

Cupid Tutorial

The ESMF software distribution contains a demonstration program called Coupled Flow. In this tutorial you will use the Cupid code generation tool to create the architectural skeleton of an application similar to the Coupled Flow demo.

For more information about the Coupled Flow demo itself, visit the ESMF website at the following link:

http://www.earthsystemmodeling.org/users/code_examples/external_demos/ESMF_CoupledFlow_ed.shtml

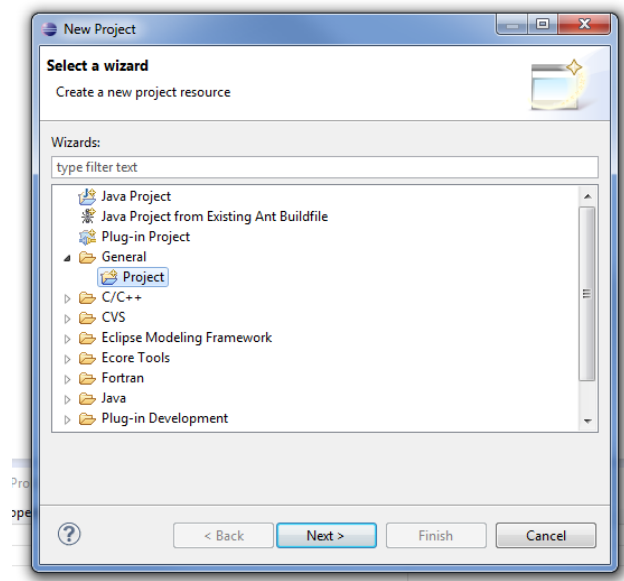
Before starting this tutorial, make sure you have installed Eclipse and the Cupid plugin. Installation instructions are available in a separate document.

During this tutorial you will specify a sample ESMF application. If desired, the completed specification that results from this tutorial can be downloaded from the following location:

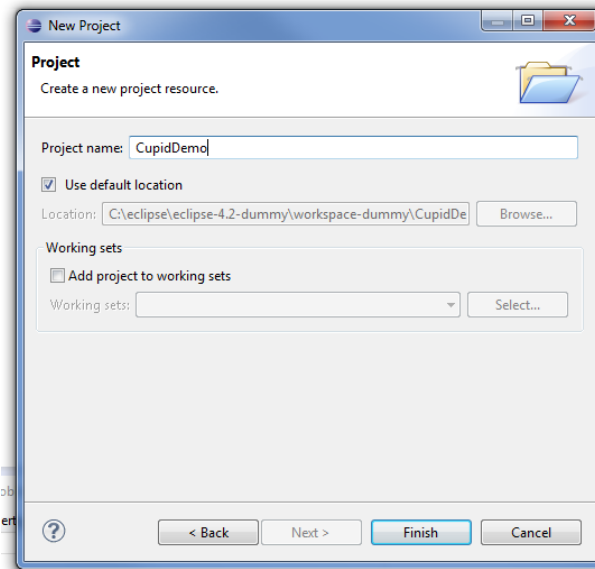
<https://raw.githubusercontent.com/rsdunlapiv/cupid/master/org.earthsystemcurator.cupid.esmf/instances/tutorial/CFlowWorkspace.esmf>

1. Create a new empty Eclipse project

- Create a new project in Eclipse. Select File→New→Project... Choose “Project” under the “General” category and click next.



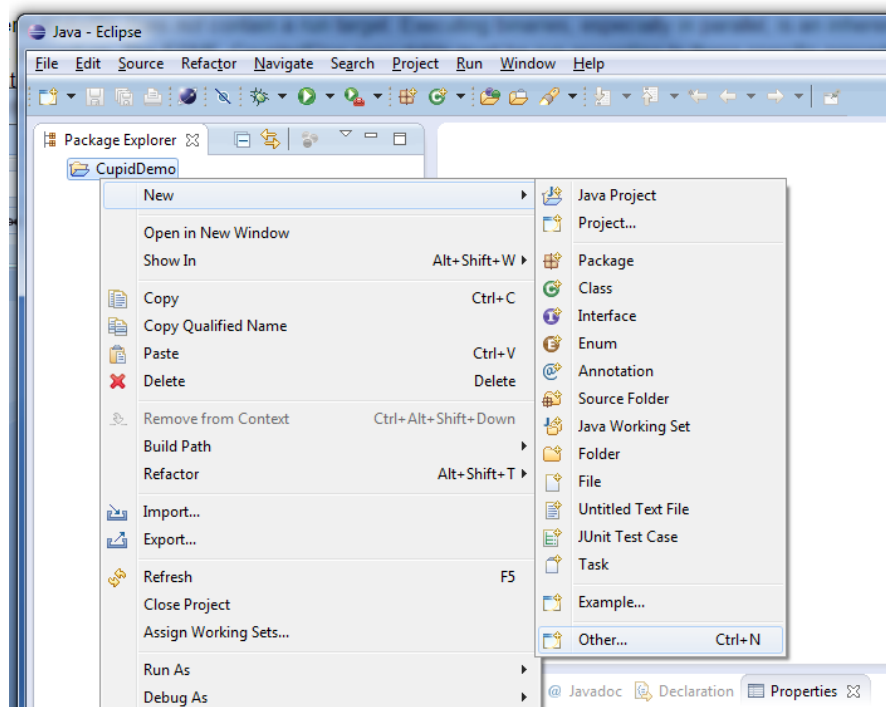
- Name the project “CupidDemo” and click Finish.



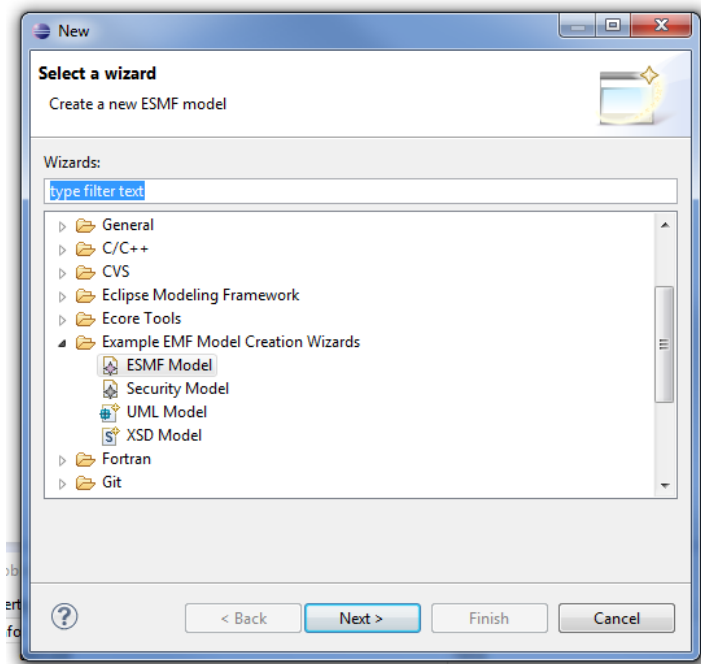
2. Create a new ESMF Workspace

You should now have an empty project folder named CupidDemo. We will now create an empty ESMF Workspace where gridded and coupler components and drivers may be added.

