'PassError' is returned. This sequence is run when the 'Login' button is pressed.

2. Register: This sequence looks down the first column of usernames and makes sure none of the usernames are the same as the submitted username. If they are then 'username in use' is returned as the player has to choose a different username. If it is not in use then 'Success' is returned after adding a new row to the database containing the submitted username and password with 'Hello World!' as the data string. The sequence is run when the player presses 'Register' to show the register UI and 'Register' again to register a new account.

3. Get Data: This sequence is very similar to the 'Login' sequence. The sequence looks down the first column of the database for the player's username. If it is found then it checks the submitted password with the value from the second column of the same row. If the password is correct then the sequence returns the data string which is retrieved from the third column of the row. If the username is not found or the password is not correct, 'Error' is returned. The sequence is run when the player has logged in and the 'Get Data' button is pressed.

4. Set Data: This sequence is also very similar to the 'Login' sequence. The sequence looks down the first column of the database for the player's username. If it is found then it checks the submitted password with the value from the second column of the same row. If the password is correct then the sequence returns 'Success' after setting the value in the third column of the row to the submitted data string. If the username is not found or the password is not correct, 'Error' is returned. The sequence is run when the player has logged in and the 'Set Data' button is pressed.

# The Third Person Demo

The Third Person Demo is very similar to the Login Demo as it has a very simple Login/Register system. The difference is that the player's data is saved (submitted by the game's code) rather than a data string submitted by the player.

The Third Person Demo saves and loads the player's position, and the position of the 5 cubes in the scene and whether they are in the player's inventory or not.

Make sure you have setup the demo before you try using it.

Once you have setup the demo, you can delete the 'DeleteMe' object from the scene. (It is a child of the canvas)

The Third Person Demo can be found at:

Assets>Database Control Pro>Demos>Main Demos>Third Person Demo>DCP Third Person Demo

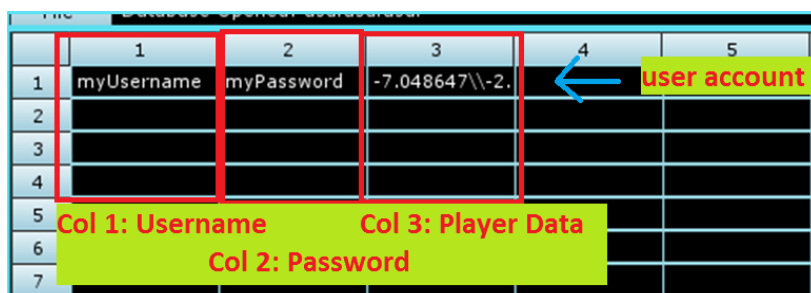
The Third Person Demo uses the Smooth UI animation system to translate the in-game ui menu on and off the screen.

All of the code is split into two groups: the menu code and the in-game code. The menu code handles logging in and registering while controlling the login and register UI, whereas the in-game code handles the save game, load game and inventory code as well as controlling the in-game UI. The 'Transition Manager' object controls which of these two groups is active. E.g. When a player logs in, it disables the menu code (and UI) and enables the in-game code (and UI) by enabling/disabling objects.

## The Structure of the Database:

First play the demo and register at least one account and save the game. Then [Open the Setup Window](#) and [View the Database](#) which you [set the demo up](#) with.

The Database should look something like this:



|   | 1               | 2          | 3                  | 4 | 5 |
|---|-----------------|------------|--------------------|---|---|
| 1 | myUsername      | myPassword | -7.048647\\-2.     |   |   |
| 2 |                 |            |                    |   |   |
| 3 |                 |            |                    |   |   |
| 4 |                 |            |                    |   |   |
| 5 | Col 1: Username |            | Col 3: Player Data |   |   |
| 6 |                 |            | Col 2: Password    |   |   |
| 7 |                 |            |                    |   |   |

The first column of the database stores the player's usernames.

The second column of the database stores the player's passwords.

The third column of the database stores the players data string. The data string combines the player's position, the cubes in the player's inventory and the positions of the cubes not in the player's inventory. The structure of the data string is explained in the comments of the DCP\_TPD\_SaveLoadManager script (line 120 onwards).

This means every row of the database represents a single player's account.

## The Command Sequences:

The Third Person Demo comes with 3 Command Sequences which are very similar to the Login demos sequences (where the Login and Get Data sequences have been combined).

1. Login: This sequence looks down the first column of the database (through all of the usernames), and looks for the submitted player username. If it isn't found, 'UserError' is returned. If it is found the 2nd column of the player's row

is looked at (the player password), if this is the same as the submitted password 'Success' is returned along with the player's data, otherwise 'PassError' is returned. This sequence is run when the 'Login' button is pressed.

2. Register: This sequence looks down the first column of usernames and makes sure none of the usernames are the same as the submitted username. If they are then 'username in use' is returned as the player has to choose a different username. If it is not in use then 'Success' is returned after adding a new row to the database containing the submitted username and password. The data string is left blank as a blank data string tells the game to load the default data with the player and all the cubes in their default positions. The sequence is run when the player presses 'Register' to show the register UI and 'Register' again to register a new account.

3. Save Data: The sequence looks down the first column of the database for the player's username. If it is found then it checks the submitted password with the value from the second column of the same row. If the password is correct then the sequence writes the submitted data string to the third column of the row and returns 'Success'. If the username is not found or the password is not correct, 'Error' is returned. The sequence is run when the player has logged in and the 'Save Game' button is pressed in the in-game menu.

## The City Constructor Demo

---

The City Constructor Demo is based on a more advanced login/register system which also checks email's and IP Addresses to prevent a single user registering more than one account.

It also says the time since the player last logged in, it has a daily reward, and the player can create buildings, upgrade them, move them and sell them. It is the most secure of all of the demos as everything is checked server-side (on the Command Sequences).

As you would expect, it is the most complex system out of all of the demos. If you want a similar system in your game we would still advise building upon the Login Demo. Using this demo might give you a head start, but it might be more difficult for you to understand which slows down your development speed.

Make sure you have setup the demo before you try using it.

Once you have setup the demo, you can delete the 'DeleteMe' object from the scene. (It is a child of the canvas)

The City Constructor Demo can be found at:

Assets>Database Control Pro>Demos>Main Demos>City Constructor Demo>DCP City Constructor Demo

The City Constructor Demo uses the Smooth UI animation system to translate the UI elements on and off the screen. The UI is split up into the Menu UI, the in-game UI and the loading UI in a similar way to the Third Person Demo. All of the code is distributed over multiple scripts and objects, but it should be clear which are the more important ones. As usual, all code is well commented to help explain how the demo works.



## The Structure of the Database:

First play the demo, register an account and create a few buildings. Then Open the Setup Window and View the Database which you set the demo up with.

The Database should look something like this:

|   | 1             | 2            | 3           | 4             | 5   | 6              | 7              | 8 | 9  | 10 | 11 | 12 | 13   | 14 |
|---|---------------|--------------|-------------|---------------|-----|----------------|----------------|---|----|----|----|----|------|----|
| 1 | myUsername    | myEmail@ema  | 123.123.123 | myPassword    | 0   | 2              | 2016           | 7 | 12 | 11 | 56 | 46 | 2016 | 7  |
| 2 |               |              |             | 0             | 1   | $0.5*0.5*-0.5$ | Building Row 1 |   |    |    |    |    |      |    |
| 3 |               |              |             | 0             | 1   | $-3.5*0.6*0$   | Building Row 2 |   |    |    |    |    |      |    |
| 4 |               |              |             | 0             | 1   | $-0.5*0.5*3.5$ | Building Row 3 |   |    |    |    |    |      |    |
| 5 | anotherUserna | anotherEmail | anotherIP   | anotherPasswc | 100 | 1              | 2016           | 6 | 32 | 55 | 55 | 23 |      |    |
| 6 |               |              |             |               |     |                |                |   |    |    |    |    |      |    |

|               |       |            |  |                |                   |                                   |     |     |     |     |     |                                |     |     |
|---------------|-------|------------|--|----------------|-------------------|-----------------------------------|-----|-----|-----|-----|-----|--------------------------------|-----|-----|
| Player Row:   |       |            |  |                |                   |                                   |     |     |     |     |     |                                |     |     |
| Username      | Email | IP Address | Password   | Money          | Level             | Last Request Time                 | ... | ... | ... | ... | ... | Last Claimed Daily Reward Time | ... | ... |
| Building Row: |       |            |  |                |                   |                                   |     |     |     |     |     |                                |     |     |
| Blank         | Blank | Blank      | Building Type (0 for House, 1 for Factory, 2 for Bank) | Building Level | Building Position | (If Factory) Time of Last Collect | ... | ... | ... | ... | ... | ...                            | ... | ... |

Player 1 Account with 3 buildings  
Player 2 Account with 0 buildings

In this demo, every row is not a single account. Instead every account has a main row followed by rows for each building.

The Structure of the Player's main row:

The first column stores the player's username.

The second column stores the player's email.

The third column stores the player's IP Address which they registered the account with.

The fourth column stores the player's password.

The fifth column stores the player's money.

The sixth column stores the player's level.

Columns 7,8,9,10,11 and 12 store the time since the player last ran a sequence (used to calculate time after player logs in) in the form Year, Month, Day, Hour, Minute and Second (Milliseconds ignored).

Columns 13, 14 and 15 store the time the player last claimed the daily reward (blank if never claimed) in the form Year, Month and Day.

