Python Commands

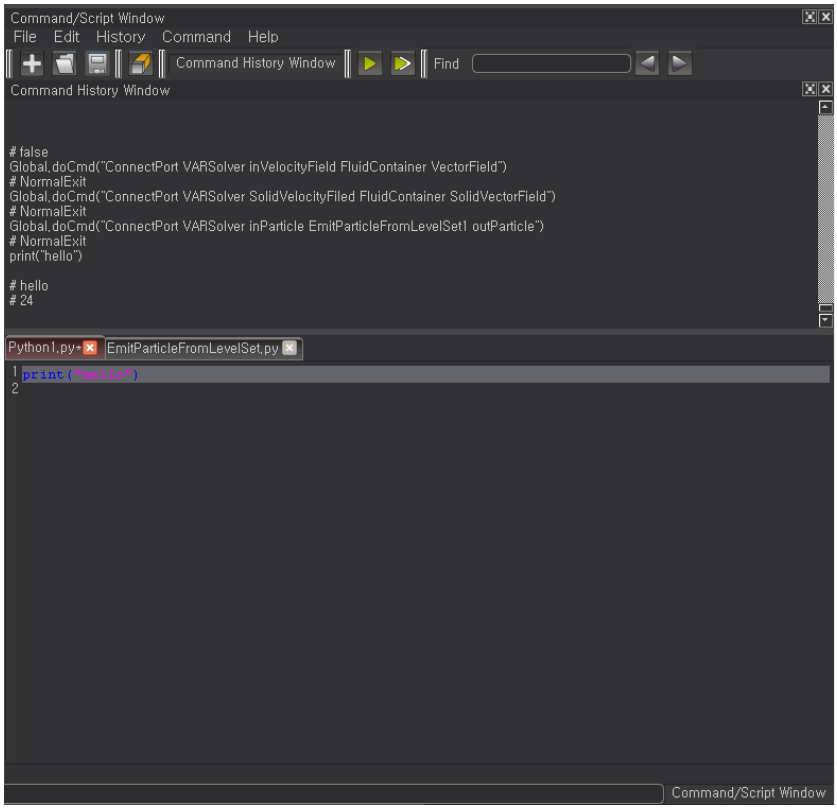
&

Tutorials

Python in Flux

FluX supports Python script. This chapter introduces Python commands for controlling FluX.

Python commands work in the input window of “Command/Script Window”.



Click the button above the play button in the bottom right corner to bring up the “Command/Script Window”.

Command format

Global.doCmd(“command option option ...”)

Script Commands

Command: "ConnectPort"

Syntax: "ConnectPort NodeA Port1 NodeB Port2"

Function: Connects NodeA's Port1 and NodeB's Port2.

Example: Global.doCmd("ConnectPort VARSolver inParticle Particle Particle")

Command: "CreateNode"

Syntax: "CreateNode NodeType (NodeName)"

Function: Creates a node that matches with the inputted node type as parameter. The second parameter will be the name of node. (Optional)

Example: Global.doCmd("CreateNode Particle MyParticle")

Command: "DeleteNode"

Syntax: "DeleteNode NodeName"

Function: Deletes the node that has the matching name with Nodename

Example: Global.doCmd("DeleteNode ImplicitSphere1")

Command: "DisconnectPort"

Syntax: "DisconnectPort NodeA Port1 NodeB Port2"

Function: Deletes the link between NodeA's Port1 and NodeB's Port2.

Example: Global.doCmd("DisconnectPort Particle Particle DragParticle inParticle")

Command: "GetConnectedPorts"

Syntax: "GetConnectedPorts NodeA Port1"

Function: Returns the nodes and ports linked to Port1 of NodeA in Node.Port format.

Example: Global.doCmd("GetConnectedPorts FluidContainer CellSize")

Command: "GetInputPorts"

Syntax: GetInputPorts NodeA

Function: Returns the list of NodeA's all input ports.

Example: Global.doCmd("GetInputPorts VARSolver")

Command: "GetNodeBoundingBox"

Syntax: "GetNodeBoundingBox NodeA"

Function: Returns the bounding box of geometries that NodeA displays. Only works with displayable nodes.

Example: Global.doCmd("GetNodeBoundingBox BoxShape")

Command: "GetNodeHelpFileName"

Syntax: "GetNodeHelpFileName NodeA"

Function: Displays the name of the Help file related to NodeA.

Example: Global.doCmd("GetNodeHelpFileName Global")

Command: "GetNodeQuery";

Syntax: "GetNodeQuery NodeA"

Function: Returns all the information of the node name, catergory, summary, all input/output ports and the data type list of each port etc.

Example: Global.doCmd("GetNodeQuery Root")

Command: "GetNodeType";

Syntax: "GetNodeType NodeA"

Function: Returns the type of NodeA.

Example: Global.doCmd("GetNodeType MeshShape")

Command: "GetOutputPorts";

Syntax: "GetOutputPorts NodeA"

Function: Returns the output port list of NodeA.

Example: Global.doCmd("GetOutputPorts MeshShape")

Command: "GetPortList"

Syntax: "GetPortList NodeA"

Function: Returns the list of all input/output ports of NodeA.

Example: Global.doCmd("GetPortList ParticleShape")

Command: "GetPortType";

Syntax: "GetPortType NodeA Port1"

Function: Outputs the type ID of Port1 port of NodeA.

Example: Global.doCmd("GetPortType ParticleShape Color")

Command: "GetSingletonData"

Syntax: "GetSingletonData NodeA Port1"

Function: Returns the Port1 value of NodeA when it is one-dimensional variable.

Example: Global.doCmd("GetSingletonData Global CFL")

Command: "IsInputPort";

Syntax: IsInputPort NodeA Port1

Function: Checks if Port1 of NodeA is an input port and returns 'True' if it is, 'False' if not.

Example: Global.doCmd("IsInputPort Camera Pivot")

Command: "IsSceneNode";

Syntax: IsSceneNode NodeA

Function: Checks if NodeA is SceneNode and returns 'True' if it is, 'False' if not.

Example: Global.doCmd("IsSceneNode MeshShape")

Command: "RenameNode";

Syntax: RenameNode NodeA NodeB

Function: Changes the node name that is named as NodeA to NodeB

Example: Global.doCmd("RenameNode Cache1 HoHo")

Command: "SetSingletonData";

Syntax: "SetSingletonData NodeA Port1 ValueSet"

Function: Sets the Port1 value of NodeA' as the value of ValueSet.

Example: Global.doCmd("SetSingletonData VARSolver Gravity 0 -9 0")

Command: "TogCacheMode";

Syntax: "TogCacheMode true/false"

Function: Changes the Cache mode to : true=Wirte->Read, false=Read->Write

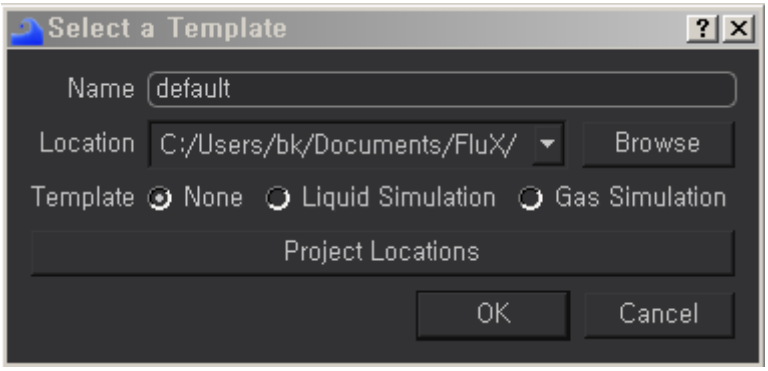
Example: Global.doCmd("TogCacheMode false")

Using MPI Simulation

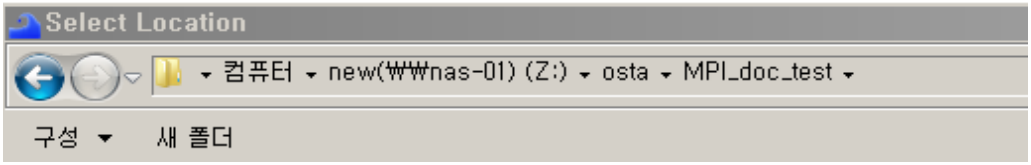
A. Basic Setting

i. Create a new project in the public storage where the simulation farm is accessible. (Menu>File>New Project)