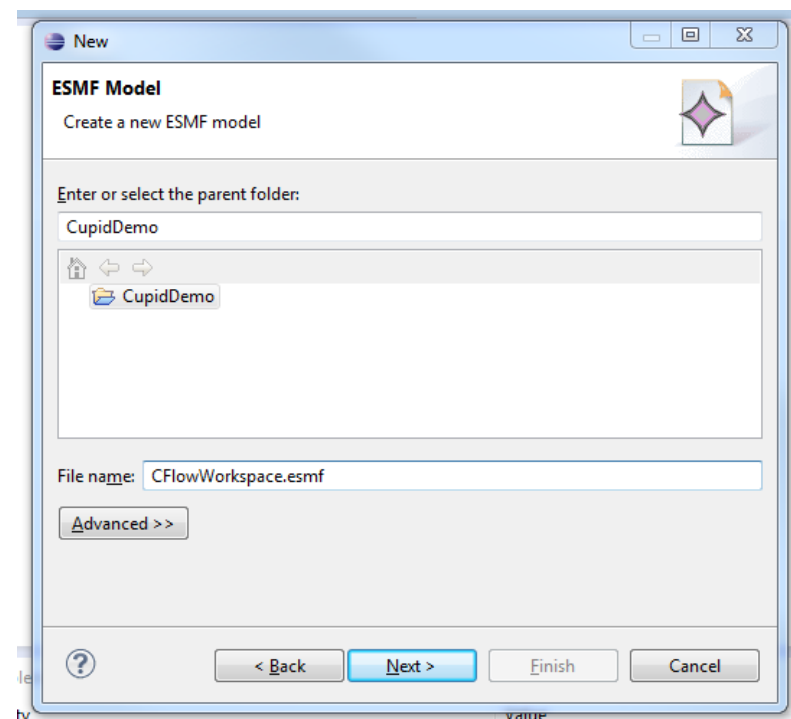
- Right click on the empty project and select New→Other...



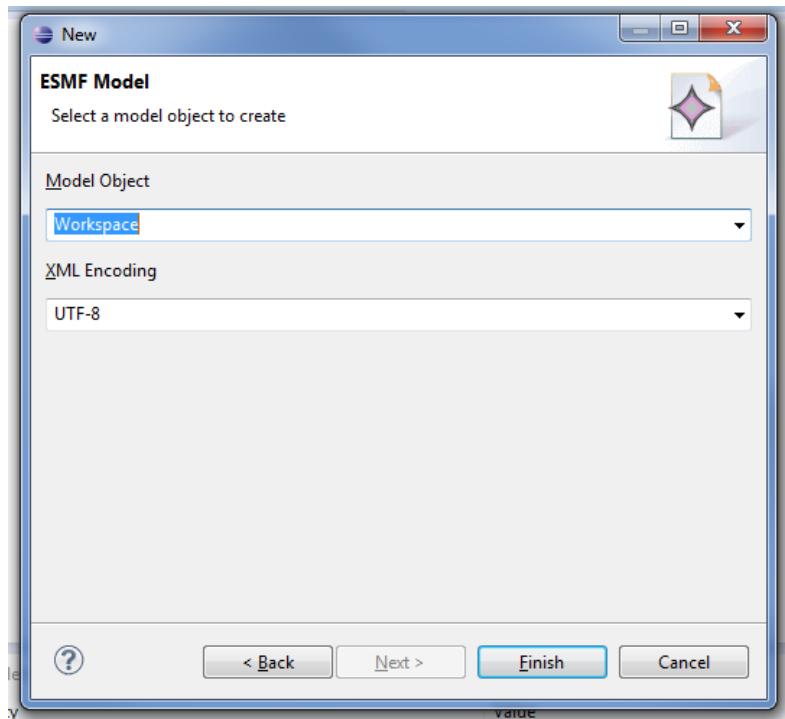
- Select “ESMF Model” under the folder “Example EMF Model Creation Wizards” and click Next. (A word about terminology: Don’t confuse EMF with ESMF. EMF stands for *Eclipse Modeling Framework*.)



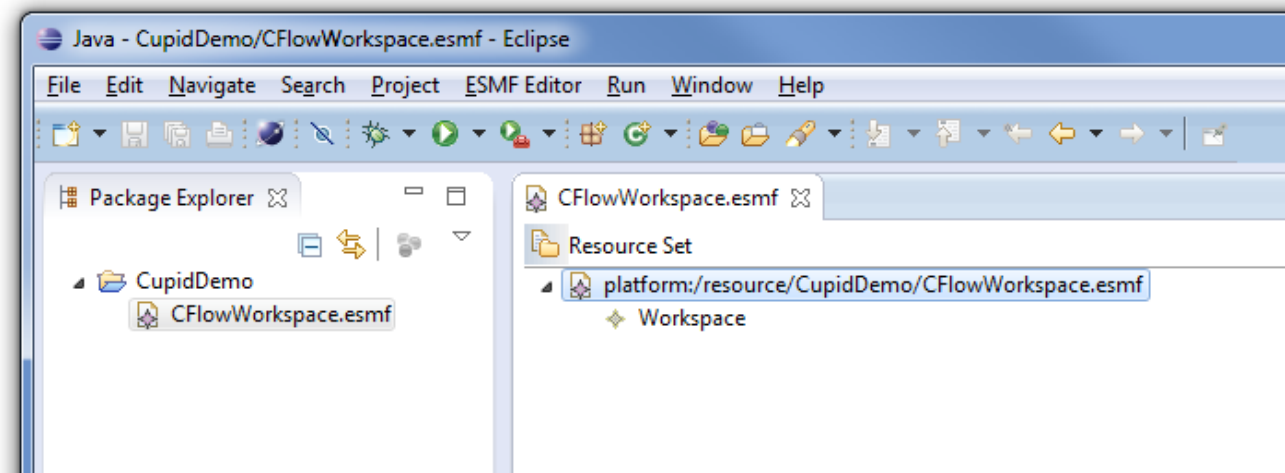
- Name the ESMF model file “CFlowWorkspace.esmf” and click Next.



