Cosmos Cove Control Panel

for SpacetimeDB Version 0.4.2 - Online Documetation

Thank you for your interest in using Cosmos Cove Control Panel for SpacetimeDB!

My wish is for this addon is to simplify anyones journey into creating their own MMO using the new database system developed by Clockwork Labs. Gone is the necessity of having to setup individual SQL services or to construct complicated server logic relationships across a dozen different systems to be able to run your game or app. In SpacetimeDB you write your multiuser logic locally with both database and server logic in the same module, and then publish it to your server.

Initially this asset focuses on all developers using Windows 10/11 for Unity client development and Windows Subsystem for Linux (WSL) with Debian as a local SpacetimeDB server and CLI. This then allows you also to publish to a custom remote Debian server via the Custom Remote mode, or to use Maincloud which is the official paid hosting solution by Clockworklabs (it's free to use for the time being). This asset allows you to install everything needed to get your server running in an easy and intuitive installation process - whether locally or remotely.

Pre-requisites

On your Windows 10 or 11	**On your Debian WSL VM** *(auto installer)*
Unity 6.2.0f (any render pipeline) or Unity 2021.2+ (any render pipeline) *WSL1 or WSL2 with Debian (auto installer)*	*Debian 13 Trixie* *Curl* *SpacetimeDB* *SpacetimeDB service (standard systemd)* *Rust (optional, recommended)* *Wasm-Opt (optional, recommended)*

Note! This is an early version. I can't guarantee everything will work for your particular configuration. **Use version control** and notify me if you find any bugs. All important actions like modifying any data will display confirmation dialogs before proceeding.

Cosmos Cove Control Panel features an automatic **Installer Window** that should get everything needed installed on your Windows PC. Select it from the Welcome screen or through Unity´s toolbar **SpacetimeDB > Installer Window** to get started.

In case the automatic **Installer Window** won't work for your configuration, you can try the manual steps below to get your SpacetimeDB server running in a fresh Unity project.

Quick Start

1. Add the asset to your project from the Package Manager.

2. A Welcome window will appear. Click on **Main Window**.

3. In **Pre-Requisites** enter your settings for the **WSL Local** server which you need to create first in order to be able to have access to a local SpacetimeDB CLI.
    - **Client Language**, **Server Language** and **URL** are fine with default settings.
    - **Client Path** is where SpacetimeDB will autogenerate local bindings scripts. You can choose or create any directory in your Assets folder. You will never edit these directly, but will refer to the results in your scripts.
    - **Module New Entry** is where you enter the name of your new SpacetimeDB module and press **Add**. A module is a self-contained bundle of server logic that can be the entire server or you can have several, for example one per responsibility. Start with creating just one. If you already have a module since before you can just enter the name and location of it when you press **Add**. **Don't Init New Module** before having the WSL Local server installed which is the next step.

4. Open **Installer Window** (or follow the Manual Install Process below) and start from the top to **Install** each item. This will install WSL and all necessary software on your WSL instance to be able to run SpacetimeDB.
    - Command line windows will appear showing the progress and when finished requires a key input to close.
    - **Install WSL with Debian** asks for a UNIX username and password when it is successfully finished. Trixie is now included by default in the Debian install.
    - **Install SpacetimeDB Service** may ask for the UNIX password to install.
    - Install all available items for WSL Local.

5. Press **Init New Module** (if you created a new module), **Start SpacetimeDB WSL** and **Publish** your default module. After a successful publish you can now **Edit Module** and watch the results in the **Logs**, **Database** and **Reducer** windows. **Best of luck with your MMO development!**

    (If you try to start the server before the pre-requisites are met a dialog will appear that shows what you need to fix.)

Tip! Even if you wish to only run a **Custom Remote** or **Maincloud** server you still need to first install a **WSL Local** server. This because you need a local SpacetimeDB CLI to be able to publish and generate SpacetimeDB scripts and push them to the server.