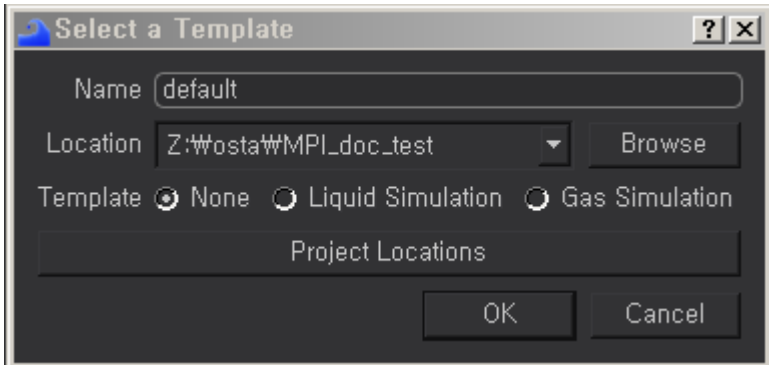
1. Select the browser.



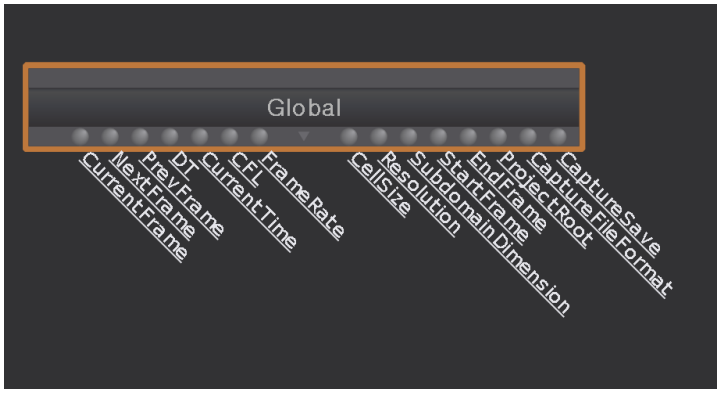
i. Create a folder of public storage which is accessible from the farm and press the OK button.



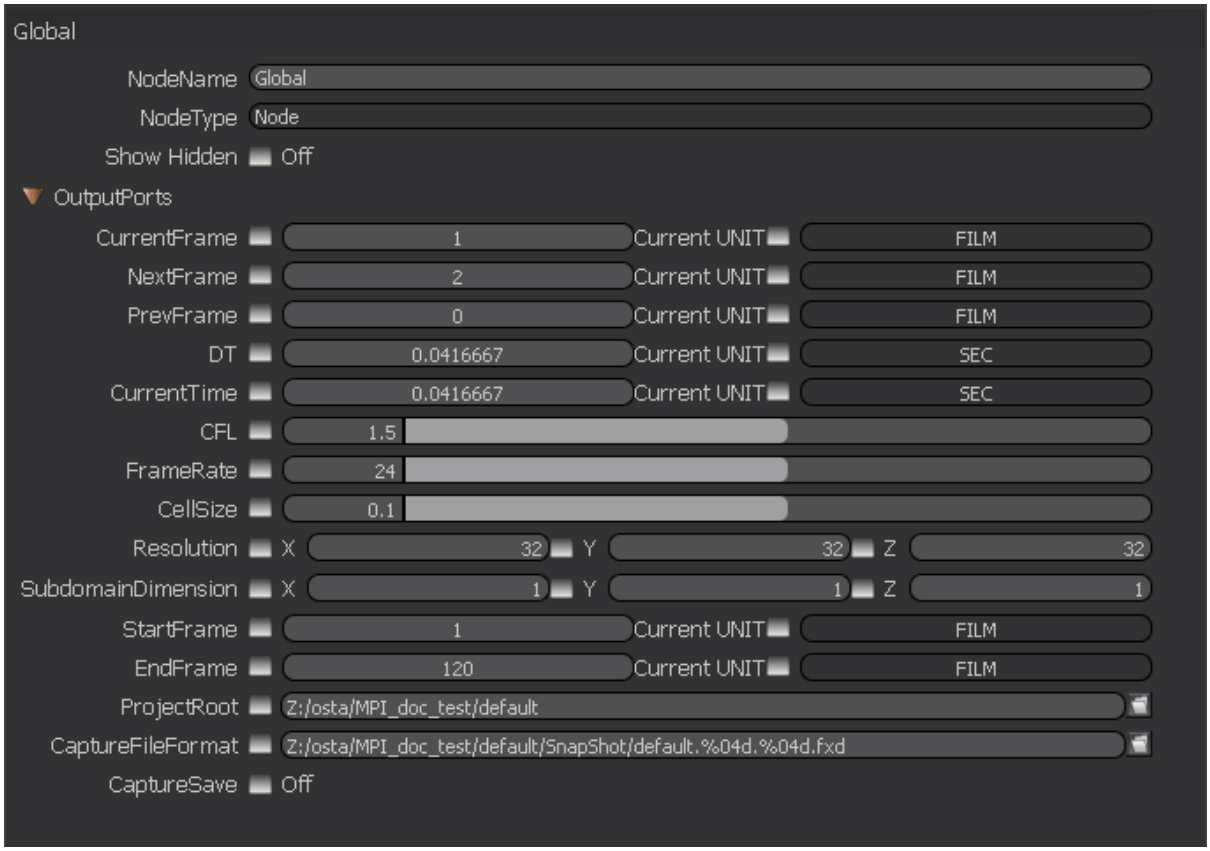
ii. Type the name of the project and press OK.



iii. Set up the Global node.



iv. Double click on the Global node to activate the parameter window.

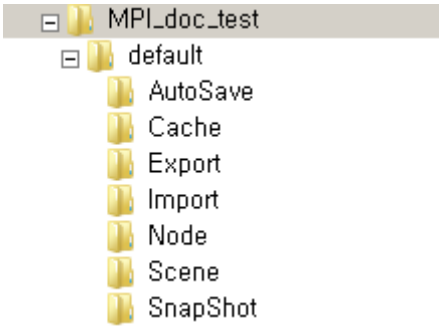


v. Change the path to the network (universal) path.

Z:/osta/MPI_doc_test/default -> //nas-01/new/osta/MPI_doc_test/default

vi. Besides the main path of the Global node, it is recommended to use the relative path based on the root directory of the project