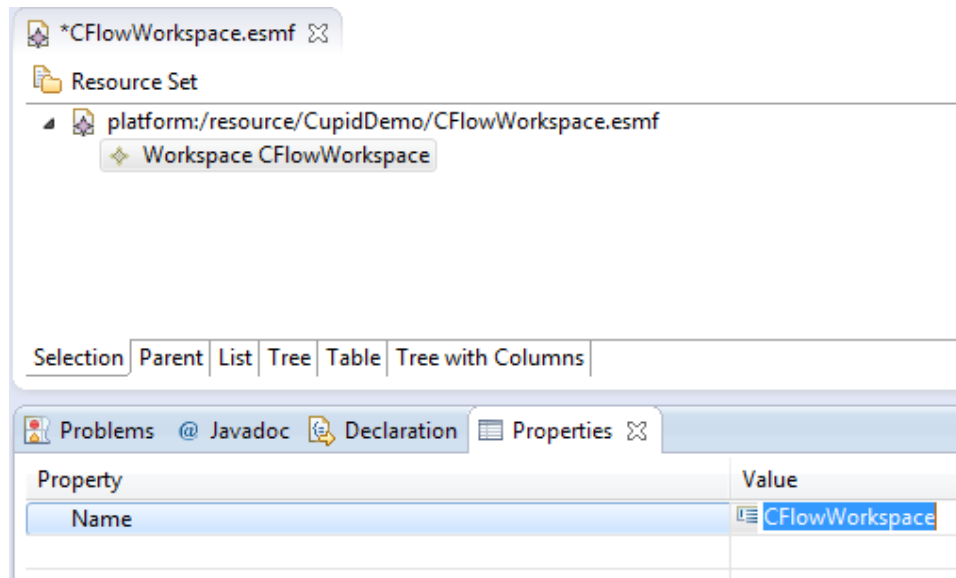
- You must choose which model object to create. Choose “Workspace” and click Finish.



- You should see the new file in the CupidDemo project folder and it should be open in the model editor. If the file does not open automatically, right click on CFlowWorkspace.esmf and select Open With→ESMF Model Editor. You should see an empty top-level element called Workspace.

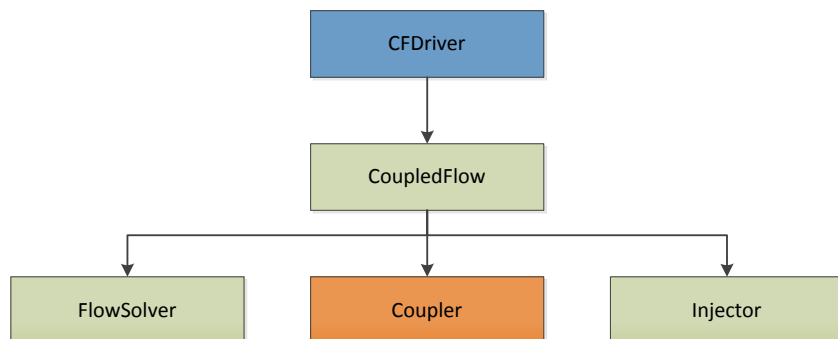


- A Workspace is a container for a set of ESMF components. Give the workspace a name by modifying the *name* property in the Properties View. To show the Properties View, right click on the Workspace node and choose “Show Properties View.” Click on the Workspace element and set the *name* property to “CFlowWorkspace” in the properties view. Then save the model file.

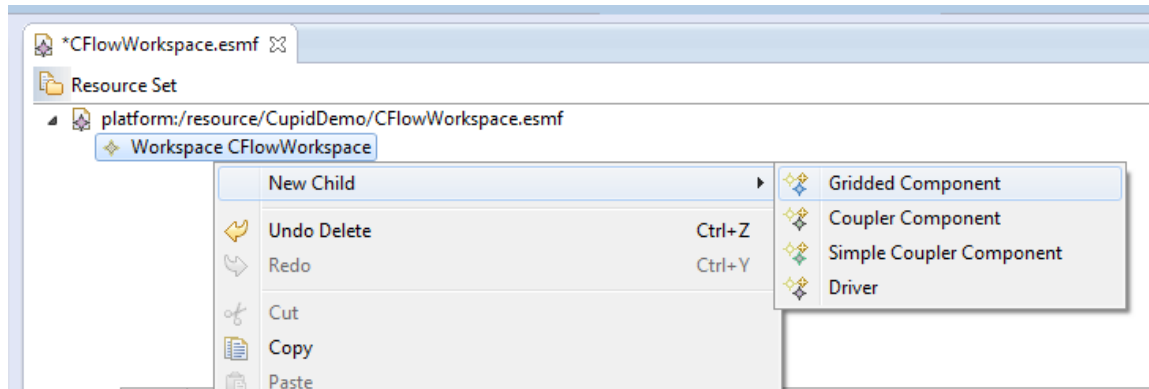


3. Add ESMF gridded components to the workspace

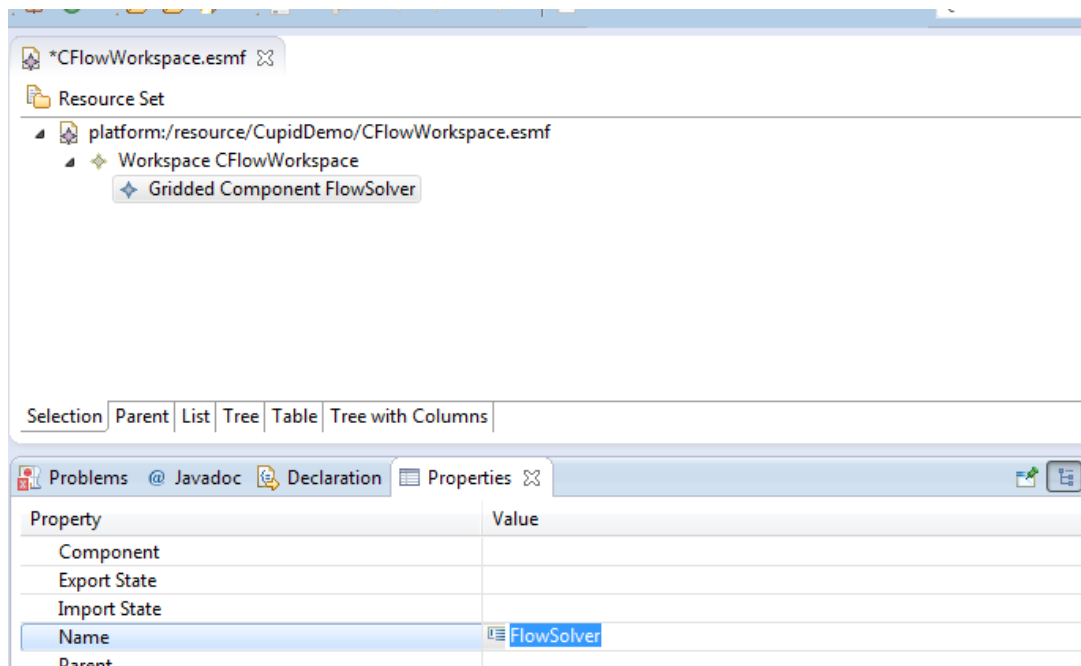
We will now add a series of components to the workspace that duplicates the architecture of the Coupled Flow demo. The architecture of the demo application is shown below. There is a top-level driver shown in blue, three Gridded Components shown in light green, and a Coupler Component shown in orange.