The 16th column stores the number of buildings the player has so the Command Sequence know how many lines to loop through, bellow the player's line in order to go through all of the player's buildings.

The Structure of one of the Player's building rows:

The first three columns are blank so the 'Register' Command Sequence can easily loop through player usernames, email addresses and IP addresses to make sure a registering player has different values for these.

The fourth column stores the building type. 0 for House, 1 for Factory and 2 for Bank.

The fifth column stores the building's level number.

The sixth column stores the building's position in the scene.

If it is a factory, columns 7, 8, 9, 10, 11, 12 and 13 store the time when the factories money was last collected in the form Year, Month, Day, Hour, Minute, Second and Millisecond.

## The Command Sequences:

The City Constructor Demo comes with 9 Command Sequences. All of the sequences check everything server-side (except the position of the buildings). These sequences are explained in more detail when you open the sequence and look at the 'Readme' Command.

1. Register: This sequence checks the submitted username is not already in use. If it is then an error is returned. If it is not the the IP Address and submitted email address are also compared with the other values in the database. If they are already in the database an error is returned for the game to say 'You already have an account!', otherwise

the registration is successful and a new row is added to the database with the player's details.

2. Login: This sequence checks the submitted username and password. If they are correct then the player data string is put together and returned. This means working out the time since the last request and the time since the last claimed daily reward (to tell if one can be claimed now or not) and combining it with the player's level and money. Then the sequence loops through all of the building rows adding on the building data.

3. Build: This sequence is run when the player wants to create a new building. The amount of money is retrieved from the player's main row in the database and if there is enough money, a new row is inserted into the database for the created building.

4. Upgrade: This sequence is run when the player wants to upgrade an existing building. The amount of money needed to upgrade is calculated in the sequence and the player's money is checked to make sure there is enough. Other things are also checked including the player level (to make sure it is high enough). If all of these conditions are satisfied, Success is returned and the level number on the building's row in the database is increased.

5. Move: This is the simplest of all the sequences as the position of the buildings is not checked in any sequences as it is less important. Instead the building number is checked to make sure it is a valid building. If it is then the building position string is replaced and Success is returned.

6. Sell: This sequence makes sure the building is valid, adds up all of the upgrade costs of the building to find the total amount ever spent on it. This value is halved to give the sell price. If all conditions are satisfied, the building's row is deleted from the database, and Success is returned while increasing the player's money (and checking it doesn't exceed the maximum amount).

7. Collect: This sequence checks the submitted building number corresponds to a factory. It finds the difference in time since the building's money was last collected and works out the amount it must have generated from this while limiting it based on the factory's level. It then updates the time when it was last collected to now and adds the generated money to the player's money while checking the player's money does not exceed the maximum amount.

8. Level Up: This sequence is run by the game's code after building or upgrading to check for a level up. The sequence iterates through the player buildings, checking for houses. It uses the level of the houses to work out the number of people they contain. This allows it to calculate the total population of the city. It uses the player's current level number to work out the population required to level up. If the population is high enough it returns Success and adds 1 to the player's level.

9. Claim Daily Reward: This sequence works out the time since the Daily Reward was last claimed. If it was a different day, the date in the database is updated, the player money is increased by 100, the player money is limited to its maximum (if required) and 'Success' is returned.

## The Chat Demo

---

The Chat Demo is different to the other demos as it is not based on user accounts. Instead the Chat Demo provides an example of an online chat which anyone can use with any username.

The Chat Demo allows a player to see the online chat which is taking place and post their own comments. The chat is saved to the database so even if the player closed the game and reopened it, they could carry on from where they left and would still be able to see all their previous comments.

This demo could be combined with a multiplayer networking chat (e.g. Photon's Chat) to create a more responsive chat system which remembers all posts.

Make sure you have setup the demo before you try using it.

Once you have setup the demo, you can delete the 'DeleteMe' object from the scene. (It is a child of the canvas)

The Chat Demo can be found at:

Assets>Database Control Pro>Demos>Main Demos>Chat Demo>DCP Chat Demo

The Chat Demo uses the Smooth UI animation system to fade the UI elements in and out. It also makes use of some Unity UI elements for automatic UI layouts.

The UI is split up into the Chat UI, the Choose Name UI, and the loading UI (shown when transitioning between the other two). Nearly all the code is in the DCP\_CD\_MainControl script attached to the 'MainControl' GameObject.

### The Structure of the Database:

First play the demo and post a few comments preferably with different usernames. Then [Open the Setup Window](#) and [View the Database](#) which you [set the demo up](#) with.

The Database should look something like this:

| File Database Opened: Chat Demo |                  |              |             |   |   |   |
|---------------------------------|------------------|--------------|-------------|---|---|---|
|                                 | 1                | 2            | 3           | 4 | 5 | 6 |
| 1                               | user1            | comment1     | 1st Comment |   |   |   |
| 2                               | user2            | comment2     | 2nd Comment |   |   |   |
| 3                               | user3            | comment 3    | 3rd Comment |   |   |   |
| 4                               | john             | hello world! | 4th Comment |   |   |   |
| 5                               | Col 1: Usernames |              |             |   |   |   |
| 6                               | Col 2: Comments  |              |             |   |   |   |
| 7                               |                  |              |             |   |   |   |
| 8                               |                  |              |             |   |   |   |

The first column of the database stores the usernames used to make each comment.

The second column of the database stores the text which makes up each comment.

This means every row of the database represents a comment. The comments at the top were the first to be posted and the comments at the bottom were posted more recently.

### The Command Sequences:

The Chat Demo comes with 2 Command Sequences.

1. Get Chat Data: This sequence gets all of the data from the first and second columns and returns it to the game to show to the player. This sequence is run when the player has submitted a username to join the chat and when the chat is refreshing.

2. Post Comment: This sequence adds a new comment to the database. It adds the submitted username and submitted comment text to the first and second columns of the next row in the database. It is run when the player submits a new comment in the chat.

# Introduction to the Main Windows

There is a Basic Introduction Tutorial Video available here:

<https://www.youtube.com/watch?v=c9sjxnJtI9A>

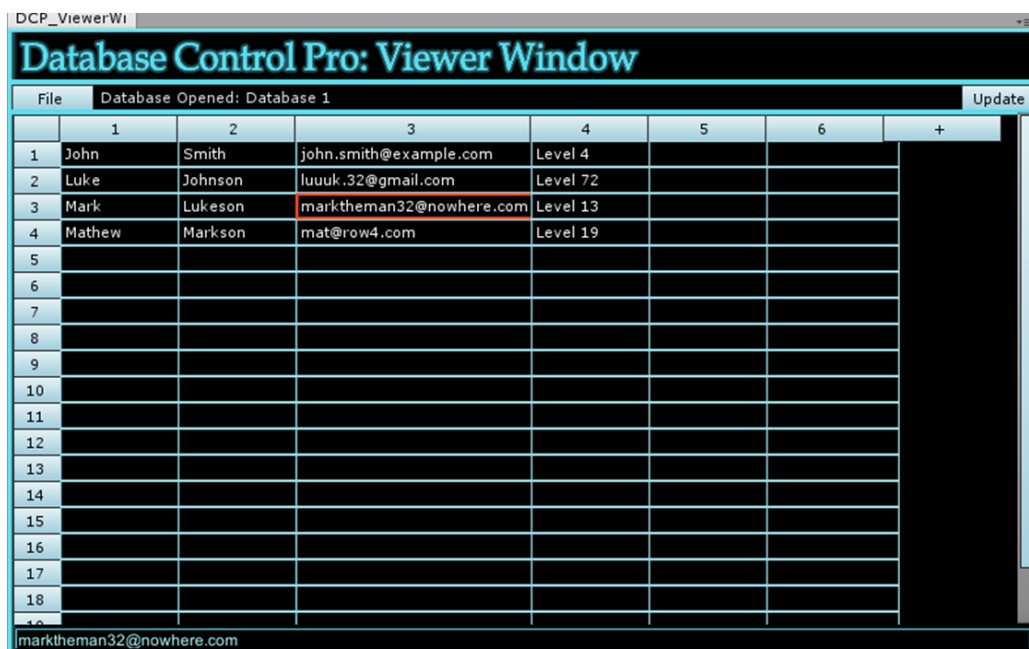
There are three main windows which you need to know how to use:

1. The Setup Window
2. The Viewer Window
3. The Sequencer Window

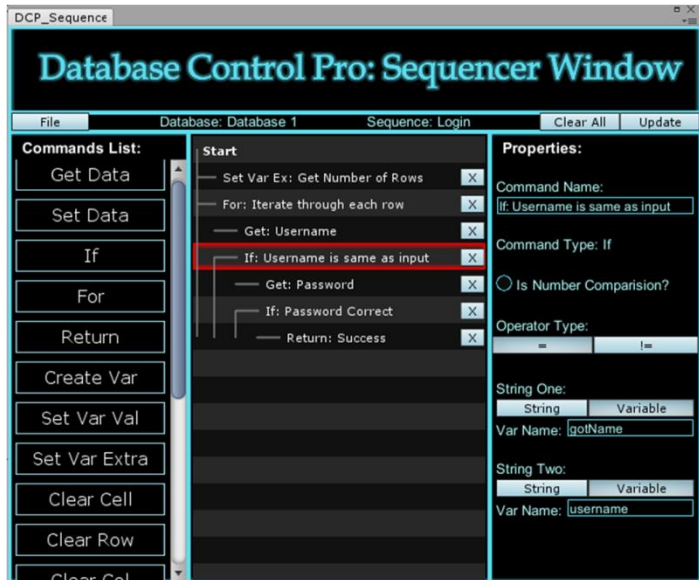
1. The Setup Window can be used to create/rename/delete databases associated with your account. It should also be used to create/rename/delete Command Sequences for each database, as well as publishing/unpublishing databases and setting up databases with Command Sequences for the included demo scenes to work.