Manual Install Process

For clarity and in case the automatic installer doesn't work for you, here are the manual steps of what the **Installer Window** accomplishes behind the scenes.

To install Windows Subsystem for Linux (WSL) is the first and most important step, because it enables Virtual Machine Platform hypervisor, which can be incompatible with WMWare and VirtualBox. If you use any other virtualization on your Windows, please be sure to check your compatibility with WSL1 and WSL2 before continuing.

The **Installer Window** of Cosmos Cove Control Panel features a quick compatibility report that checks if your PC has the necessary features to enable WSL2. Even if you are planning to install everything manually, you can still launch only that tool by selecting "Install WSL with Debian" and see the result. You have to confirm before the actual install begins.

> **Note!** It is recommended to install WSL2 for your system for near native Linux performance. Regardless of chosen WSL, remember to always backup anything important on your PC before continuing. While being an official Microsoft feature it changes system settings that may require some time to restore on an incompatible PC. Check the next chapter **WSL** for common fixes.

Enter the step 1 command into your Powershell to install WSL2 with Debian. Then on step 2 and onwards you will enter them one by one in the Debian CMD window which you launch from the Debian shortcut in your Windows start menu. The "Settings and Debian" foldout in Server Management Panel also has a button to launch your Debian CMD window.

1. **Install WSL with Debian Trixie (13) in Powershell**
   wsl --install -d Debian
   (above is for WSL2, check the WSL chapter below for WSL1 instructions)

   > **Info!** Afterwards launch Debian from your start menu to create your Debian credentials if you weren't asked during the WSL install process. Write down your username so you can enter it into CCCP when prompted.

2. **Optional: (in case the new Trixie version wasn't installed directly) Launch Debian and update Debian Bookworm (12) to Trixie (13)**
   sudo apt update && sudo apt upgrade -y
   sudo apt install -y update-manager-core
   sudo sed -i 's/bookworm/trixie/g' /etc/apt/sources.list
   sudo apt update
   sudo apt full-upgrade -y
   wsl.exe --shutdown

   > **Info!** If you attempted to install Trixie with the Installer Window and it didn't finish, you can often finish the install manually with the commands:
   > sudo dpkg --configure -a
   > sudo apt --fix-broken install
   > sudo apt update

```
sudo apt full-upgrade -y
```

3. **Start Debian again and Verify that it´s Trixie**
   cat /etc/os-release

4. **Install cURL**
   sudo apt update
   sudo apt install curl

5. **Install SpacetimeDB with default settings**
   curl -sSf https://install.spacetimedb.com | sh

6. **Add SpacetimeDB to Debian PATH**
   nano ~/.bashrc
   Add this line at the bottom
   export PATH="/home/m/.local/bin:$PATH" (check what it tells you to add at install)
   Restart to use spacetime command

7. **Create SpacetimeDB Services**
   nano /etc/systemd/system/spacetimedb.service
   Add this to the newly created spacetimedb.service file (replace yourusername)
   [Unit]
   Description=SpacetimeDB Server
   After=network.target
   [Service]
   User=yourusername
   Environment=HOME=/home/yourusername
   ExecStart=/home/yourusername/.local/bin/spacetime
   --root-dir=/home/yourusername/.local/share/spacetime start
   --listen-addr=0.0.0.0:3000
   Restart=always
   WorkingDirectory=/home/yourusername
   [Install]
   WantedBy=multi-user.target
   Ctrl+O and Enter to save. Then Ctrl+X to exit.
   nano /etc/systemd/system/spacetimedb-logs.service
   [Unit]
   Description=SpacetimeDB Database Logs
   After=spacetimedb.service
   Requires=spacetimedb.service
   [Service]
   User=yourusername
   Environment=HOME=/home/yourusername
   Type=simple
   Restart=always
   RestartSec=5
   ExecStart=/home/yourusername/.local/bin/spacetime logs magical -f