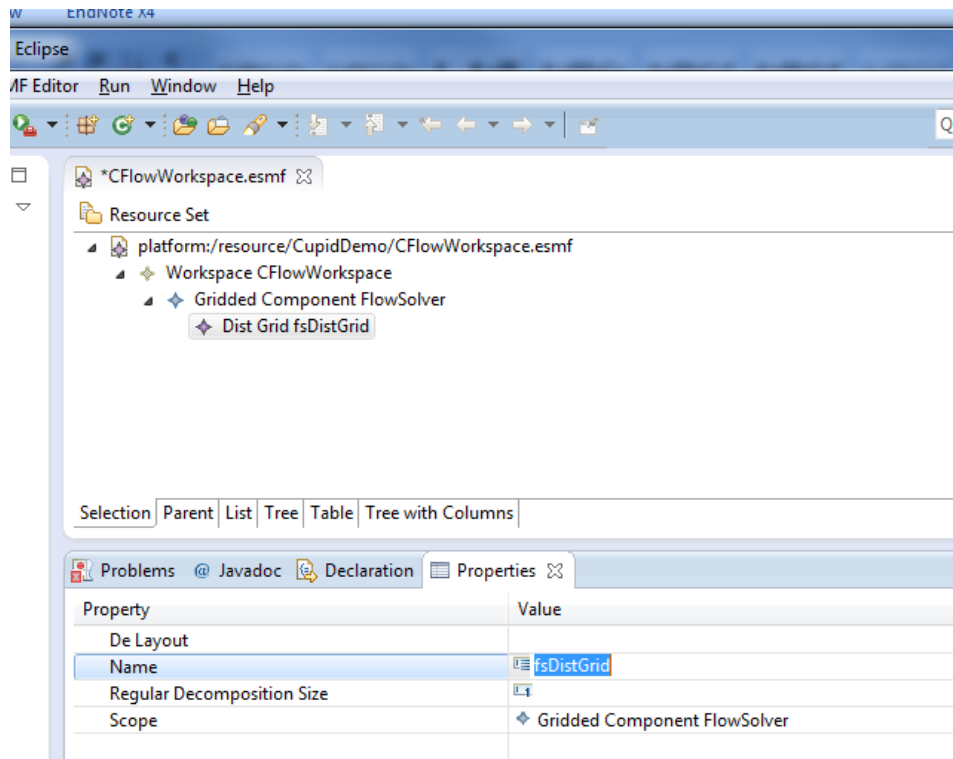
- Add a new Gridded Component to the workspace by right clicking on the Workspace element and selecting New Child→Gridded Component.



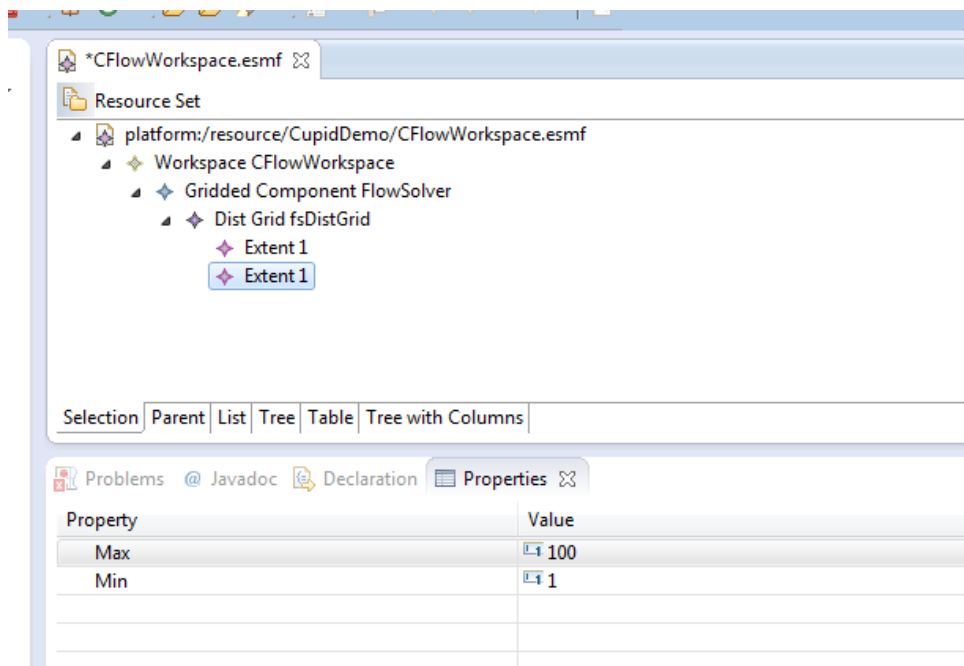
- Using the Properties View, set the *name* property of the Gridded Component to “FlowSolver”.



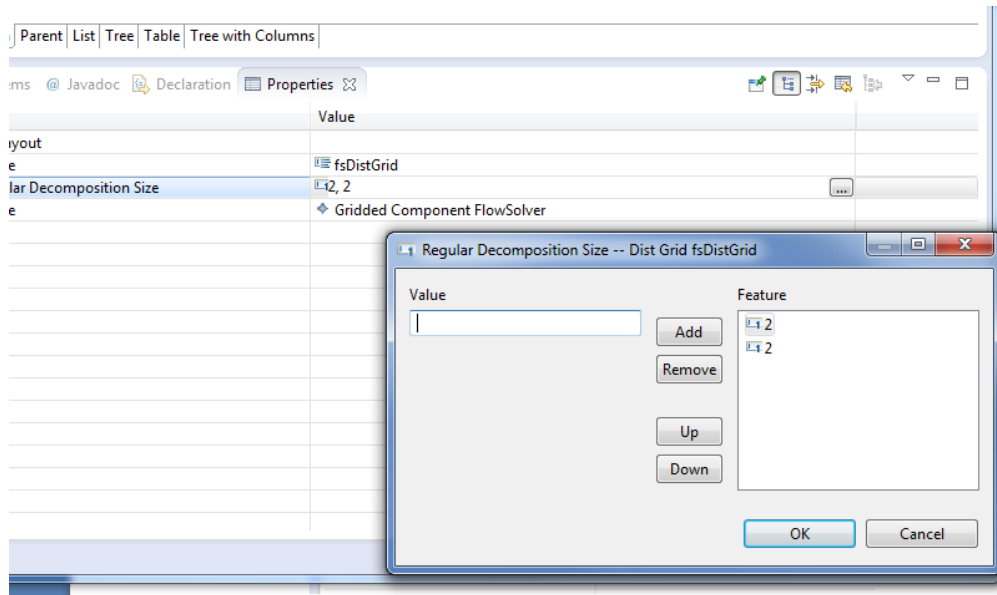
- We will now define some basic properties of the numerical grid for the FlowSolver component. We'll assume that the grid is a two-dimensional Cartesian grid that of size 100 x 100. First, add a DistGrid element to the FlowSolver component by right clicking and selecting New Child→Dist Grid. Name the Dist Grid object “fsDistGrid”.