2. The Viewer Window can be used to view/edit the contents of any of your unpublished databases. When a database is loaded in this window, it looks similar to a excel spreadsheet.



3. The Sequencer Window can be used to view/edit the Command Sequences of an unpublished database. A Command Sequence is a list of commands which our server will follow to deliver a response when a request is made. Basically, if a player wants to login/register etc, it tells our server what you want it to do in more detail. It acts like a visual scripting system (which works in a similar way to RAIN AI's behaviour tree editor available on the Asset Store). This is the window you will spend the most time using.



## The Setup Window



Opening the Setup Window:

On the top menu bar go to Window>Database Control Pro>Setup Window

Setting up Database Control Pro:

The Setup Window should be used to setup Database Control Pro in your Unity Project.

### Creating Databases:

A database is where all of your online game data will be stored. The Setup Window can be used to create as many databases as you like. Every database must have a unique name to identify it and the setup window allows you to create, rename and delete databases. The databases can also be opened within the Unity Editor through the Setup Window. This will allow you to view and edit them in the Database Viewer Window.

### Creating Command Sequences:

A Command Sequence is a list of instructions which our server will run whenever a request is made to run it in-game. They are the most important part of making a game using Database Control Pro. Every database has its own set of Command Sequences. Command Sequences can be created in the setup window with a database selected. Command Sequences must all have unique names within a single database to identify them. The Setup Window can create, rename and delete Command Sequences. You can also open Command Sequences through the Setup window and they will open in the Sequencer Window.

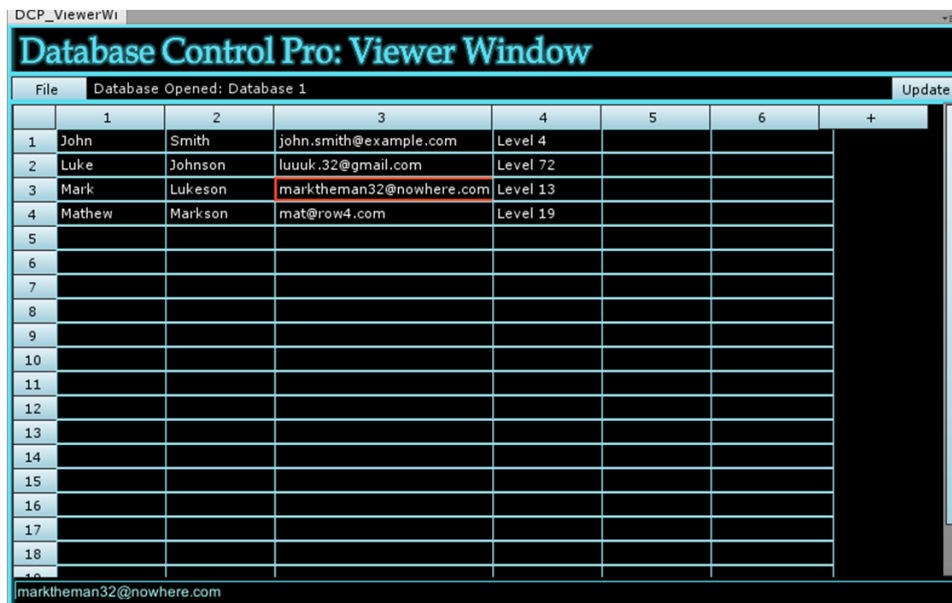
### The Demos:

Database Control Pro comes with some demos which need to be setup before use. The Setup window allows you to setup the demos. All this does is create a database on your account (or use an existing one) and copy pre-made Command Sequences onto the database. It also adds a link into the demo scene so the scripts know which database the Command Sequences are on. All databases created for the demos are in an unpublished state.

### Publish/Unpublish Databases:

When a database is created it is always in an unpublished state. This means security is slightly weaker as you (as the developer) can view and edit the database. You can also create and edit Command Sequences and theoretically so could anybody else if they managed to find out your details such as your invoice number. When you publish your game, we would advise you to publish your databases in the Setup Window. This means our server will not accept any edits to the database or its command sequences using the Unity Editor tools. When you publish a database you will require a password. It is very important that you remember the password you use. For example, if you want to edit the Command Sequences or view the content in the published database, you can use the password to unpublish the database (still within the Setup Window). Unpublishing a database gives you control over the database again in the Unity Editor. If you need to release an update for your game, you can unpublish the database, make your changes, and publish the database again with the same or a different password.

# The Viewer Window



## Opening the Viewer Window:

The Database Viewer Window can be opened in two different ways. Use whichever you find more convenient.

1. On the top menu bar go to Window>Database Control Pro>Viewer Window, then you can select the database you want to open from the list.
2. Open the Setup Window: Window>Database Control Pro>Setup Window, and click on 'View/Edit' next to the database you want to open.

In order to open a database you need to have Setup Database Control Pro and have created a database in the Setup Window.

## Viewing a Database:

When a database is opened in the window it will look similar to an excel spreadsheet. If you cannot see all of the data in a cell then you can click on the cell which will expand it. The contents of the cell is also displayed in a text field at the bottom of the window used for editing.

Note: The Database Viewer window was made for testing Command Sequences when developing your game and to help you visualise the structure of your databases. It was not made to display very large databases so it might cause problems when displaying large amounts of data. We aim to improve this in future updates.

## Editing a Database:

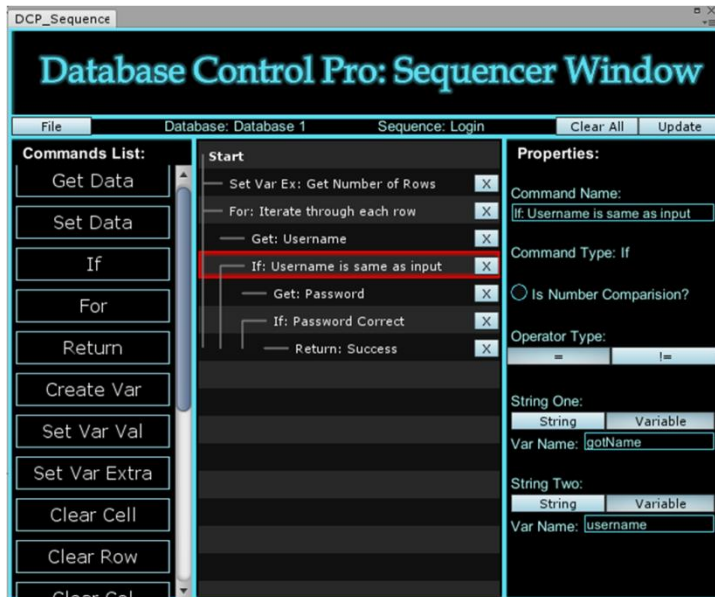
Only a copy of the database is made and displayed. Therefore in order to edit a database you can click on a cell and enter some text, however this will only edit the local copy. In order to apply the changes you need to press the 'Update' button in the top right corner of the window.

## Other Features:

- More than one Viewer Window can be open at any one time. But, two Viewer Windows cannot have the same database open.
- You can press the return key to select the cell bellow. Most of the time this will require two presses of the return key due to the responsiveness of the event in the Unity Editor.
- You can press the row/column numbers to Insert/Duplicate/Delete/Clear rows/columns.
- You can press the blank button in the top left of the database (aligned with the row and column numbers) to clear/delete all cells
- You can press the 'File' button to open a different database in the window or close the window.
- You can press the '+' buttons to add more columns/rows (Note: You do not need to do this in Command Sequences)

where you can 'Get Data'/'Set Data' outside the range of filled cells. This is only done in the Viewer Window so fewer cells need to be drawn on the screen improving performance)

## The Sequencer Window



### Opening the Sequencer Window:

The Command Sequencer Window can be opened in two different ways. Use whichever you find more convenient.

1. On the top menu bar go to Window>Database Control Pro>Sequencer Window, then you can select the Database and Command Sequence you want to open from the list.
2. Open the Setup Window: Window>Database Control Pro>Setup Window, and click on 'Sequences' next to the database the Command Sequence is on and click 'View/Edit' next to the Command Sequence you want to open.

In order to open/edit a Command Sequence you need to have setup Database Control Pro, Created a Database, and Created a Command Sequence.

### The Layout of the Window:

Along the top (from left to right) the file button allows you to close the window or open another Command Sequence. The names of the database and Command Sequence are displayed, the 'Clear All' button can be used to remove all commands from the sequence and the 'Update' button sends the sequence to our server to update it. On the left-hand side of the window there is a list of the different Commands available to use in the sequence. In the centre of the window is the list of Commands.(like the hierarchy window in Unity) The right hand side of the window shows the properties for the selected Command (like the inspector window in Unity). When the 'Start' Command is selected it allows you to create variables for the sequence.

### Window Shortcuts:

- Left clicking on a Command in the left panel adds it to the bottom of the sequence.
- Right clicking on a Command in the left panel allows you to drag and drop the Command to anywhere in the sequence.
- Left clicking on a Command in the centre view selects the Command.
- Right clicking on a Command in the centre view allows you to move the Command by drag and drop.
- You can use the Up and Down arrow keys to scroll the centre scroll view up and down. This makes it easier to move Commands by drag and drop from the very top of the sequence to the very bottom.