WorkingDirectory=/home/yourusername
[Install]
WantedBy=multi-user.target
Ctrl+O and Enter to save. Then Ctrl+X to exit.

sudo systemctl daemon-reload
sudo systemctl enable spacetimedb.service
sudo systemctl start spacetimedb.service
sudo systemctl enable spacetimedb-logs.service
sudo systemctl start spacetimedb-logs.service

8. **Install Rust (Recommended, if using Rust for server development)**
   sudo apt install build-essential
   curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
   . "$HOME/.cargo/env"

9. **Install wasm-opt (binaryen) (Recommended)**
   wget
   https://github.com/WebAssembly/binaryen/releases/download/version_123/binaryen-version_123-x86_64-linux.tar.gz
   tar -xzf binaryen-version_123-x86_64-linux.tar.gz
   sudo cp binaryen-version_123/bin/* /usr/local/bin/
   Then delete the temporary install files
   ls -la (to list files and) rm -r <file or folder name> (to delete the files)

10. **Install Git (Needed by certain spacetime commands)**
    sudo apt update
    sudo apt install -y git

11. **Initialize a new module from Main Window**

12. **Start server and publish new module from Server Manager Window**

13. **You can now start developing your SpacetimeDB server! Change code in your lib.rs located at <Server Directory>/src/lib.rs and publish to see changes, or enable Auto Publish Mode that will publish, generate and compile every time you return to Unity after having changed server code.**

WSL

If your system is incompatible with WSL2, reported by the compatibility tool in Server Installer Window or by your own research - you can install WSL1 for similar functionality by entering these commands in Powershell (Administrator).

1.

```
dism.exe /online /enable-feature
/featurename:Microsoft-Windows-Subsystem-Linux /all /norestart;
```

2.

```
wsl --set-default-version 1 && wsl --install -d Debian
```

If WSL1 reports as incompatible with messages like:
“WSL1 is not supported with your current machine configuration.”
Restart your PC and try again. It should now work.

Note! If WSL was installed previously and Windows Subsystem for Linux is still active, a Windows popup may appear when opening the Server Installer Window prompting to install WSL. Disregard it and use the install button to install it.

About WSL2 vs WSL1

WSL2 is the most performant and accurate way to run Debian Linux on your new Windows PC. This alternative is better for new PCs. Memory usage: 1-2 GB	WSL1 is more compatible with different Windows PCs since it doesn’t use virtualization. This alternative is better for older PCs. Memory usage: A few mb

Note! In case you tried to install WSL2 on an incompatible PC in rare cases it can’t boot normally. A common fix for this is to force-power-off your PC (hold the power button) three times in a row during the logo splash and on the third boot Windows will enter the blue Windows recovery tool called WinRE. Then Troubleshoot → Enable advanced startup. Then on the next boot you select Safe Mode (4). Windows will in most cases fix itself and you can reboot normally.
You can also enter command prompt in WinRE by going to Troubleshoot → Advanced options → Command Prompt and entering “bcdedit /set hypervisorlaunchtype off”.
Then you can start normally and install WSL1.

Useful Powershell commands for WSL

Enter your Debian
wsl -d Debian
List the current active distributions on your WSL
wsl --list --verbose

Update WSL
wsl --update

To export a hdd image of your entire WSL Debian
wsl --export Debian C:\Debian.vhdx --vhd

To import a hdd image of your entire WSL Debian
wsl --unregister Debian (this deletes your current Debian)
wsl --import Debian "C:\path\to\install\location" "C:\path\to\your\debian.vhdx"

To export a tar of your entire WSL Debian
wsl --export Debian C:\Debian.tar

Remove Debian
wsl --unregister Debian

Uninstall WSL (first unregister any OSes)
wsl --uninstall

SpacetimeDB as a Service

Services in the Linux systemd are automatically configured if you run the automatic **Server Installer** window. Both WSL Local mode and Custom Remote mode works this way by default.