The project directory structure



"default" - ProjectRoot

"AutoSave" - AutoSave path

"Cache" – Simulation Cache path

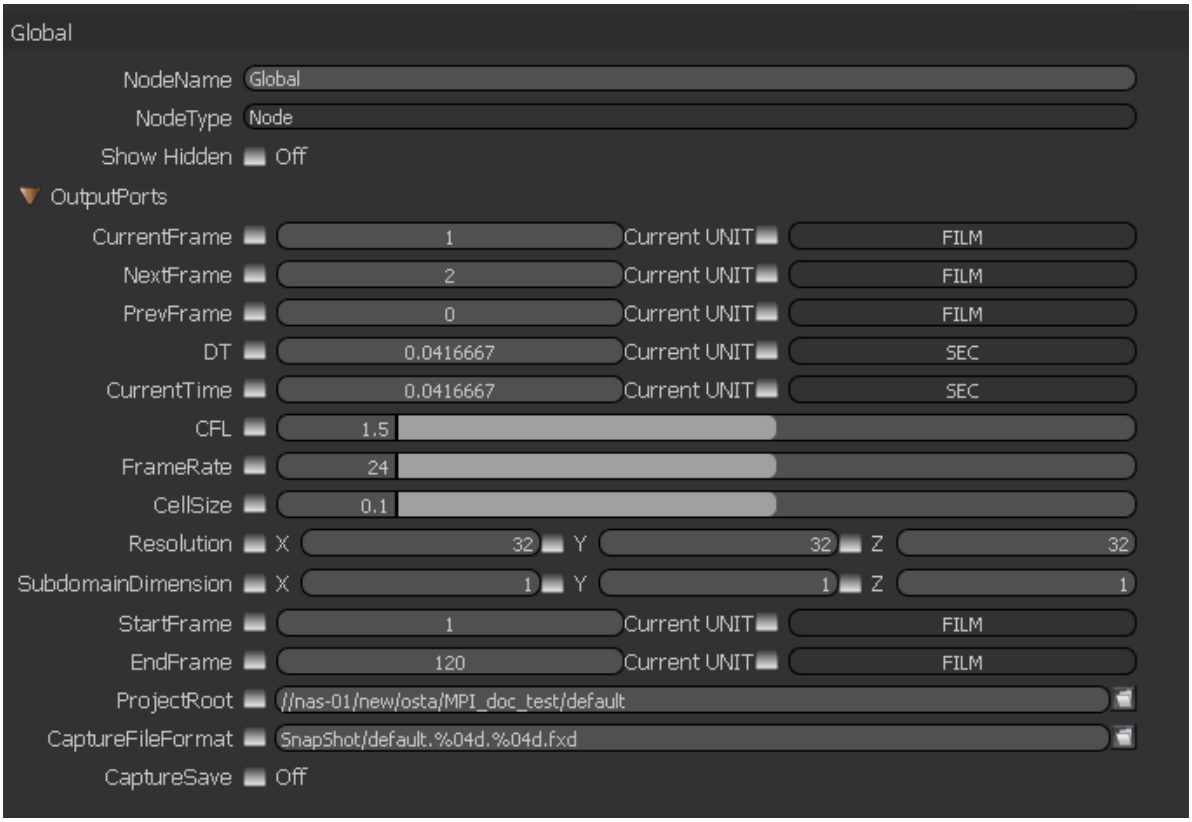
"Export" – Saving path of gathered result

"Import" - External geometry file path for simulation

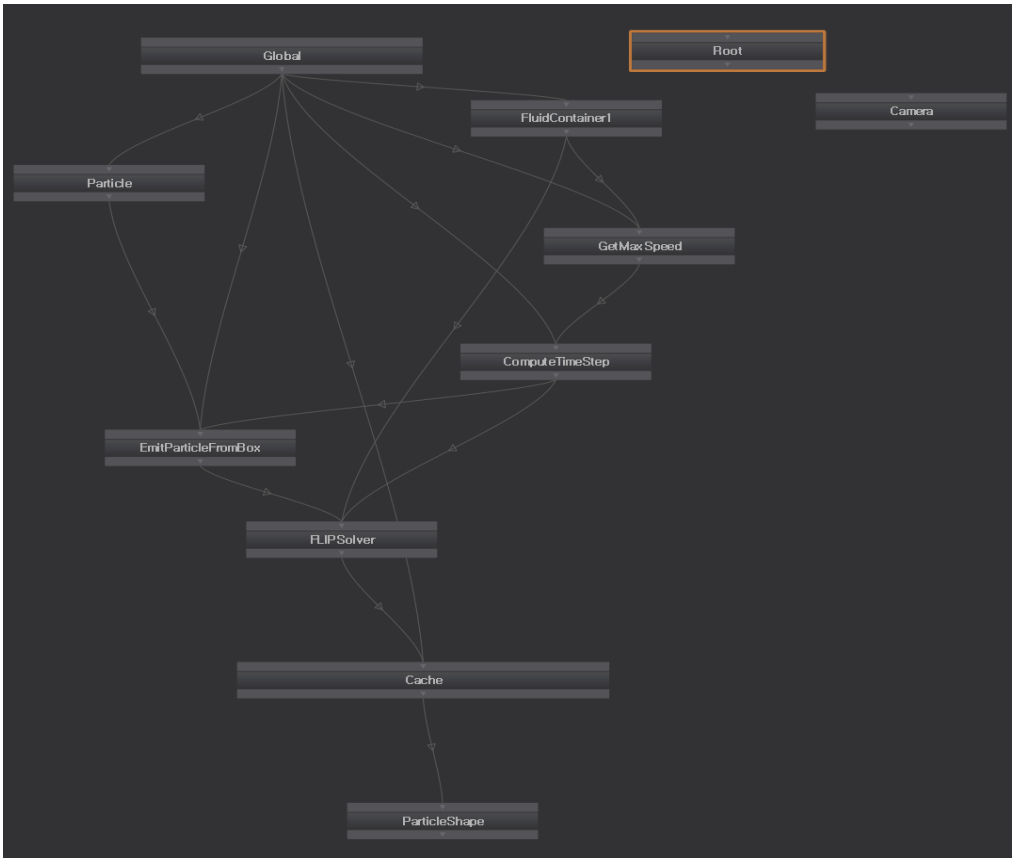
"Scene" – Saving path of simulation scene file

"SnapShot" – Saving path of snapshot image

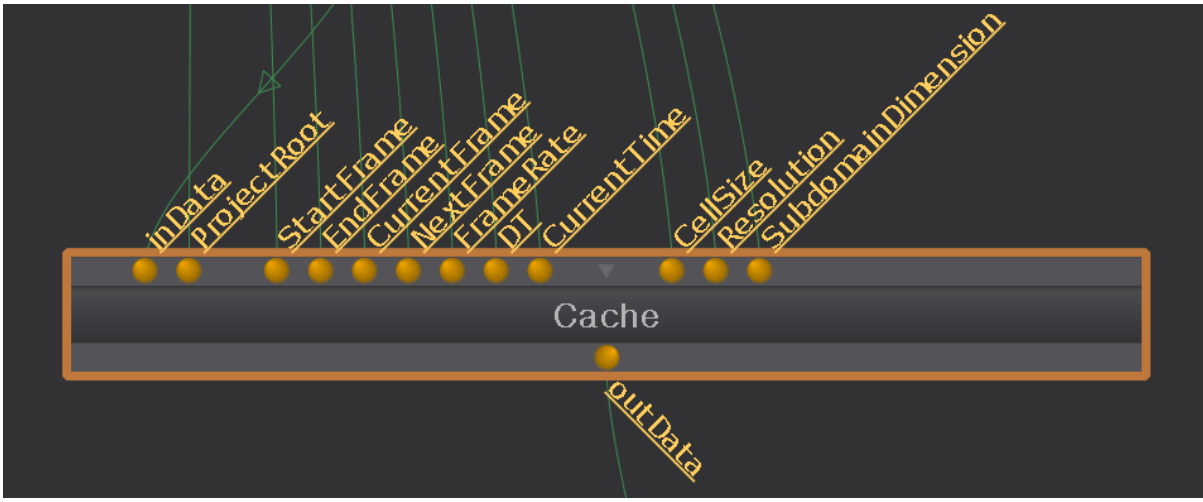
vii. The following setting is the changed setting of the global node.