Note: Whenever you make a change to a Command Sequence it must be updated using the 'Update' button.



## Using the Setup Window

---

The DCP Setup Window has the following uses:

- Setting Up a Project
- Creating a Database
- Creating a Command Sequence
- Publishing a Database
- Unpublishing a Database

## Setting up a Project

---

The Setup Window must be used to Setup your project before you can use Database Control Pro.

First you need to open the Setup Window, go to the Top Menu Bar, Database Control Pro, Setup Window.

Next you need to find and enter your invoice Id and hit 'Setup'.

The setup window should now be checking your invoice Id and retrieving information about your account from our server. If it is your first time, it will be creating an account for you. After this short wait, Database Control Pro should be setup in your project. If you have any problems please contact us.

Your account data will be stored in your project and checked with every request made for security. But, when you build your project, your invoice Id will not be included, because if anyone found it they could gain access to your Database Control Pro account.

## Creating a Database

---

The Database Control Pro Setup Window can be used to create databases. Open the window at:  
Top Menu Bar, Window, Database Control Pro, Setup Window

If you have Setup your Project you will see a list of your existing databases (if there are any).

Either click the 'Create Database' at the bottom of the list or the '+' tab. Then enter a name for your database in the 'Database Name:' field. The name must be different to the names of your other database. Something including your project name might be a good name.

Then click 'Create' and wait for the window to finish loading. A Message will appear if the database has been created successfully.

In order to use your database, you might want to edit the database in the Unity Editor. To do this press 'View/Edit' next to your database in the list of databases.

You might also want to add a Command Sequence which connect your game to the database through our server running a list of Commands which you have created while developing your game. To do this you will need to click 'Sequences' next to your database's name.

## Creating a Command Sequence

---

The Database Control Pro Setup Window can be used to create Command Sequences. Open the window at: Top Menu Bar, Window, Database Control Pro, Setup Window

If you have Setup your Project you will see a list of your existing databases (if there are any). Every Command Sequence is assigned to a database. You will need to have created a database to continue.

Now choose a database from the list and press the 'Sequences' button next to it. This will show you a list of the database's Command Sequences. To create a new Command Sequence, look for the 'Create New Sequence:' at the bottom of the window. Fill in the 'Name:' field with a suitable name for your Command Sequence (it must be different to the other Command Sequences in the same database) and press 'Create'. Wait for it to finish loading and you should see a message stating that the Command Sequence has been created.

Now you will probably want to add some functionality to your Command Sequence. Next to your Command Sequence's name press 'View/Edit', this will open the DCP Sequencer Window.

## Publishing a Database

---

The Database Control Pro Setup Window can be used to publish Databases. Open the window at: Top Menu Bar, Window, Database Control Pro, Setup Window

If you have Setup your Project you will see a list of your existing databases (if there are any).

By this point you have probably added Command Sequences to your database and you are want to publish the database as an extra security measure before you release your game. Publishing a database means you cannot view/edit your database or add, remove or edit Command Sequences. It means you cannot do anything with the database in the Unity Editor apart from Unpublish it to reverse the publishing process. We recommend all databases are published when the game is released.

In the Setup window, go onto the 'Publish' tab to manage published/unpublished databases. At the top you will see a list of your databases which are unpublished (not published) click the 'Publish' button next to one to select it.

To publish the database you will need to provide a password (which you make up, it can be anything).

Note: The stronger the password the better, but make sure you REMEMBER the password, as you will need it if you ever want to unpublish the database and it is not stored by Database Control Pro.

Enter your chosen password into the field which has appeared next to your database's name. Click 'Publish' to publish the database. If it was successful a Message will be displayed.

If you want to update your game or view/edit the database or Command Sequences then you will want to unpublish your database.

## Unpublishing a Database

---

The Database Control Pro Setup Window can be used to unpublish your database. You need to have a database published to do this. It is a very similar procedure to publishing a database.

Open the Setup Window at:

Top Menu Bar, Window, Database Control Pro, Setup Window

Go to the 'Publish' tab and look for the list of published databases (the second list down). Then find the database in the list which you want to unpublish and click 'Unpublish' next to it. Enter the password which you used to publishing the database and click 'Unpublish' again.

If the password was correct a Message will be displayed to say the database has successfully been unpublished.

## Using the Viewer Window

---

Here are some explanations of how to use the DCP Viewer Window:

Opening a Database

Editing a Database

Updating a Database

## Opening a Database

---

In order to open a database you must open the Database Control Pro Viewer Window. There are two ways of doing this:

Method 1:

Open the Database Control Pro Setup Window, look for the database you want to open in the list of databases and click 'View/Edit' next to the database name.

Method 2:

Open the Database Control Pro Viewer Window by going to the Top Menu Bar, Window, Database Control Pro, Viewer Window. Then select the database you want to open from the list of databases.

Note: The Viewer Window in its current state is not very good for opening large databases. We aim to improve this functionality in a future update.

## Editing a Database

---

To open a database you will need to open it in the Viewer Window. There are two ways of doing this described in the previous section.

Now you should be able to see your database. If the database is quite small use the '+' buttons to add more columns and rows. This will not change the database in any way as the added cells are all empty and will be ignored and not updated to the database.

Note: Any changes you make using the methods bellow will only effect a local copy of the database on your computer. To update the online version of the database you need to press the 'Update' button in the top right of the window which appears when the database has been changed.

Important: Seriously, don't forget the point above. After spending time editing the database, it is so easy to just close the window forgetting to apply the changes.

You can change a cell value in a similar way to Microsoft Excel. Click on the cell and there is a text field at the very bottom of the window where you can enter the cells value. Once you have selected this text field once it should remain selected as you click on other cells to edit them.

Shortcut: With a cell selected you can press the Return key to select the cell bellow. In the Unity Editor this shortcut isn't very responsive so you may have to press the key twice.

To clear all cell values or delete all cells from the database press the black button which is in line with the row numbers and column numbers. This will show you a context menu allowing you to clear/delete all cells.

To perform Row or Column operations such as Insert, Clear and Delete, press the button which displays the row/column number of the row/column you want to perform the operation on.

## Updating a Database

---

When using the Viewer Window make sure you remember to Update the database it is very easy to forget:

Any changes you make using the methods bellow will only effect a local copy of the database on your computer. To update the online version of the database you need to press the 'Update' button in the top right of the window which appears when the database has been changed.

## Using the Sequencer Window

---

These are the important parts of the DCP Sequencer Window:

- Opening a Command Sequence
- Planning a Command Sequence
- Creating a Command Sequence
- Variables
- Commands
- How Commands Handle Errors
- Command Sequence Efficiency

## Opening a Command Sequence

---

To Open a Command Sequence you need to open it in the Sequencer Window. This can be done in two ways:

### Method 1:

Open the DCP Setup Window at the Top Menu Bar, Window, Database Control Pro, Setup Window. Find the database in the list which has the Command Sequence attached to it. Click the 'Sequences' button next to the Database's name. Then you should see a list of the Command Sequences in that database. Click the 'View/Edit' button next to the Command Sequence you want to open and it should open it in the Sequencer Window.

### Method 2:

Open the DCP Sequencer Window directly at the Top Menu Bar, Window, Database Control Pro, Sequencer Window. In the central panel you should see a list of your databases. Select the database with the Command Sequence attached, then press the button for the Command Sequence you want to edit in the panel on the right. This should open the sequence.

## Planning a Command Sequence

---

Before you start creating your Command Sequence, you should always plan it first.

If you are confident with Command Sequences, you could draw a quick diagram of your database and identify where different pieces of information should be stored.

If you are new to Command Sequences you might want to outline look at the Command Sequences used in the demo scenes. The first thing to think about is the Input Variables you sequence will have and the result you want to get out of the sequence. Then you can fill in the gaps thinking about variables and the different Commands available to achieve this.

The fewer Commands you use the better. The Get Data and Set Data Commands have the highest demands, especially if you are using them inside a For Command. You can use If Commands to prevent to prevent Get Data and Set Data Commands from running when they don't need to. For example, when logging in a user you could iterate through every row in the database and use two Get Data commands to get the username and password and then check them. This could be improved by checking the username before you use Get Data to get the password, this means if the username is wrong, the second Get Data Command will not be run to retrieve the password.

At a last resort, we might be able to help you writing your Command Sequence. Contact us at [solution\\_studios@outlook.com](mailto:solution_studios@outlook.com) or ask us a question through the Bug Reporter Window.

## Creating a Command Sequence

---

You can add Commands to your Command Sequence by left clicking on the Command Name (in the list of Commands in the left hand panel) to add it to the end of the sequence.

You can also right click on the Command Name (in the list of Commands in the left hand panel), and then drag and drop the Command to the required location in the Sequence.

You can left click on any created Command in the central list to select it. This allows you to fill in the properties of the Command in the panel on the right.

You can right click and drag and drop any created Command in the central list to move it.

If you have a longer sequence and you want to drag and drop a Command from the top of the sequence to the bottom (or the other way around), you can use the up and down arrow keys to scroll up and down the central panel while dragging and

dropping a Command. This gives the same ability as the middle mouse button on a three button mouse to users with a touchpad when dragging and dropping Commands.

You can press the cross next to any Command (apart from the Start Command) in the central list of created Commands to delete it.

You can left click on the Start Command to select it. This will show you the variables information in the properties panel.