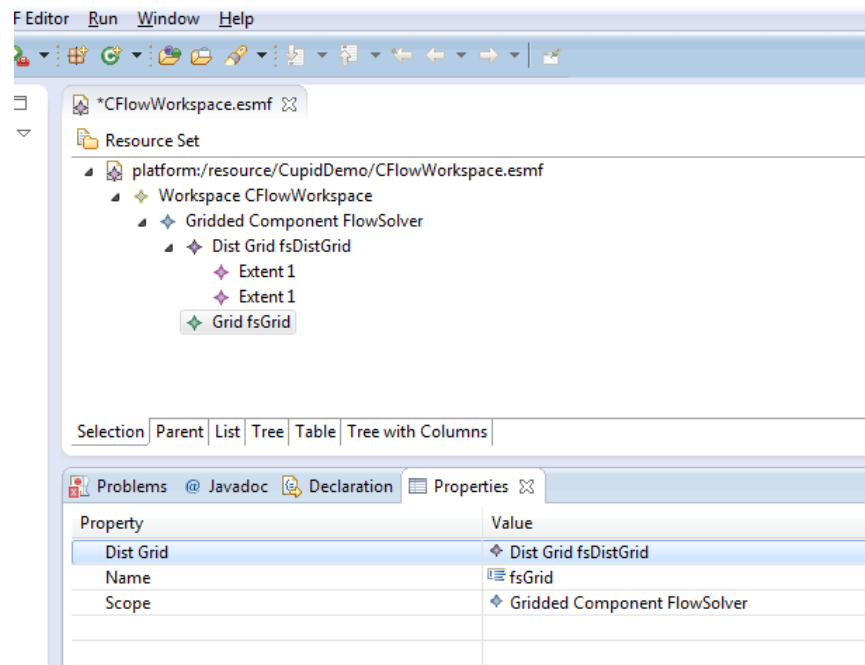
- Set the min and max indices of each dimension of the DistGrid by adding two child Extent elements to the DistGrid (Right click, Add Child→Extent). Set the *min* and *max* properties of both Extent objects to 1 and 100, respectively.



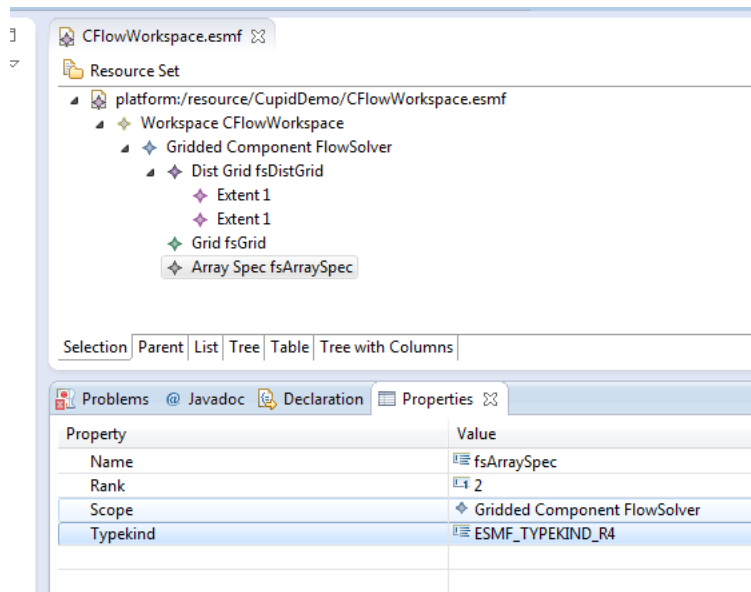
- Set the regular decomposition of the DistGrid to 2 x 2 (i.e., 2 decomposition blocks in the first dimension and 2 in the second). To do this, add the value “2” twice to the *Regular Decomposition Size* property.



- Add a Grid child element to the FlowSolver component. Set its *name* to “fsGrid” and associate the DistGrid you created with the new Grid object using the Properties View. (Cupid currently does not support the definition of Grid coordinates. For now, the user is required to modify the generated code to add coordinates to a Grid object.)

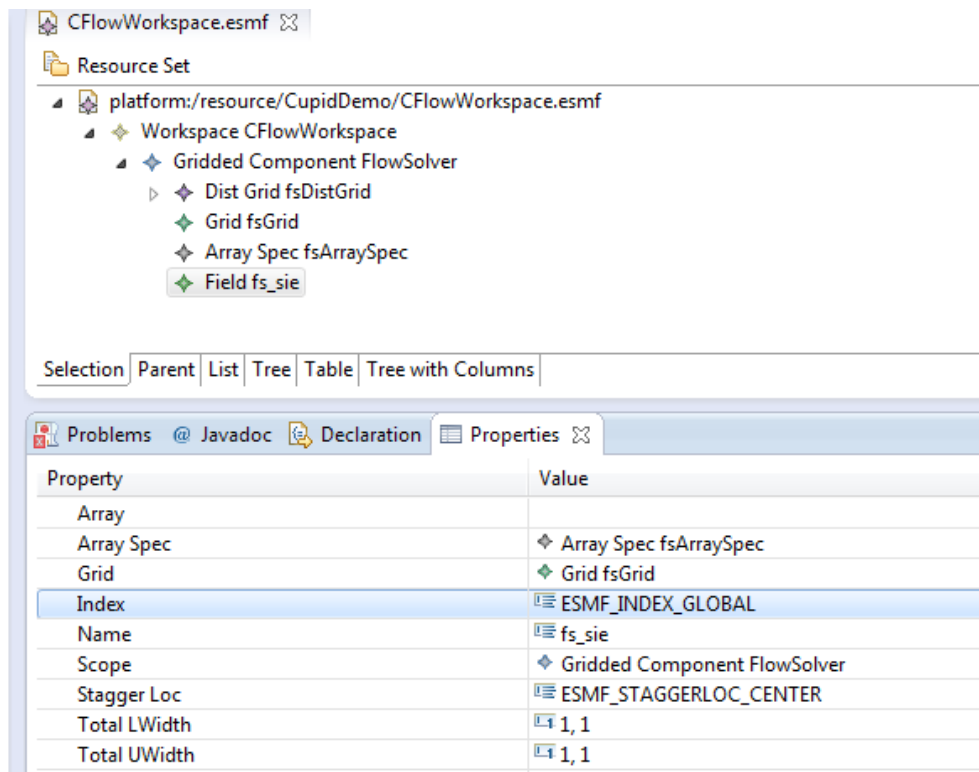


- We will now define the fields for the FlowSolver component. First, add an ArraySpec to the Gridded Component with the following properties:
 - Name: fsArraySpec
 - Rank: 2
 - Typekind: ESMF_TYPEKIND_R4