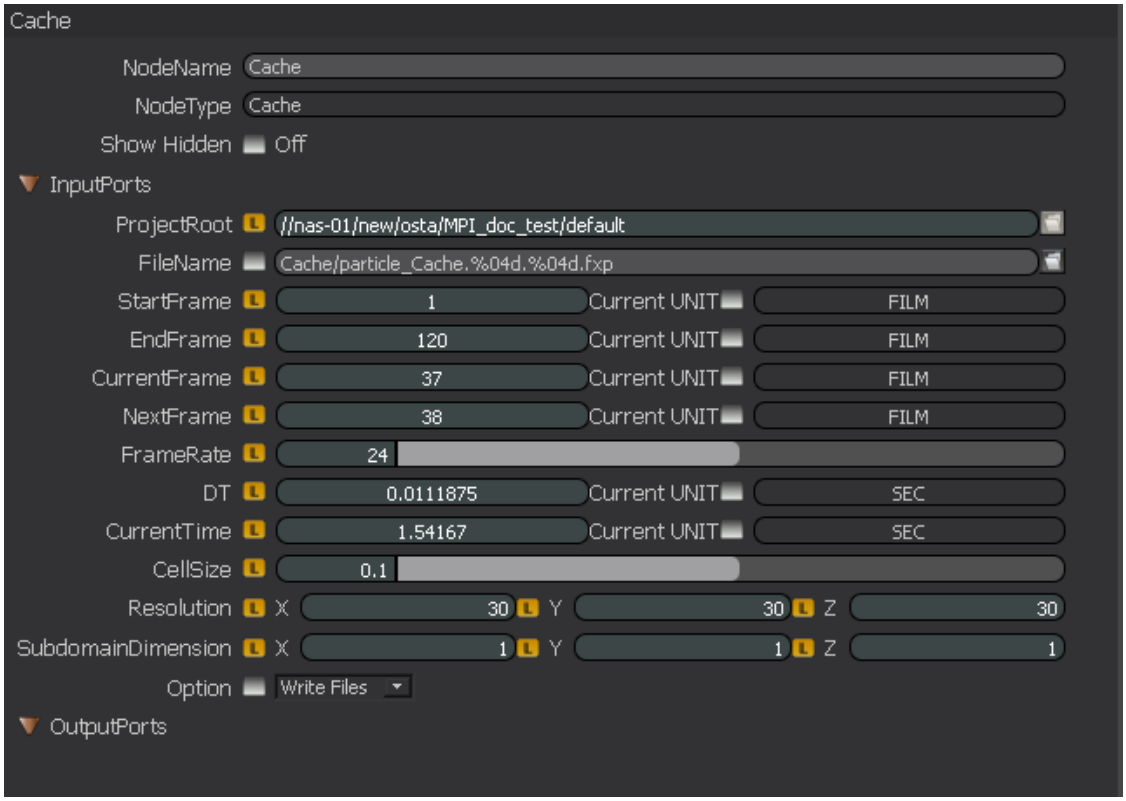
viii. Set up the simulation network.



ii. Set up the Cache node for saving.

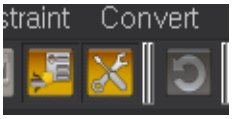


iii. Set up the path of particle cache of the Cache node. (Verify if the mode is in Write mode when you proceed with the simulation.)



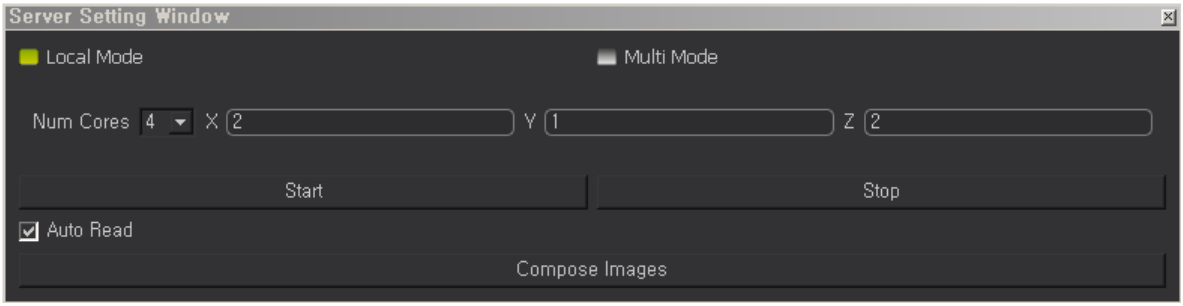
B. MPI setting

i. Bring up Server Setting Window by clicking on the Server Setting Window button.