The Commands you will find the most useful are probably the Get Data, Set Data, If, For and Return Commands (as well as the Start Command for variables).

## Variables

---

Understanding how Database Control Pro uses variables is essential when creating Command Sequences. You will need to understand how to use them and the different types of variables available in Database Control Pro.

## Using Variables

---

Variables in Database Control Pro Command Sequences have the same purpose as variables in all programming languages - To store data which varies.

Command Sequences are a list of Commands, every Command has different properties providing different processes and results. Many of these Commands are based on variables.

All a variable is, is a name and a value. The name doesn't change, but the value does.

Use 1:

You might need to use a variable to provide a Command with information. For example, if you want to write data to the database, you will need to provide the data which needs to be written to the SetData Command. Sometimes this might be the same data every time such as 'Hello World', but in other cases it might change such as when writing a username to the database as it could be 'John' or 'Mark' or some other name. If the data changes you will want to use a variable.

Use 2:

You might need to use a variable to store information created by a Command. For example, if you want to read data from the database, you will use the GetData Command which will return the data. You need a way to store the data which is by using a variable.

When using variables in Command Sequences, you will have to type the variable's name into the correct field of the property of the Command. The value of every variable is in string format. If you are using a variable as a float or int, (unlike normal programming) you can ignore the types as all strings will be parsed into the type required.

## Variable Types

---

Variables can be created in a sequence by some Commands such as the CreateVar Command.

Variables can also be created in the Properties of the Start Command. These variables are divided into three types:

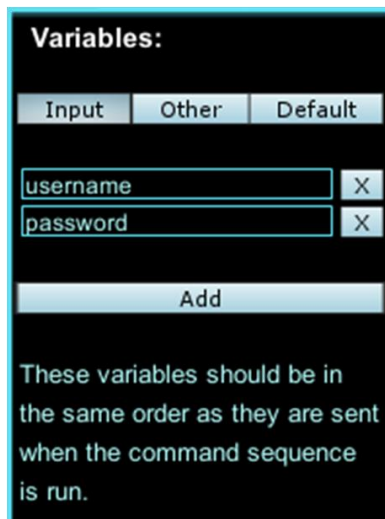
Input Variables

Other Variables

Built-in (or default) Variables

## Input Variables

---



**Variables:**

| Input    | Other | Default |
|----------|-------|---------|
| username |       |         |
| password |       |         |

Add

These variables should be in the same order as they are sent when the command sequence is run.

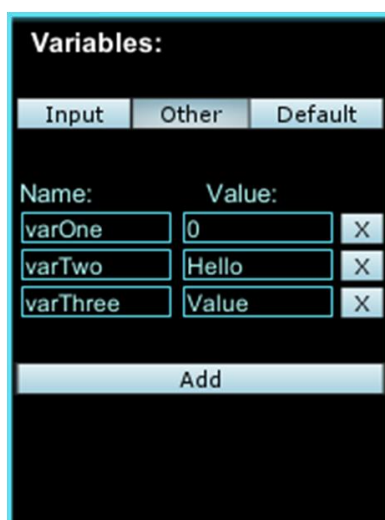
Input Variables are submitted by the game and can be used by the Command Sequence. For example, when logging in a user, the game will send the username and password entered into the UI InputFields. These username and password are input variables.

Input Variables must be sent when a Command Sequence is told to run through code.

Input Variables can be added by selecting the start node in a Command Sequence, going to the 'Input' tab in the properties panel and adding the input variable name to the list. Remember the order of the input variables as they must be added to the run Command Sequence code in the correct order.

## Other Variables

---



**Variables:**

| Input | Other    | Default |
|-------|----------|---------|
|       | varOne   | 0       |
|       | varTwo   | Hello   |
|       | varThree | Value   |

Add

Other Variables are just normal variables which you might create in a Command Sequence using a Command like CreateVar, except they can be created from within the Start node to save unnecessary Commands at the start of a Command Sequence. They must have a value (as well as a name) assigned to them when they are created.

Other Variables can be added by selecting the start node in a Command Sequence, going to the 'Other' tab in the properties panel and adding the other variable name and a value to the list.

## Built-In Variables

---



Built-In variables (also known as default variables) cannot be created; they are useful variables which are created at the start of a Command Sequence whether you use them or not. They include `ipAddress` which stores the player's IP Address and other variables for getting the current server time.

You can see the full list of Built-In variable names by selecting the Start node on any Command Sequence and going to the 'Default' tab in the properties panel.

## Commands

---

A Command Sequence is a list of Commands. A Command can be a process, statement, operation or condition (think of them like methods on our server). Commands in the Command Sequence are run from first to last which looks like top to bottom when opened in the Sequencer Window. Every Command has properties (think of them like parameters for the method) which change the result or functionality of the Command.

The If Command and the For Command can contain lists of Commands. When an If Command's condition is satisfied, the contained Commands are run. When a For Command is iterating, the contained Commands are run (multiple times, once for each iteration) as you would expect with a normal For loop.

The properties of a Command can be set by filling in the fields of the Property panel of the Sequencer Window with a Command selected from a Command Sequence.

Commands can be added to the end of a sequence by left clicking on the Command in the list of Commands on the left of the Sequencer window, or they can be dragged and dropped into the correct position by using right click. Commands in a sequence can also be moved by right click drag and drop. They can be selected by a left click.

These are the Commands Currently Available in Database Control Pro:

- Get Data
- Set Data
- Return
- If
- For
- Create Var
- Set Var Value
- Set Var Value Extra
- Clear Cell



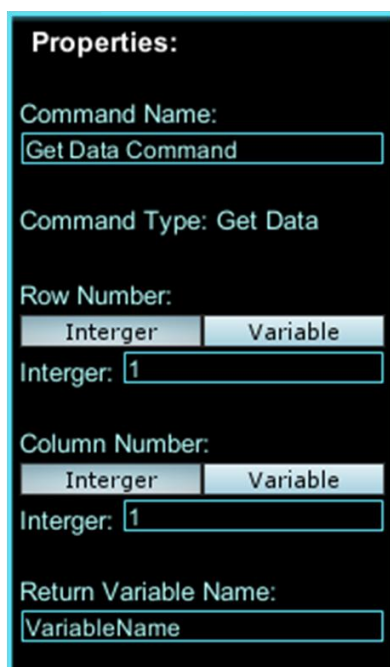
Clear Row  
 Clear Col  
 Del Row  
 Del Col  
 Ins Row  
 Ins Col  
 Float Ops  
 Int Ops  
 String Ops  
 Comment  
 Time Diff

## Get Data

---

### Description:

The GetData Command retrieves the value of a cell in the database. The cell is located by a row number and column number.



The screenshot shows a 'Properties' dialog box for the 'Get Data Command'. It has a dark background with light blue text and input fields. The 'Command Name' is 'Get Data Command'. The 'Command Type' is 'Get Data'. For 'Row Number', there are radio buttons for 'Integer' (selected) and 'Variable', with an input field containing '1'. Similarly, for 'Column Number', the 'Integer' radio button is selected and the input field contains '1'. The 'Return Variable Name' is 'VariableName'.

The Command name can be anything you like, it makes no difference when the sequence is run and is purely for organisation purposes.

The 'Command Type:' line reminds you of the Command Type as you might choose not to put it in the Command's name.

The remaining properties are specific to this Command Type.

### Properties:

Row Number - The Row number of the cell to retrieve the data from.

Column Number - The Column number of the cell to retrieve the data from.

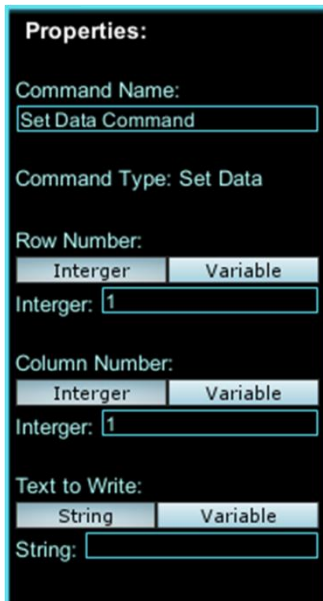
Return Variable Name - The name of the variable to save the retrieved data to. It can overwrite an existing variable or create a new one.

# Set Data

---

**Description:**

The SetData Command writes the value of a cell in the database. It can write a string or variable value. The cell is located using a row number and column number.



The screenshot shows a 'Properties:' window for the 'Set Data Command'. It contains several fields and buttons:

- Command Name:** A text box containing 'Set Data Command'.
- Command Type:** A label 'Set Data'.
- Row Number:** A section with two buttons, 'Integer' and 'Variable', and a text box containing '1'.
- Column Number:** A section with two buttons, 'Integer' and 'Variable', and a text box containing '1'.
- Text to Write:** A section with two buttons, 'String' and 'Variable', and a text box.

The Command name can be anything you like, it makes no difference when the sequence is run and is purely for organisation purposes.

The 'Command Type:' line reminds you of the Command Type as you might choose not to put it in the Command's name.

The remaining properties are specific to this Command Type.

**Properties:**

Row Number - The row number of the cell to write the value of.

Column Number - The column number of the cell to write the value of.

Text to Write - The value to write to the database. It can be a string or variable value (by entering the variable's name).

# Return

---

## Description:

The Return Command controls what text should be sent back to the game when the Command Sequence has finished running. It can force the Command Sequence to stop, or set the text to be returned which can be overwritten by another Return Command later in the sequence.

**Properties:**

Command Name:  
Return Command

Command Type: Return

Return Text:  
String Variable

String:

☒ Is End of Sequence?