It is possible to just run SpacetimeDB in the foreground with the **spacetime start** command, but Cosmos Cove Control Panel now only supports running SpacetimeDB as a service. You can always manually do the foreground command to quickly test that it runs provided the Spacetime CLI and WSL are installed.

In all other cases it is recommended to run SpacetimeDB as a service. And this is what the asset does by default.

The reasons are: stability, logging and performance.

When we run SpacetimeDB as a service it saves all logs to the journalctl logs of the Debian OS. This generally simplifies things because we can just periodically read the journalctl logs behind the scenes and output them into the Logs window.

Running as a service also allows proper start and stop commands. Otherwise we´d have to kill the process to stop if, if it is running in foreground and not as a service. This improves stability and prevents any data loss.

The location of the service files are here
\\wsl.localhost\Debian\etc\systemd\system

You are supposed to have two SpacetimeDB services
spacetimedb.service (Main service which runs the server and saves module logs)
spacetimedb-logs.service (Database service which only outputs database logs)

How to import CSV or JSON

Note! Make sure you have backed up your current database before import.

Currently Cosmos Cove Control Panel only supports importing CSV or JSON database files in Server Database Window through a custom reducer in lib.rs. The reducer needs to have matching entries for the tables that are in your database.

Here is an example reducer that works for my project:
https://github.com/strosz/com.northernrogue.cccp.spacetimedbserver/blob/main/Samples/import_reducer_example.rs

Integrate a similar server code into your project keeping the reducer name with expected variables since this is what the client code expects.

```
pub fn import_table_data(ctx: &ReducerContext, table_name: String, json_data: String) -> Result<(), String>
```

Ask your LLM of choice to transform the import_reducer_example.rs to support all your project´s tables and features after you have pasted it into your own lib.rs, or rust module.

Any new database changes you make to tables after that will also need an updated reducer for the import to work. It's quick and easy to ask your LLM to update it.

You can then use the "Import from Folder…" and "Import Single File…" in the Database Window.

The Database Window and the import process only supports public tables.

This feature is subject to change and may be improved in the future.

ClockworkLabs has announced that they are releasing their own fully featured database tool, and for this reason the development of the Server Database Window won't continue until that tool is released and we can possibly make Cosmos Cove Control Panel have an option to open that database tool.

SpacetimeDB Tips and Tricks

Tables

SpacetimeDB does not support changing existing tables while preserving the data. When you need to change an existing table you should instead use the following workflow.

1. Create a new table with the new row/rows.
2. Write a migration reducer.
3. Run the reducer with **Run Reducer** window which moves the existing data to the new table - handling the new row/rows.

Example of a migration reducer from message to messages where “id” row was added to the new message table

```
// Migration reducer to move messages from old to new table
#[spacetimedb::reducer]
pub fn migrate_messages(ctx: &ReducerContext) -> Result<(), String> {
    log::info!("Starting message table migration...");
    let mut migrated_count = 0;

    // Iterate through all messages in the old table
    for old_message in ctx.db.message().iter() {
        // Insert into new table with auto-incrementing ID
        ctx.db.messages().insert(Messages {
            id: 0, // Will be auto-assigned
            sender: old_message.sender,
            sent: old_message.sent,
            text: old_message.text.clone(),
            sender_name: old_message.sender_name.clone(),
        });
        migrated_count += 1;
    }

    log::info!("Migration complete. Migrated {} messages.", migrated_count);
    Ok(())
}
```

If you wish to delete a table, you have to first make a backup of all data for all tables. You can do this with the export to JSON or CSV command in the **Database Window**. Then Ctrl+Alt publish your module to clear all data. This will update the tables if you removed any, and then you can re-import all data.

You can always Ctrl+Alt publish your module to reset your database and this will make all your table changes or deletions be accepted by the server. This will remove all data, so always have a backup before!