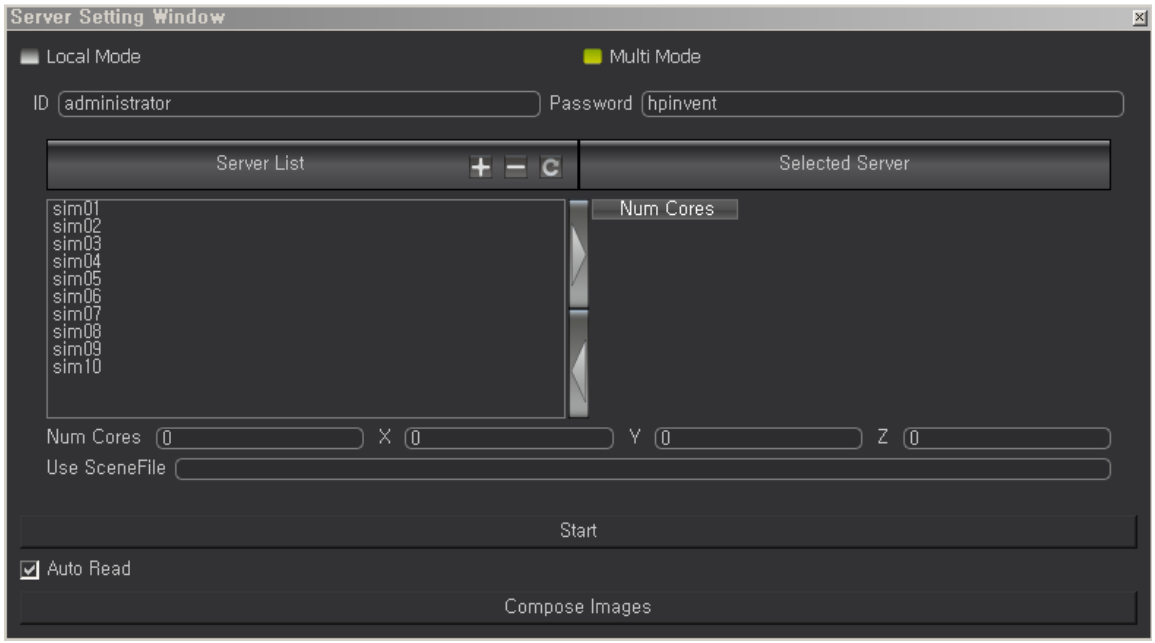


ii. Server Setting Window

1. Local mode

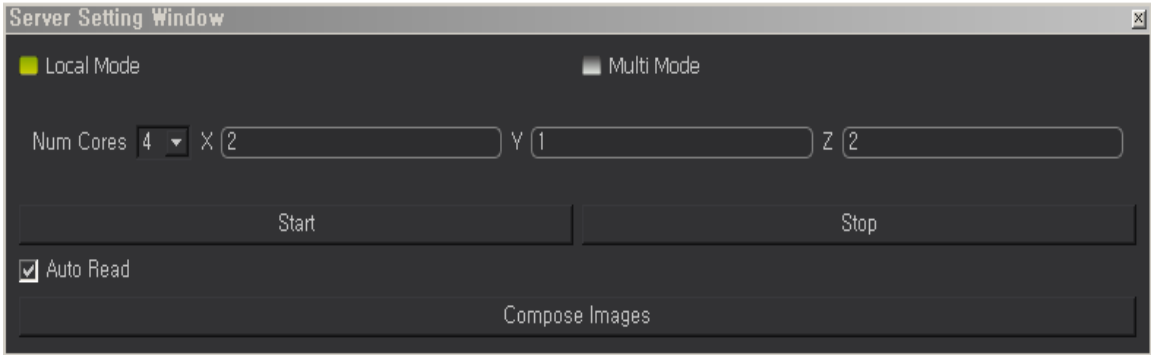


2. Multi mode



iii. MPI setting – local mode

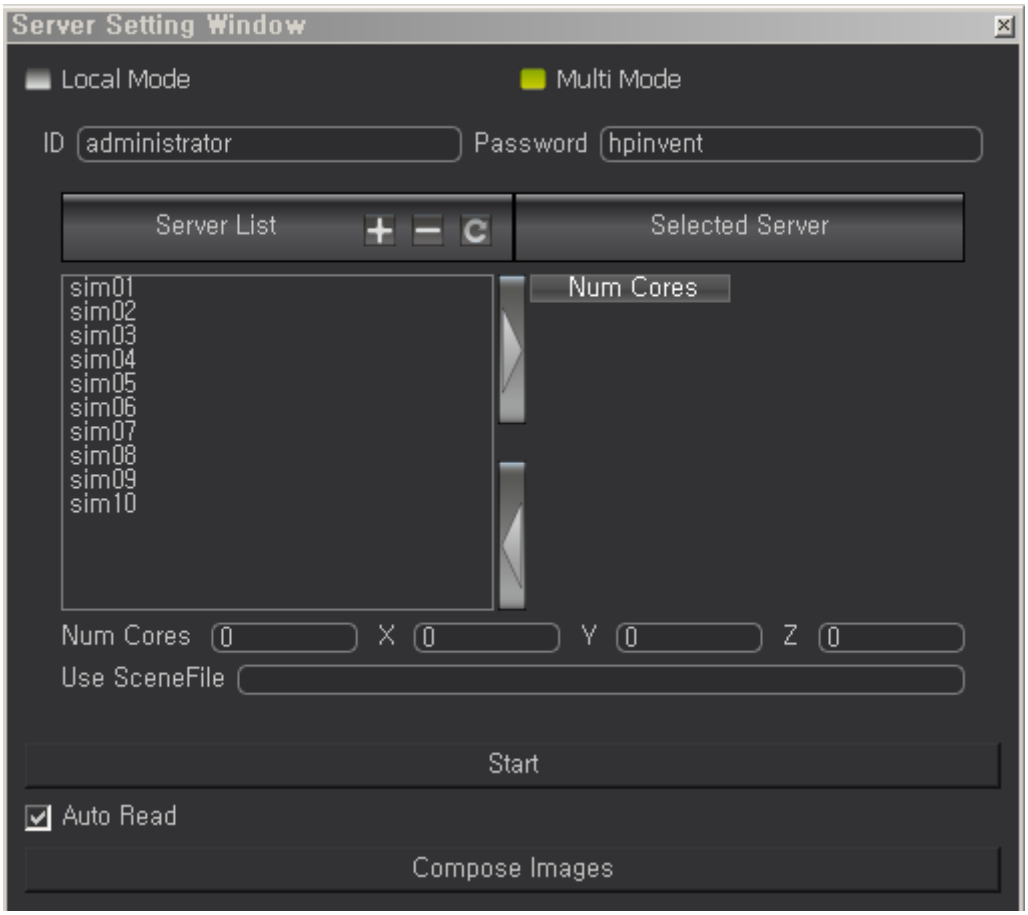
1. Setting the number of cores – Set the number of cores of the workstation (the current version does not support the multi thread. Therefore, set the number of physical cores as default.)



2. Adjust the simulation area to suit the situation
   X: 2, Y: 1, Z: 2

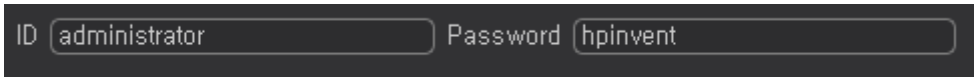
iv. MPI Setting – distributed processing mode

1. Registering the server – register the server that will be used for distributed processing

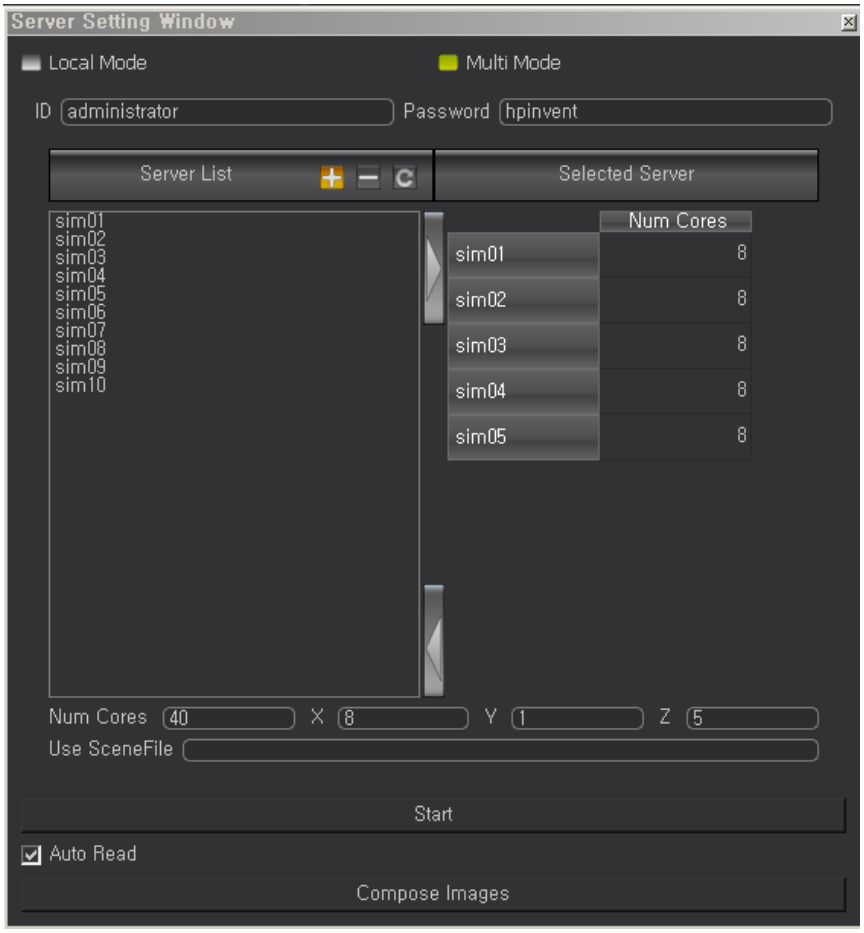


2. Manually enter the address of the server, or type the name of server to register.

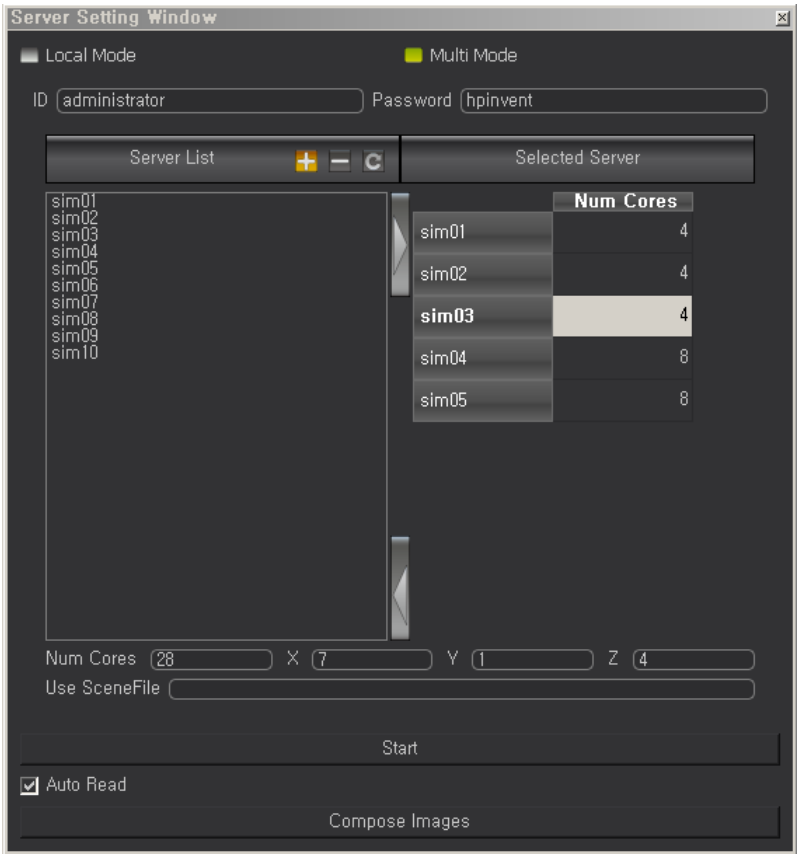
3. ID/Password setting – for all the servers those will be used must have the same ID and Password.