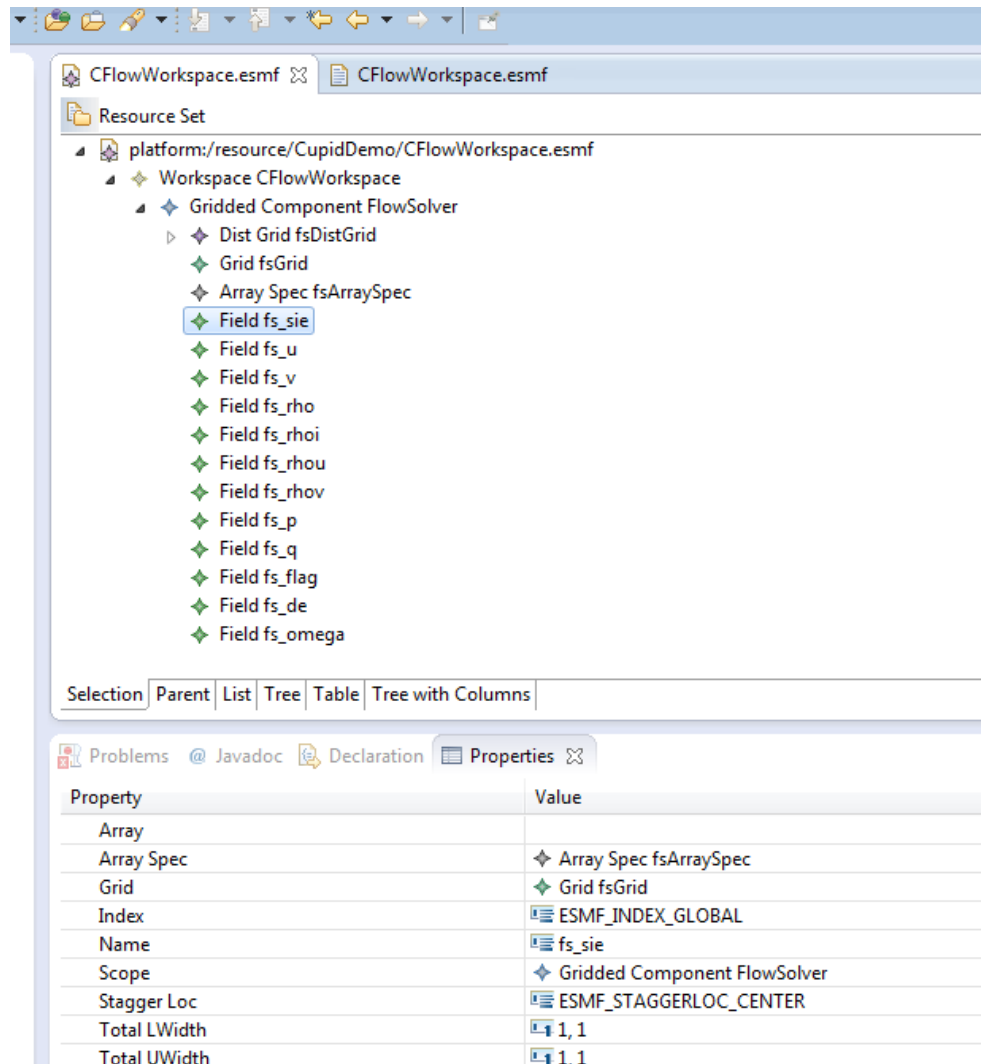


- Now that the DistGrid, Grid, and ArraySpec objects have been defined, add a Field object to the FlowSolver component with the following properties:
 - Array Spec: fsArraySpec
 - Grid: fsGrid
 - Index: ESMF_INDEX_GLOBAL
 - Name: fs_sie
 - Stagger Loc: ESMF_STAGGERLOC_CENTER
 - Total L Width: 1, 1
 - Total U Width: 1, 1

If you are familiar with ESMF, then you will recognize that these properties establish an ESMF Field called fs_sie with the given ArraySpec, on the given Grid, with indices defined globally, centered staggering, and a halo of size 1 in all directions.



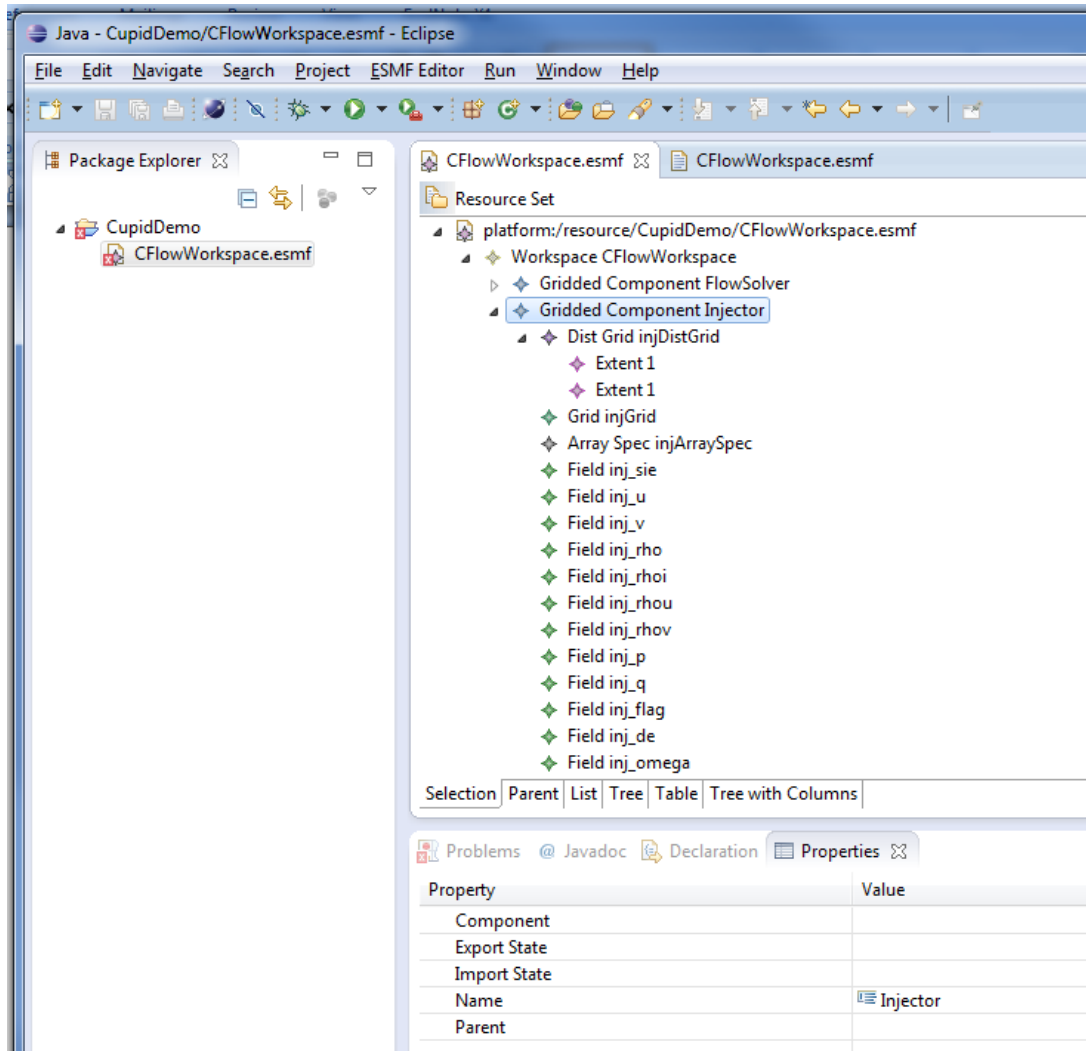
- The Flow Solver component in Coupled Flow demo has a total of 12 fields. All fields have the same properties (i.e., they share a grid and have the same stagger location, etc.). Create 11 more fields by copying and pasting the fs_sie field (in the right click menu) and changing the *name* property of each new field. (Alternatively, the underlying XML can be edited behind the scenes by opening the CFlowDemo.esmf file in a text editor.) You should end up with the following fields:
 - fs_sie
 - fs_u
 - fs_v
 - fs_rho
 - fs_rhoi
 - fs_rhou
 - fs_rhov
 - fs_p
 - fs_q
 - fs_flag
 - fs_de
 - fs_omega