The Command name can be anything you like, it makes no difference when the sequence is run and is purely for organisation purposes.

The 'Command Type:' line reminds you of the Command Type as you might choose not to put it in the Command's name.

The remaining properties are specific to this Command Type.

## Properties:

**Return Text** - The text to be sent back to the game. It can be a string or variable value (by entering the variable's name).

**Is End Of Sequence?** - If checked it will force the sequence to end without running any more Commands. If not checked the sequence will continue to run. If there is another Return Command after this Command, the text to be returned from this Command will be overwritten with the new text to return.

# If

---

## Description:

The If Command acts as a simple If statement. If the condition is satisfied the contained Commands will be run. The condition can compare two strings or two numbers.

**Properties:**

Command Name:

Command Type: If

☐ Is Number Comparison?

Operator Type:

String One:   String:

String Two:   String:

**Properties:**

Command Name:

Command Type: If

☒ Is Number Comparison?

Operator Type:

Float One:   Float:

Float Two:   Float:

The Command name can be anything you like, it makes no difference when the sequence is run and is purely for organisation purposes.

The 'Command Type:' line reminds you of the Command Type as you might choose not to put it in the Command's name.

The remaining properties are specific to this Command Type.

### Properties:

Is Number Comparison? - If checked the two strings provided will be parsed into numbers and compared. If not, the two strings will be compared.

Operator Type - The type of comparison which is taking place. '=' checks the values are equal and '!=' checks the values are not equal. If 'Is Number Comparison?' is checked then more operators are available; '<' less than, '>' greater than, '<=' less than or equal to, '>=' greater than or equal to.

If 'Is Number Comparison?' is not checked:

String One - The first string to be compared. It can be a string or a variable value.

String Two - The second string to be compared. It can be a string or a variable value.

If 'Is Number Comparison?' is checked:

Float One - The first number to be compared. It can be a float or a variable value. If it is a variable value it will be parsed into a float.

Float Two - The second number to be compared. It can be a float or a variable value. If it is a variable value it will be parsed into a float.

E.g. If the operator type is '>', Float One is 3 and Float Two is 2, then the condition is '3 > 2' which is true to the containing Commands would be run.

# For

---

## Description:

The For Command acts as a simple for loop.

The screenshot shows a 'Properties' dialog box for the 'For Command'. It contains the following fields and options:

- Command Name:** A text box containing 'For Command'.
- Command Type:** A dropdown menu set to 'For'.
- Comparison Operators:** Two buttons, '<' and '<='.
- Start Number:** A section with two tabs, 'Integer' and 'Variable'. The 'Integer' tab is selected, and the text box next to it contains the value '1'.
- End Number:** A section with two tabs, 'Integer' and 'Variable'. The 'Integer' tab is selected, and the text box next to it contains the value '1'.
- Iterating Var Name:** A text box containing 'Variable Name'.

The Command name can be anything you like, it makes no difference when the sequence is run and is purely for organisation purposes.

The 'Command Type:' line reminds you of the Command Type as you might choose not to put it in the Command's name.

The remaining properties are specific to this Command Type.

## Properties:

'<' or '<=' - This determines the condition which all iterations of the loop must satisfy. If the condition is not met the loop stops. If '<' is selected then the loop will stop when the iterating number is not less than the End Number. If '<=' is selected then the loop will stop when the iterating number is not less than or equal to the End Number.

Start Number - This is the value of the Iterating Variable for the first iteration of the loop.

End Number - This is the value of last or one after the last iteration of the loop based on whether '<' or '<=' is selected.

Iterating Var Name - This is the name of the variable which has its value changed with each iteration of the loop. It can overwrite an existing variable or create a new one.

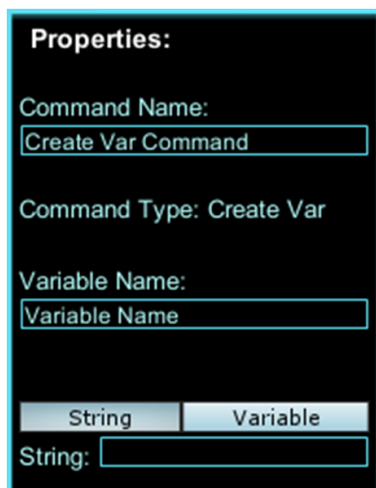
E.g. If '<=' is selected, the Start Number is 2 and the End Number is 5, the loop will be equivalent to 'for (int i=2; i <= 5; i++) {' in C# (where i is the Iterating Variable). This will cause iterations where i = 2, 3, 4 and 5.

## Create Var

---

### Description:

The CreateVar Command creates a new variable with a value. The value can be a set string or copy the value of another variable given its name.



**Properties:**

Command Name:

Command Type: Create Var

Variable Name:

String ☐ Variable ☐

String:

The Command name can be anything you like, it makes no difference when the sequence is run and is purely for organisation purposes.

The 'Command Type:' line reminds you of the Command Type as you might choose not to put it in the Command's name.

The remaining properties are specific to this Command Type.

### Properties:

Variable Name - The name of the new variable to be created.

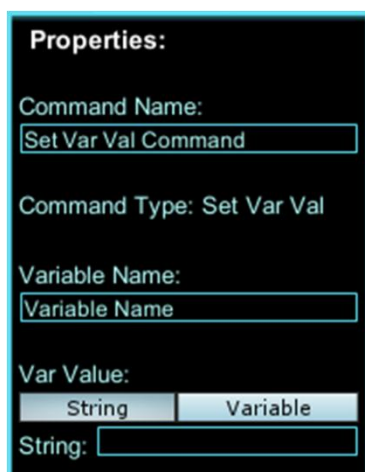
Variable Value - The initial value of the new variable. It can be a set string or copy the value of another variable. (Remember all variable values are strings. If you are using a number it will be automatically parsed.)

## Set Var Value

---

### Description:

The SetVarValue Command sets the value of a variable to a string or to another variable's value.



**Properties:**

Command Name:

Command Type: Set Var Val

Variable Name:

Var Value:  
☐ String ☐ Variable

String:

The Command name can be anything you like, it makes no difference when the sequence is run and is purely for organisation purposes.

The 'Command Type:' line reminds you of the Command Type as you might choose not to put it in the Command's name.

The remaining properties are specific to this Command Type.

#### Properties:

Variable Name - The name of the variable whose value needs to set.

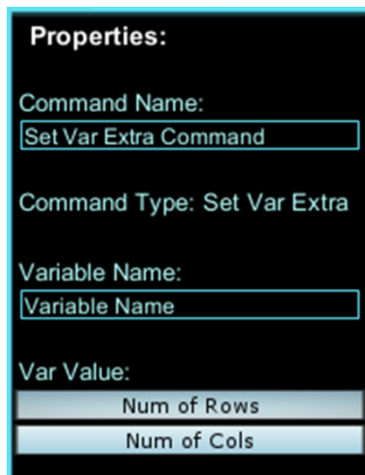
Var Value - The new value of the variable. It can be a string or copy the value from another variable.

## Set Var Value Extra

---

#### Description:

The SetVarValueExtra Command sets the value of a variable to one of the options available. There are currently only two options available in Database Control Pro. The extra options require some server 'effort' to be worked out, this is why they are not default variables.



The screenshot shows a 'Properties:' window for the 'Set Var Extra Command'. It contains the following fields and options:

- Command Name:** A text box containing 'Set Var Extra Command'.
- Command Type:** A dropdown menu set to 'Set Var Extra'.
- Variable Name:** A text box containing 'Variable Name'.
- Var Value:** A dropdown menu with two options: 'Num of Rows' and 'Num of Cols'.

The Command name can be anything you like, it makes no difference when the sequence is run and is purely for organisation purposes.

The 'Command Type:' line reminds you of the Command Type as you might choose not to put it in the Command's name.

The remaining properties are specific to this Command Type.

#### Properties:

Variable Name - The name of the variable to set the value of.

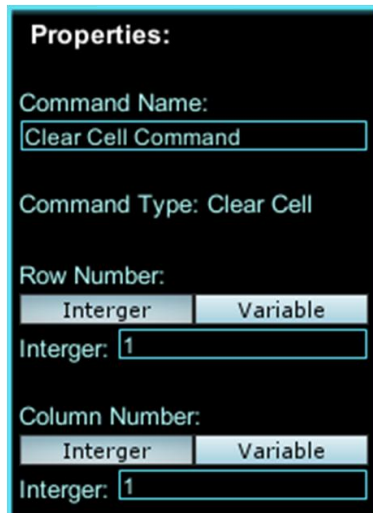
Variable Value - The new value of the variable. It has to be one of the options available, currently 'Num of Rows' and 'Num of Cols' are the only options where the number of rows or columns in the database are retrieved (by counting the rows from cell 0,0 to the cell with the highest row/column number).

## Clear Cell

---

### Description:

The ClearCell Command removes the value from the specified cell. The cell is identified by its row number and column number.



**Properties:**

Command Name:

Command Type: Clear Cell

Row Number:  
   
 Integer:

Column Number:  
   
 Integer:

The Command name can be anything you like, it makes no difference when the sequence is run and is purely for organisation purposes.

The 'Command Type:' line reminds you of the Command Type as you might choose not to put it in the Command's name.

The remaining properties are specific to this Command Type.

### Properties:

Row Number - The row number of the cell to clear the value of. It can be a set integer or be parsed from a variable value.