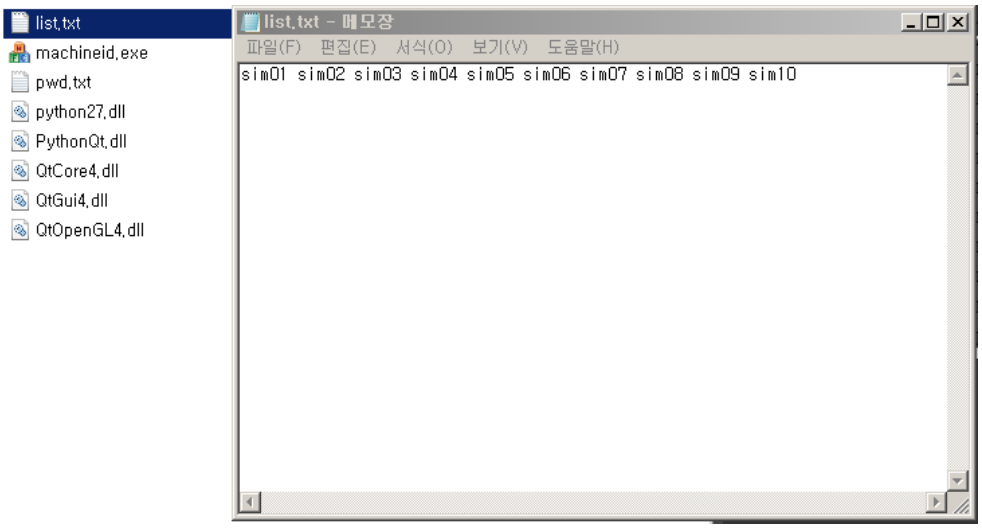
4. Selecting the server - once the registration is completed, select the servers to use for the simulation.



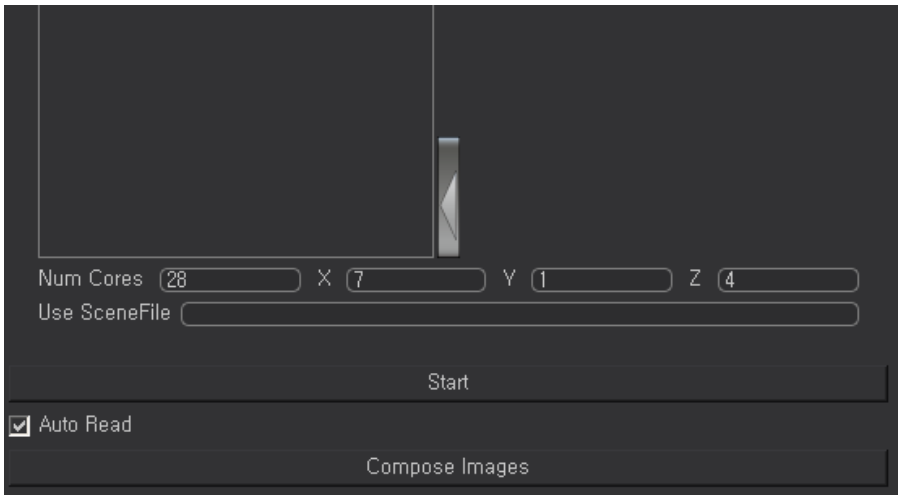
5. Core setting – enter the number of physical core of each server.



6. Saving the server setting – after setting the simulation server, press the Start button. This will save the server list as "list.txt" in FluX installation directory.



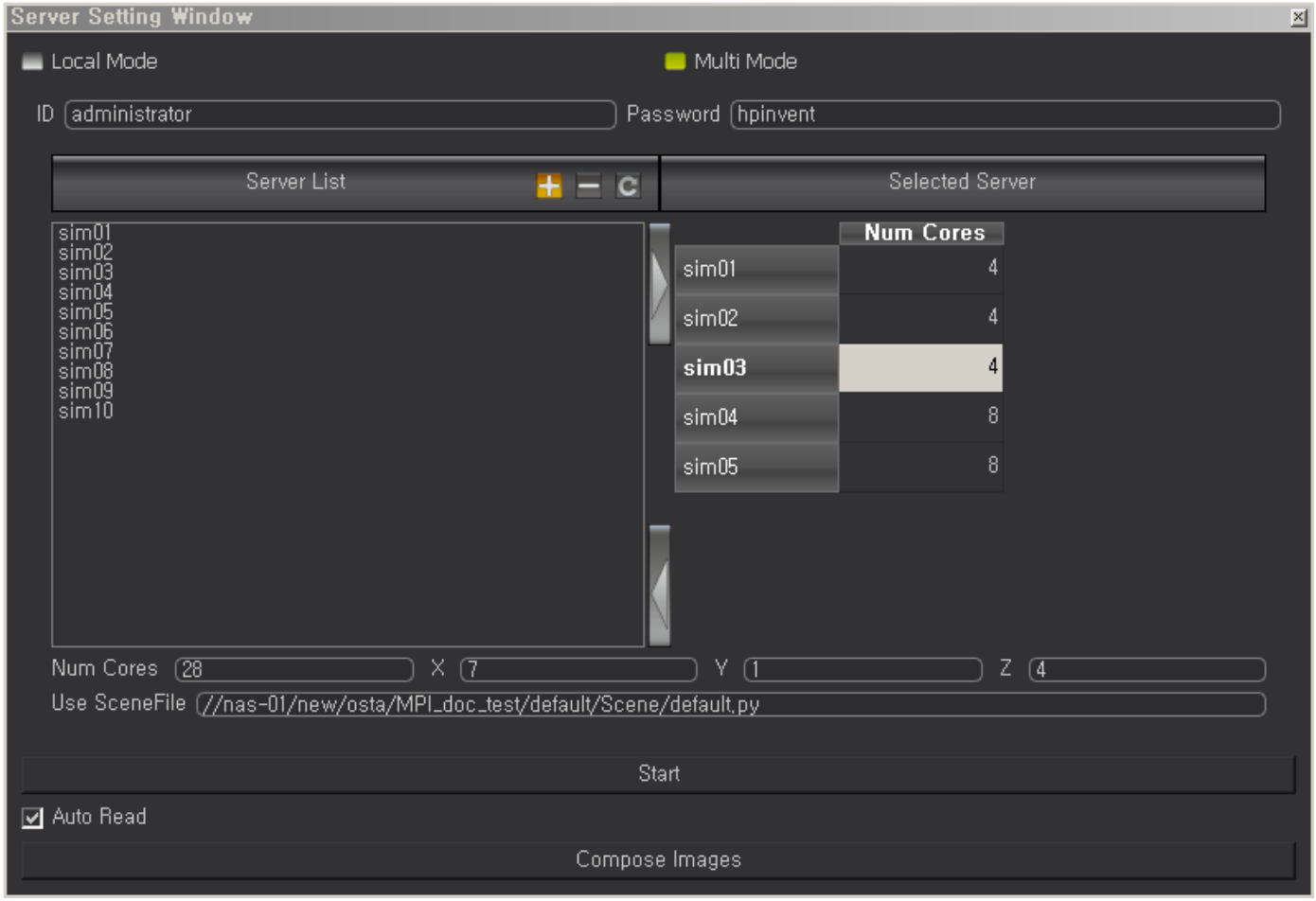
7. Subdomain setting – set up the appropriate subdomain to suit the form of the scene



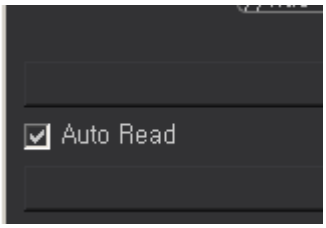
C. Scene file setting

i. Set the name of the scene file that will use MPI to proceed with the simulation.

ii. You must use the universal path.

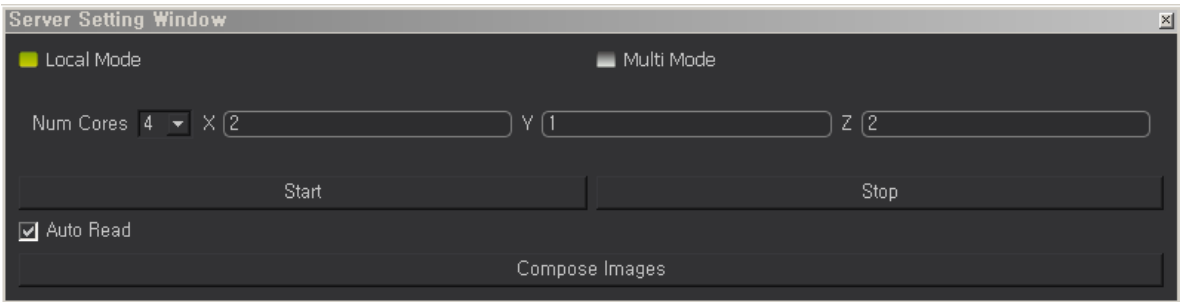


iii. Auto Read option – Auto Read also updates Viewport as the simulation proceeds. It is good to see the result but the simulation will run slower so it is recommended that you turn it off if not necessary.