- We have finished creating the FlowSolver Gridded Component. We will now create the Injector Gridded Component. The Injector component is identical to the Flow Solver except for the following differences:
 - the component name is different,
 - all “fs” prefixes should be “inj” instead (e.g., injGrid, injDistGrid, inj_sie, etc.),
 - the Dist Grid regular decomposition should be 1 x 4 instead of 2 x 2.

Therefore, the easiest way to create the Injector component is to make a copy of the FlowSolver element and make the changes listed above.

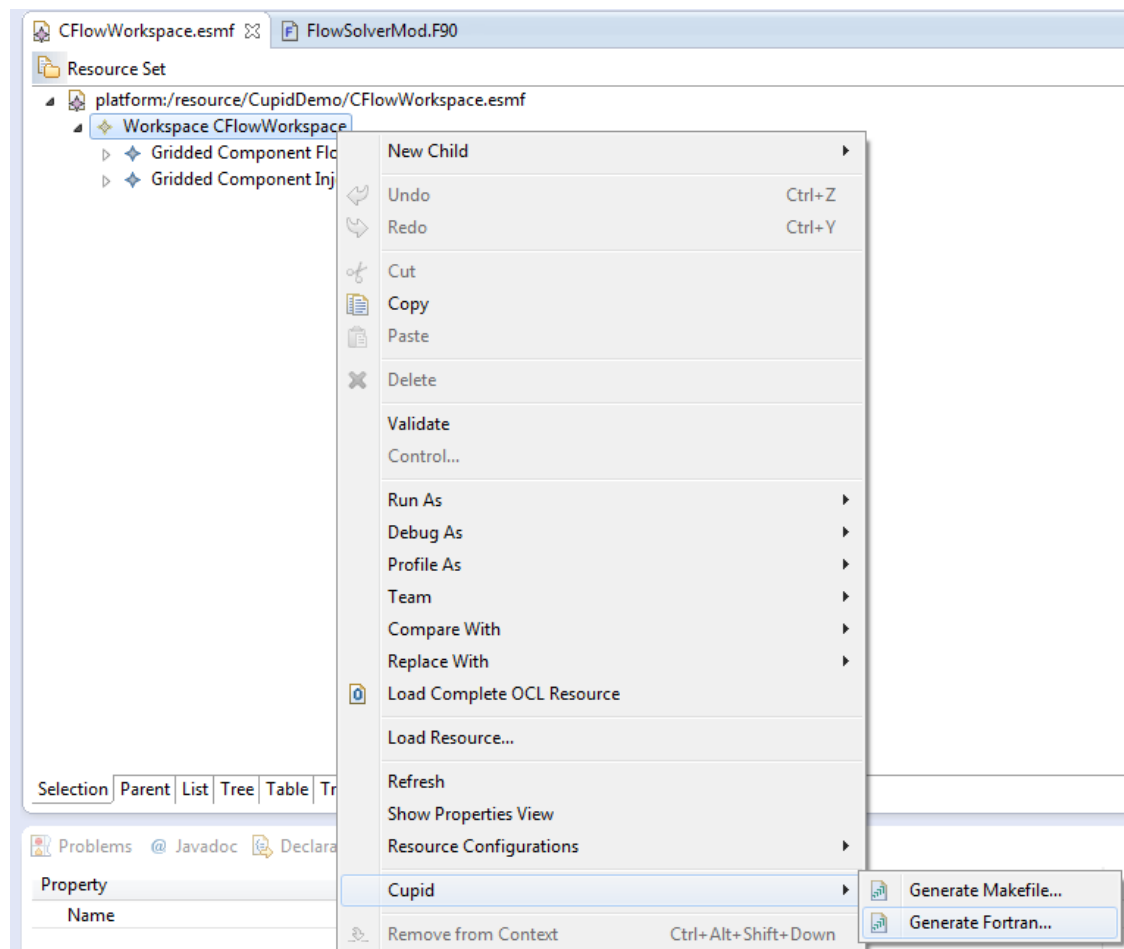
You should now have two Gridded Components in the CFlowWorkspace: FlowSolver and Injector.



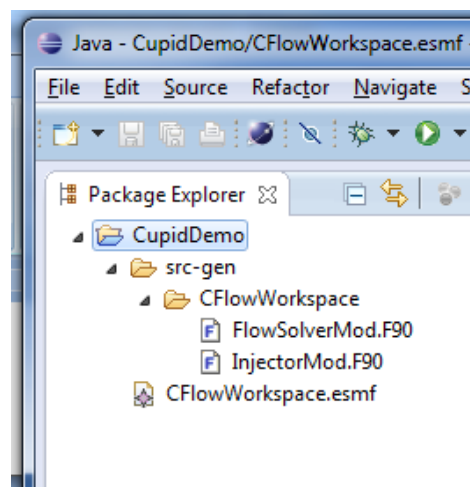
4. Generate skeleton code for gridded components

The model is now complete enough to generate skeleton code for the FlowSolver and Injector gridded components.