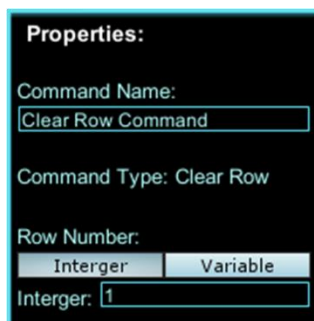
Column Number - The column number of the cell to clear the value of. It can be a set integer or be parsed from a variable value.

## Clear Row

---

### Description:

The ClearRow Command removes all values from all of the cells in the specified row. The row is identified by a row number.



**Properties:**

Command Name:

Command Type: Clear Row

Row Number:  
   
 Integer:



The Command name can be anything you like, it makes no difference when the sequence is run and is purely for organisation purposes.

The 'Command Type:' line reminds you of the Command Type as you might choose not to put it in the Command's name.

The remaining properties are specific to this Command Type.

#### Properties:

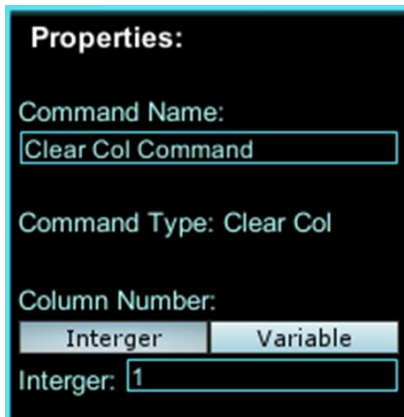
Row Number - The row number of the row to clear the values of. It can be an interger or parsed from a variable value.

## Clear Col

---

#### Description:

The ClearCol Command removes all values from all of the cells in the specified column. The column is identified by a column number.



**Properties:**

Command Name:  
Clear Col Command

Command Type: Clear Col

Column Number:  
☒ Integer
 ☐ Variable

Integer: 1

The Command name can be anything you like, it makes no difference when the sequence is run and is purely for organisation purposes.

The 'Command Type:' line reminds you of the Command Type as you might choose not to put it in the Command's name.

The remaining properties are specific to this Command Type.

#### Properties:

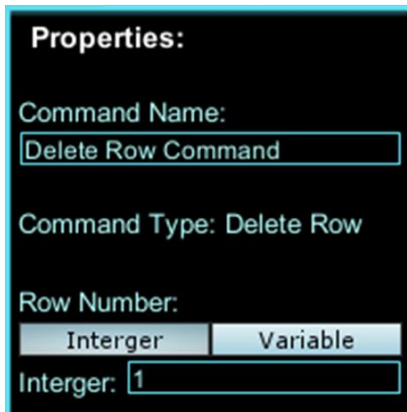
Column Number - The column number of the column to clear the values of. It can be an integer or parsed from a variable value.

## Del Row

---

### Description:

The DelRow Command completely removes the specified row from the database. All rows bellow the specified row will have their row number decrease by 1. The row is located by its row number.



**Properties:**

Command Name:  
Delete Row Command

Command Type: Delete Row

Row Number:  
☐ Integer ☐ Variable  
 Integer: 1

The Command name can be anything you like, it makes no difference when the sequence is run and is purely for organisation purposes.

The 'Command Type:' line reminds you of the Command Type as you might choose not to put it in the Command's name.

The remaining properties are specific to this Command Type.

### Properties:

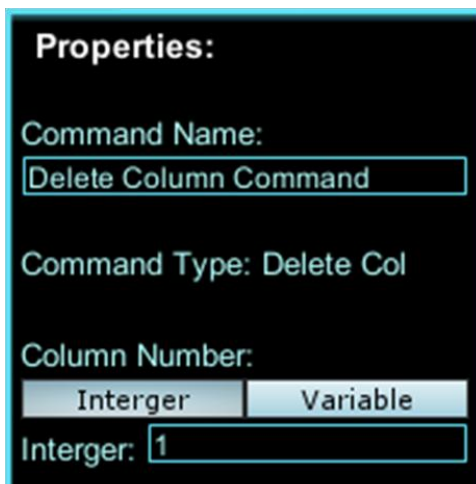
Row Number - The row number of the row to delete. It can be an integer or parsed from a variable value.

## Del Col

---

### Description:

The DelCol Command completely removes the specified column from the database. All columns to the right of the specified column will have their column number decrease by 1. The column is located by its column number.



**Properties:**

Command Name:  
Delete Column Command

Command Type: Delete Col

Column Number:  
☐ Integer ☐ Variable  
 Integer: 1

The Command name can be anything you like, it makes no difference when the sequence is run and is purely for organisation purposes.

The 'Command Type:' line reminds you of the Command Type as you might choose not to put it in the Command's name.

The remaining properties are specific to this Command Type.

#### Properties:

Column Number - The column number of the column to delete. It can be an integer or parsed from a variable value.

## Ins Row

---

#### Description:

The InsRow Command inserts a new row into the database. All rows bellow the inserted row will have their row number increase by 1. The row is inserted at the position specified by its row number.

**Properties:**

Command Name:  
Insert Row Command

Command Type: Insert Row

Row Number:  
☒ Integer ☐ Variable  
 Integer: 1

The Command name can be anything you like, it makes no difference when the sequence is run and is purely for organisation purposes.

The 'Command Type:' line reminds you of the Command Type as you might choose not to put it in the Command's name.

The remaining properties are specific to this Command Type.

#### Properties:

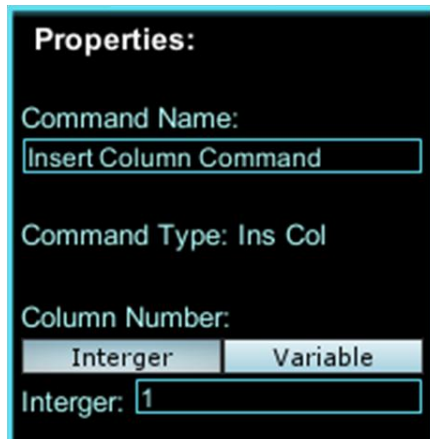
Row Number - The row number to insert the new row at. It can be an integer or parsed from a variable value.

## Ins Col

---

**Description:**

The InsCol Command inserts a new column into the database. All columns to the right of the inserted column will have their column number increase by 1. The column is inserted at the position specified by its column number.



The screenshot shows a dialog box titled 'Properties:'. It contains the following fields and options:

- Command Name:** A text box containing 'Insert Column Command'.
- Command Type:** A label 'Ins Col'.
- Column Number:** A section with two radio buttons: 'Integer' (selected) and 'Variable'.
- Integer:** A text box containing the value '1'.

The Command name can be anything you like, it makes no difference when the sequence is run and is purely for organisation purposes.

The 'Command Type:' line reminds you of the Command Type as you might choose not to put it in the Command's name.

The remaining properties are specific to this Command Type.

**Properties:**

Column Number - The column number to insert the new column at. It can be an integer or parsed from a variable value.

## Float Ops

---

**Description:**

The FloatOps Command provides the ability to do simple math operations of floats and ints. All variable values used will be parsed as floats, then the operation will be performed and they will be stored as a string variable value again.

**Properties:**

Command Name:

Command Type: Float Ops

Result Variable Name:

Float One:  
   
 Float:

Float Two:  
   
 Float:

The Command name can be anything you like, it makes no difference when the sequence is run and is purely for organisation purposes.

The 'Command Type:' line reminds you of the Command Type as you might choose not to put it in the Command's name.

The remaining properties are specific to this Command Type.

### Properties:

**Operators** - The main basic math operators are provided. '+' adds the two numbers, '-' subtracts the second number from the first number, '\*' multiplies the two numbers, '/' divides the first number by the second number and '^' puts the first number to the power of the second number.

**Result Variable Name** - The variable name of the variable which the calculated value will be stored as.

**Float One** - The first number in the calculation. It can be a set int, float or it can be parsed from a variable value.

**Float Two** - The second number in the calculation. It can be a set int, float or it can be parsed from a variable value.

# Int Ops

---

## Description:

The IntOps Command provides some very simple operations to use to get intergers. If you want to perform simple interger calculations use the FloatOps Command.

| Properties:   |  |
|---|--|
| Command Name:   | <input type="text" value="Int Ops Command"/>                                 |
| Command Type: Int Ops   |  |
| <input type="button" value="Round"/> <input type="button" value="Floor"/> <input type="button" value="Length Of String"/> |  |
| Result Variable Name:   | <input type="text" value="Variable Name"/>                                   |
| Number to Round:  | <input type="button" value="Float"/> <input type="button" value="Variable"/> |
| Float: <input type="text" value="0"/>   |  |

| Properties:   |   |
|---|---|
| Command Name:   | <input type="text" value="Int Ops Command"/>                                  |
| Command Type: Int Ops   |   |
| <input type="button" value="Round"/> <input type="button" value="Floor"/> <input type="button" value="Length Of String"/> |   |
| Result Variable Name:   | <input type="text" value="Variable Name"/>                                    |
| String to get length of:  | <input type="button" value="String"/> <input type="button" value="Variable"/> |
| String: <input type="text"/>  |   |

The Command name can be anything you like, it makes no difference when the sequence is run and is purely for organisation purposes.

The 'Command Type:' line reminds you of the Command Type as you might choose not to put it in the Command's name.

The remaining properties are specific to this Command Type.

## Properties:

Operations - There are currently three different operations available. 'Round' rounds the provided float to an integer, 'Floor' rounds the provided float down to the highest integer less than or equal to it and 'Length Of String' gets an integer from finding the length of the provided string.