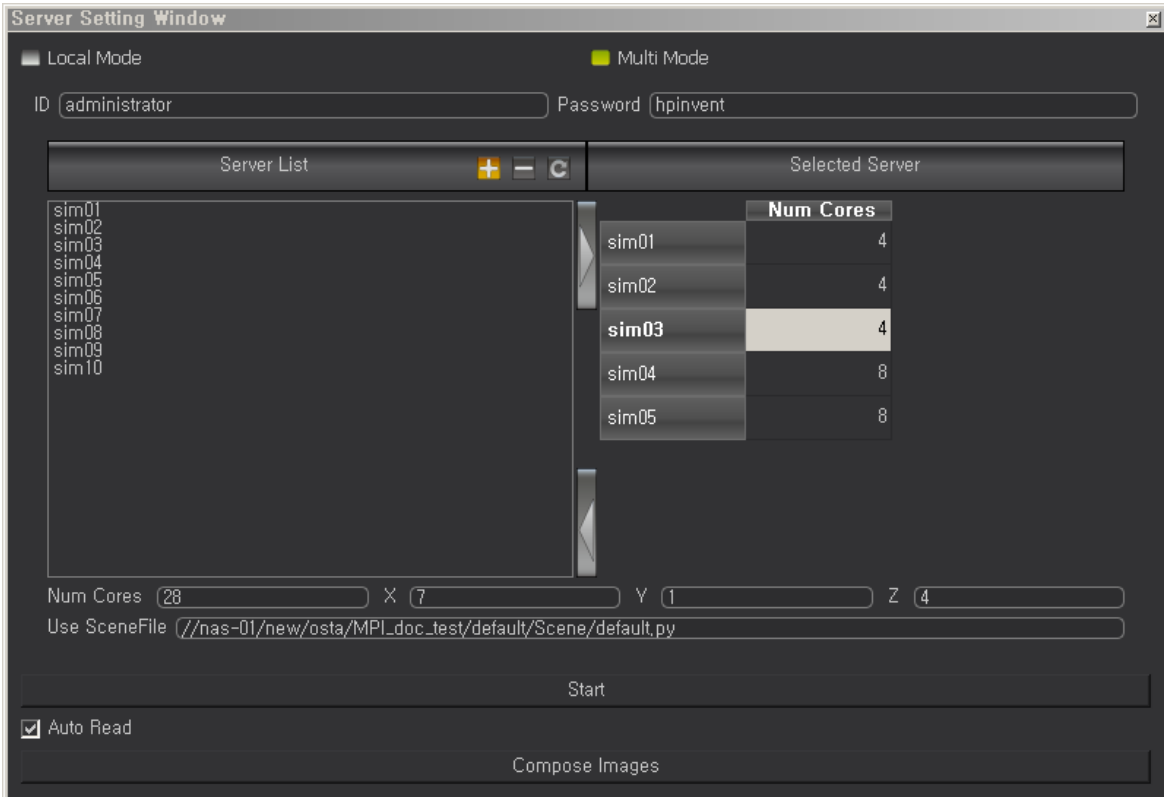


iv. Proceeding with MPI simulation - When all settings are completed, press the Start button to proceed with (MPI) distributed simulation

1. Local mode

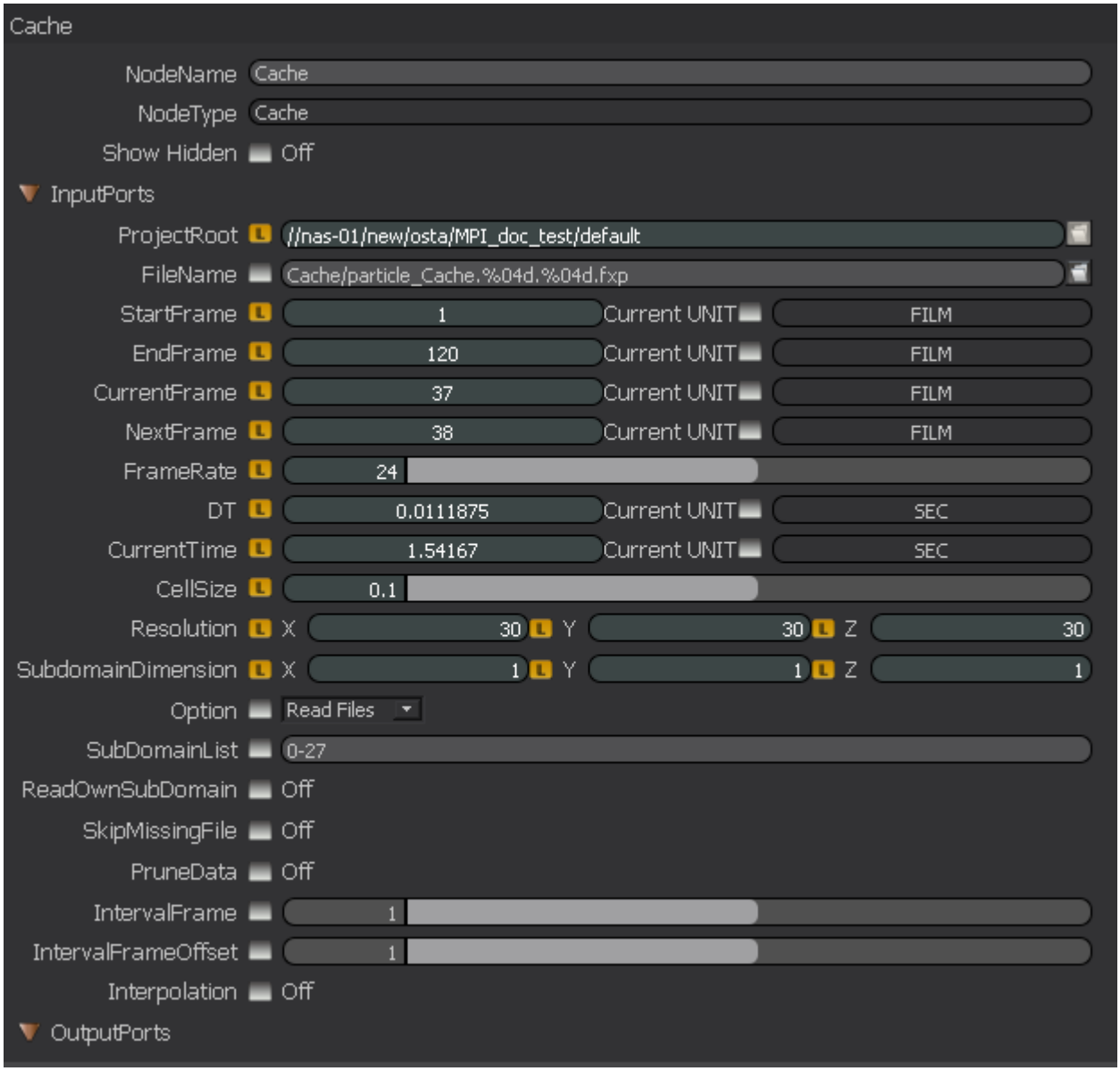


2. Multi mode



D. Gathering the result

i. The result can be gathered by selecting the Read option. (the file extension of the particle = fxp and the mesh = tri)



ii. When gathering the result, set the subdomain list as if 8 cores = 0-7, 28 cores = 0-27.

E. Export

i. Connect the Cache node to the gathered result and set the saving path, then press the Play button to start exporting as a single file.
ii. When exporting the particle or mesh, using the Realflow format bin is recommended.

F. Things you need to beware of

1. Save the scene file before proceeding with the MPI simulation

2. If you need to stop the MPI simulation for any reason, you must use the Stop MPI button on the MPI simulation window.

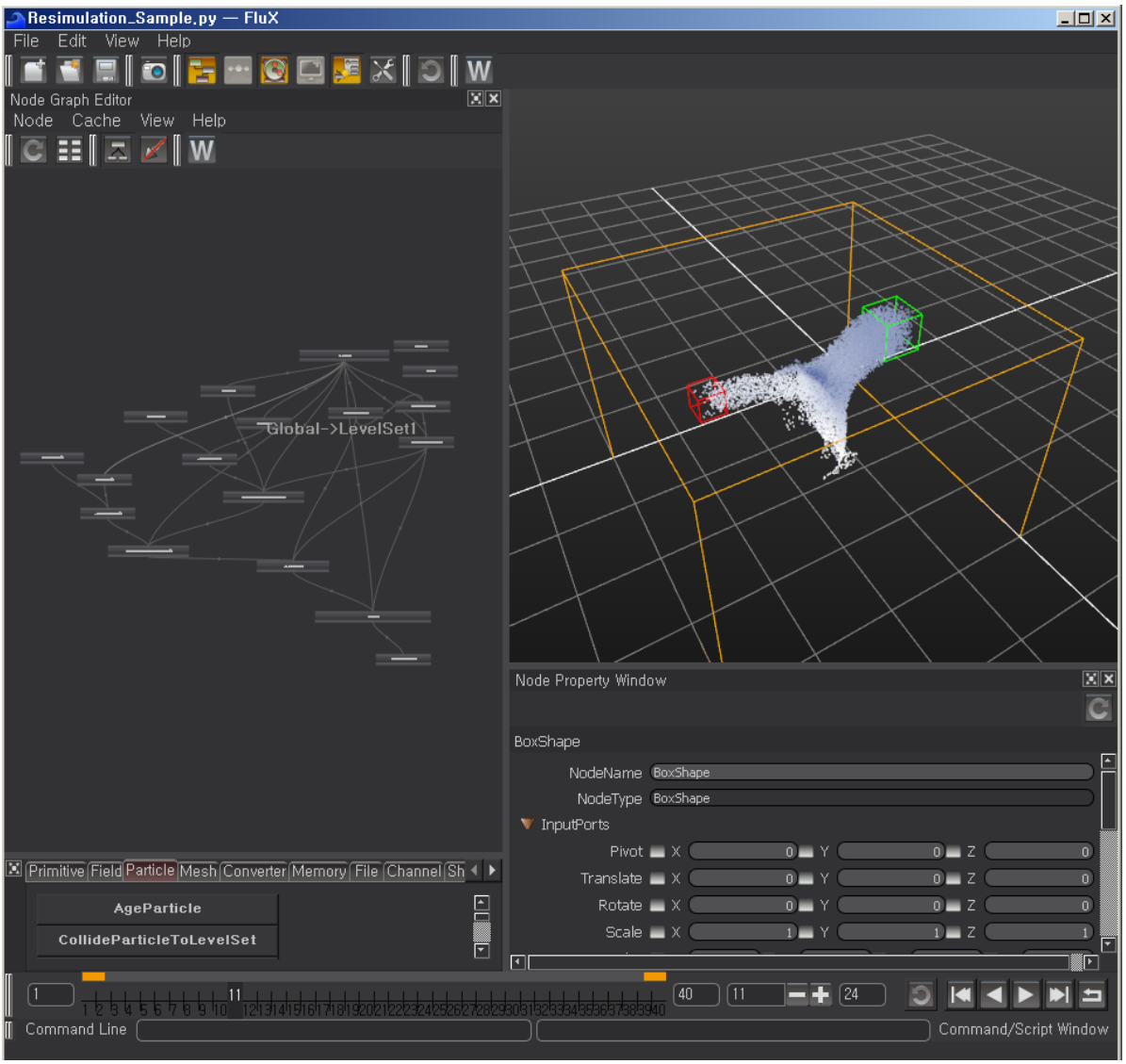
3. With the Cache node, the Write mode switches to Read mode after the simulation starts, so setting the Read/Write mode of each Cache when working on the simulation requires careful attention.

Resimulation

Flux Supports resimulation.

When simulation is interrupted or simulation range is expanded, it simply can continue with existing data from the former process with resimulation. Starts simulation from the beginning is not necessary. (Example file is available) Also, it can be used when another node to be set up in the middle of simulation process.

Simulation

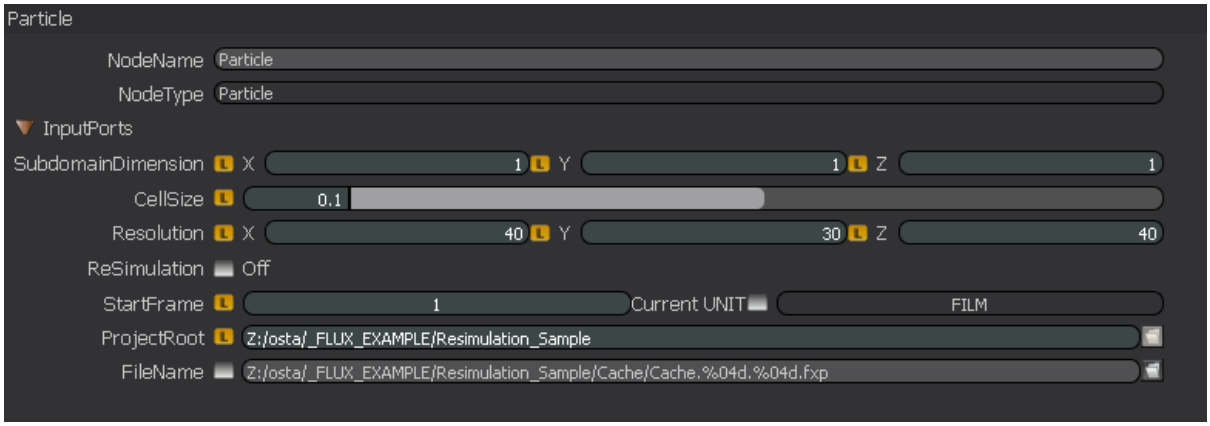


First, sets simple scene and process simulation. The example simulation file only lasts 40frames. (Example reference Resimulation_Sample.py)

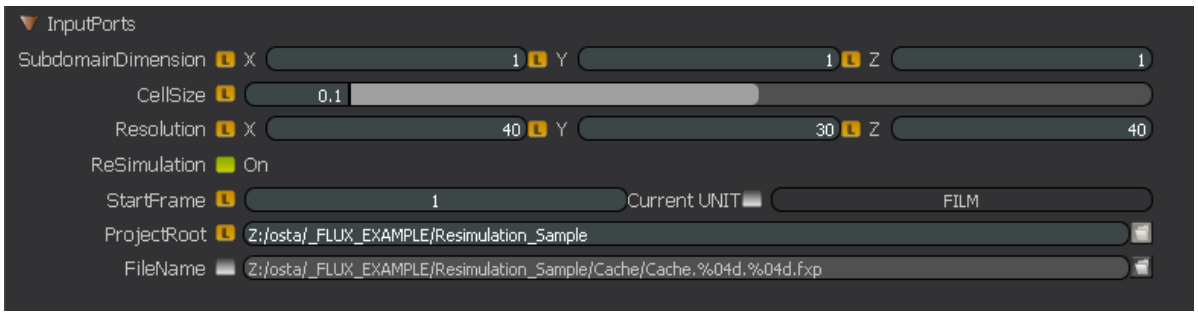
Resimulation

Processes resimulation after the end of the simulation frame. (Reference Resimulation_Sample.resimpy)

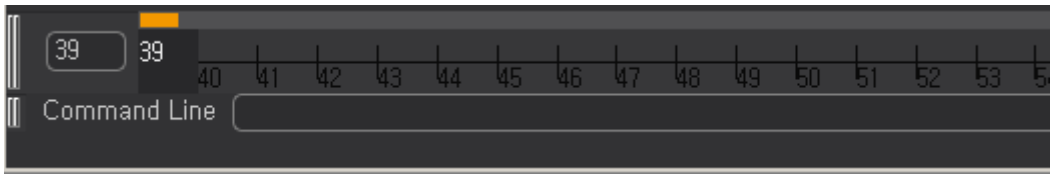
Opens node property window of particle node and address existing simulation data routes at FileName field.



1. Checks ReSimulation at node property window of particle node.



2. Sets time slider start frame as one frame before resimulation's starting frame.



3. Processes simulation with play button or MPI window.