If 'Round' or 'Floor' selected:

Number to 'Round'/'Floor' - The float to be rounded or floored to give an integer. It can be a set float or a variable value parsed to a float.

If 'Length Of String' selected:

String to get length of: - The string to get the integer length of. It can be a set string of a variable's value.

# String Ops

---

## Description:

The StringOps Command provides three simple string operations to perform on your strings. You can use set strings or variable values.

The image displays two side-by-side screenshots of a configuration window for a command named 'String Ops Command'. Both windows show the 'Properties' section with the following fields and options:

- Command Name:** String Ops Command
- Command Type:** String Ops
- Operation Tabs:** Concat, ToUpper, ToLower (In the left window, 'Concat' is selected. In the right window, 'ToUpper' is selected).
- Result Variable Name:** Variable Name
- String One:** String (selected) / Variable
- String:** [Empty text box]
- String Two:** String (selected) / Variable
- String:** [Empty text box]

The Command name can be anything you like, it makes no difference when the sequence is run and is purely for organisation purposes.

The 'Command Type:' line reminds you of the Command Type as you might choose not to put it in the Command's name.

The remaining properties are specific to this Command Type.

## Properties:

**Operations** - The three simple string operations. 'Concat' joins the two strings together (the first string followed by the second string) to form a larger string, 'ToUpper' makes all of a string's characters uppercase and 'ToLower' makes all of a string's characters lowercase.

**Result Variable Name** - The variable name of the variable to store the string result as.

**String One** - The first/only string to perform the operation on.

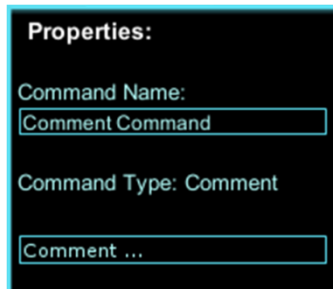
**String Two** (if 'Concat' selected) - The second string to perform the operation on.

## Comment

---

### Description:

The Comment command has no role in the running of a Command Sequence. It is for organization use only and allows you to add a comment explaining something which someone might read at a later date. It is used as the first Command in all demo Command Sequences to explain how the sequence works.

A screenshot of a software interface showing the 'Properties' for a 'Comment Command'. The dialog box has a dark background with light text. It contains three input fields: 'Command Name' with the value 'Comment Command', 'Command Type' with the value 'Comment', and a text area labeled 'Comment ...'.

The Command name can be anything you like, it makes no difference when the sequence is run and is purely for organisation purposes.

The 'Command Type:' line reminds you of the Command Type as you might choose not to put it in the Command's name.

The remaining properties are specific to this Command Type.

### Properties:

Comment Text - The text associated with the comment. It has no effect on the running of the Command Sequence.

## Time Diff

---

### Description:

The TimeDiff Command calculates the difference between two times which can be returned in a variety of different lengths of time. Each time consists of 7 floats: Milliseconds, Seconds, Minutes, Hours, Days, Months and Years. Each float can be a set number or can be parsed from a variable value. The Default Variables can be used to get the current time or you can use the toggle for Time One.

Note: Month is zero-based, this means January is represented by 0 and December by 11 (not 12).

(Images on next two pages)



**Properties:**

Command Name:

Time Diff Command

Command Type: Time Diff

Return Time in ...

Milliseconds

Seconds

Minutes

Hours

Days

Months

Years

☐ Round Result?

Return Variable Name:

Variable Name

☐ Allow Negative Diffs?

Time One:

☐ Use Current?

Milliseconds:

Interger

Variable

Interger:

1

Seconds:

Interger

Variable

Interger:

1

Minutes:

Interger

Variable

Interger:

1

Hours:

Interger

Variable

Interger:

1

Days:

Interger

Variable

Interger:

1

Months: (0 is Jan)

Interger

Variable

Interger:

1

Years:

Interger

Variable

Interger:

1

Time Two:

Milliseconds:

|          |          |
|----------|----------|
| Interger | Variable |
|----------|----------|

Interger: 1

Seconds:

|          |          |
|----------|----------|
| Interger | Variable |
|----------|----------|

Interger: 1

Minutes:

|          |          |
|----------|----------|
| Interger | Variable |
|----------|----------|

Interger: 1

Hours:

|          |          |
|----------|----------|
| Interger | Variable |
|----------|----------|

Interger: 1

Days:

|          |          |
|----------|----------|
| Interger | Variable |
|----------|----------|

Interger: 1

Months: (0 is Jan)

|          |          |
|----------|----------|
| Interger | Variable |
|----------|----------|

Interger: 1

Years:

|          |          |
|----------|----------|
| Interger | Variable |
|----------|----------|

Interger: 1

The Command name can be anything you like, it makes no difference when the sequence is run and is purely for organisation purposes.

The 'Command Type:' line reminds you of the Command Type as you might choose not to put it in the Command's name.

The remaining properties are specific to this Command Type.

### Properties:

Return Time In - The length of time to return the time in. For example if 'Minutes' is selected an interger for the number of minutes will be returned.

Round Result - If checked, the result will be rounded to the nearest interger.

Return Variable Name - The name of the variable which the calculated value should be stored with.

Allow Negative Diffs? - The calculation does 'Time One - Time Two'. If checked and time two is after time one the result will be negative.

Time One Use Current? - If checked time one will be set to the current time, if not you can set all of the floats yourself. Time One/Two lengths:

Every length of time is a float which can be set or parsed from a variable value. If you don't want to include a certain length of time (e.g. don't need milliseconds) then just set the float for that length of time to 0 in both times.

## Running a Command Sequence at Runtime

---

To run a Command Sequence at runtime you will need to write your own code or use the code from the demos. A Command Sequence can be run using C# or UnityScript.

Database Control Pro includes a very basic example scene to get you started. There is a C# version and a UnityScript version of the scene.

The C# version can be found at Assets>Database Control Pro>Demos>Extras>Run CS On Space Bar>C#>RunCSOnSpaceBar

The UnityScript version can be found at Assets>Database Control Pro>Demos>Extras>Run CS On Space Bar>Js>RunCSOnSpaceBar

Before you hit play make sure you have setup a Command Sequence and fill in the fields on the script attached to the Manager gameObject.

When you run the scene, hit the space bar and the Command Sequence will be run. The result will be displayed in the console.

To see how the Command Sequence is run, take a look at the script attached to the Manager gameObject or look through the code on the next two pages.

## C#

---

In order to run a Command Sequence in C# you will need to reference the Database Control Pro namespace. To do this add the following to the top of your script:

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

Now to run the sequence you will need to create an IEnumerator and use StartCoroutine to run it. A IEnumerator is needed as it will contain a delay to wait for the result when the Command Sequence finishes.

To Run the IEnumerator, use the following code wherever you need it:

```
StartCoroutine(RunSequence());
```

Finally create your IEnumerator anywhere in the class just like a normal method and make sure it includes the following code:

```
IEnumerator RunSequence()
{
    IEnumerator e = DCP.RunCS("Database 1",
        "Login", new string[2] {"johnSmith", "password123"});

    while(e.MoveNext()) {
        yield return e.Current;
    }
    string returnText = e.Current as string;

    Debug.Log("Result: " + returnText);
}
```

The Debug.Log line just writes the Command Sequence response to the Console. This can be replaced with any code you like using the returnText variable (string) as the response.

In this example the Command Sequence called "Login" is being run on the "Database 1" database. The first input variable (username) will have its value set to "johnSmith" and the second (password) to "password123". The string[] for input values needs to match the number of input variables in your Command Sequence and they need to be in the correct order. If your sequence doesn't have any input variables this third parameter can be completely left out.

Look at the code used in the Demo Scenes for more examples.

## UnityScript

---

In order to run a Command Sequence in UnityScript you will need to reference the Database Control Pro namespace. To do this add the following to the top of your script:

```
import DCPTestCompiled;
```

We would advise running the sequence in a separate function as it will involve a delay. To do this we need to create the function and run it.

To Run the function, use the following code wherever you need it:

```
RunSequence();
```

Finally create your function anywhere in the script and make sure it includes the following code:

```
function RunSequence () {
    var e = DCP.RunCS(("Database 1",
        "Login", ["johnSmith", "password123"]));

    while(e.MoveNext()) {
        yield e.Current;
    }
    var returnText = e.Current;

    Debug.Log("Result: " + returnText);
}
```

The Debug.Log line just writes the Command Sequence response to the Console. This can be replaced with any code you like using the returnText variable (string) as the response.

In this example the Command Sequence called "Login" is being run on the "Database 1" database. The first input variable (username) will have its value set to "johnSmith" and the second (password) to "password123". The string[] for input values needs to match the number of input variables in your Command Sequence and they need to be in the correct order. If your sequence doesn't have any input variables this third parameter can be completely left out.

Look at the code used in the Demo Scenes for more examples.

## The End

---

Thanks again for purchasing Database Control Pro.

If you find any bugs please let us know using the Bug Reporting Window at:

Top Menu Bar > Window > Database Control Pro > Other > Bug Reporter

We will try to update Database Control Pro regularly, to see the latest news take a look at the Database Control Pro Unity Forum or our website.

If you have any questions or problems, you can contact use through the Bug Reporter Window or you can email us at: [solution\\_studios@outlook.com](mailto:solution_studios@outlook.com)

or you can post you issue on the Database Control Pro Unity Forum or fill in a contact form on our website. (links on the first page)

Thank you!