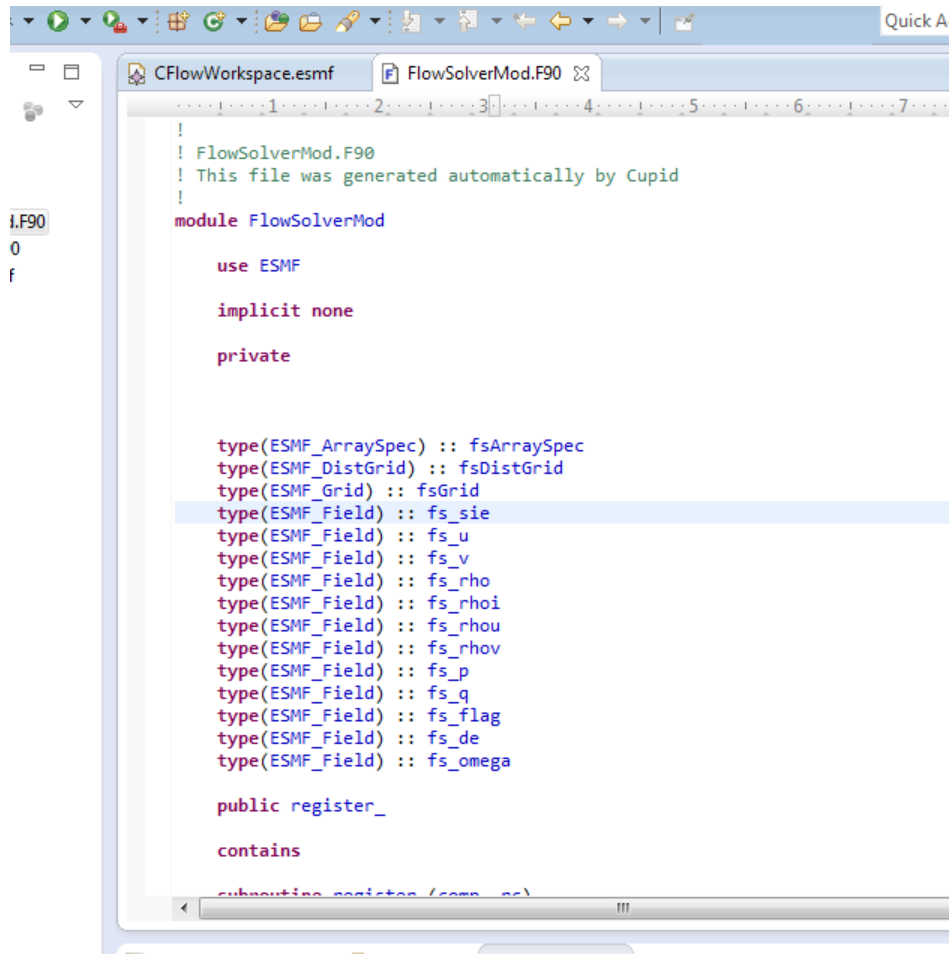
- First, we need to validate the model to make sure there are not any inconsistencies or missing elements. To validate, right click on the CFlowWorkspace element and choose Validate. You should see a dialog that says “Validation completed successfully.” If not, you will see which model constraints have been violated. Correct them before moving to the next step.
- Once the model is validated, you can generate Fortran code. Right click on the CFlowWorkspace element in the model editor and select Cupid→Generate Fortran.



- You should now see a new folder named “src-gen” in the Package Explorer window. It contains a folder CFlowWorkspace with two Fortran 90 files FlowSolverMod.F90 and InjectorMod.F90.



- View the source code of FlowSolverMod.F90 by double-clicking the file. If you have installed the Fortran Development Tools (Photran) the file will open with the Fortran language editor with syntax highlighting, etc.



```

! FlowSolverMod.F90
! This file was generated automatically by Cupid
!
module FlowSolverMod

  use ESMF

  implicit none

  private

  type(ESMF_ArraySpec) :: fsArraySpec
  type(ESMF_DistGrid) :: fsDistGrid
  type(ESMF_Grid) :: fsGrid
  type(ESMF_Field) :: fs_sie
  type(ESMF_Field) :: fs_u
  type(ESMF_Field) :: fs_v
  type(ESMF_Field) :: fs_rho
  type(ESMF_Field) :: fs_rhoi
  type(ESMF_Field) :: fs_rhou
  type(ESMF_Field) :: fs_rhov
  type(ESMF_Field) :: fs_p
  type(ESMF_Field) :: fs_q
  type(ESMF_Field) :: fs_flag
  type(ESMF_Field) :: fs_de
  type(ESMF_Field) :: fs_omega

  public register_

  contains

  subroutine register_(comp_nc)

```

- Explore the generated code in FlowSolverMod.F90. A few things to note:
 - The Flow Solver module has subroutines called register_, init_, run_, and finalize_. These are created implicitly.
 - The Array Spec, DistGrid, Grid, and Field objects are created in the init_ method and destroyed in the finalize_ method.
 - The run_ method is empty. Cupid does not support definition of the scientific code, only the coupling infrastructure. This method needs to be filled in manually by the user.
 - Some debugging print statements are generated automatically.
 - ESMF return codes are currently ignored.

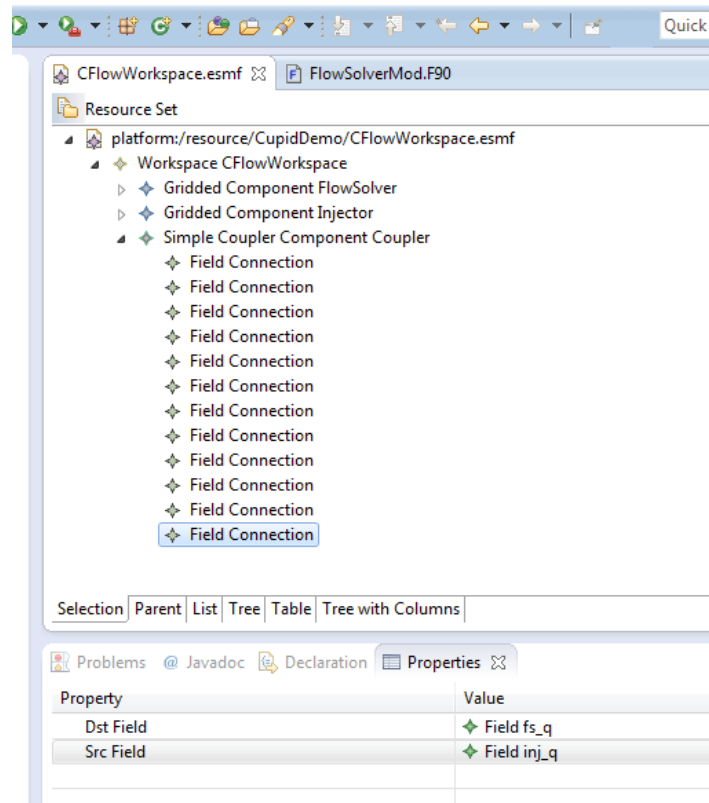
5. Add an ESMF coupler component, parent gridded component, and driver to the workspace

Now we will add an ESMF coupler component and driver so that we have the architecture of a complete ESMF application. Recall that the decompositions of the DistGrid's differ: FlowSolver has a parallel data decomposition of 2 x 2 while the Injector has a parallel data decomposition of 1 x 4. Therefore an ESMF redistribution operator is required to transfer data between the two components. Cupid supports a simple coupler component for basic transformations like redistribution and regridding.

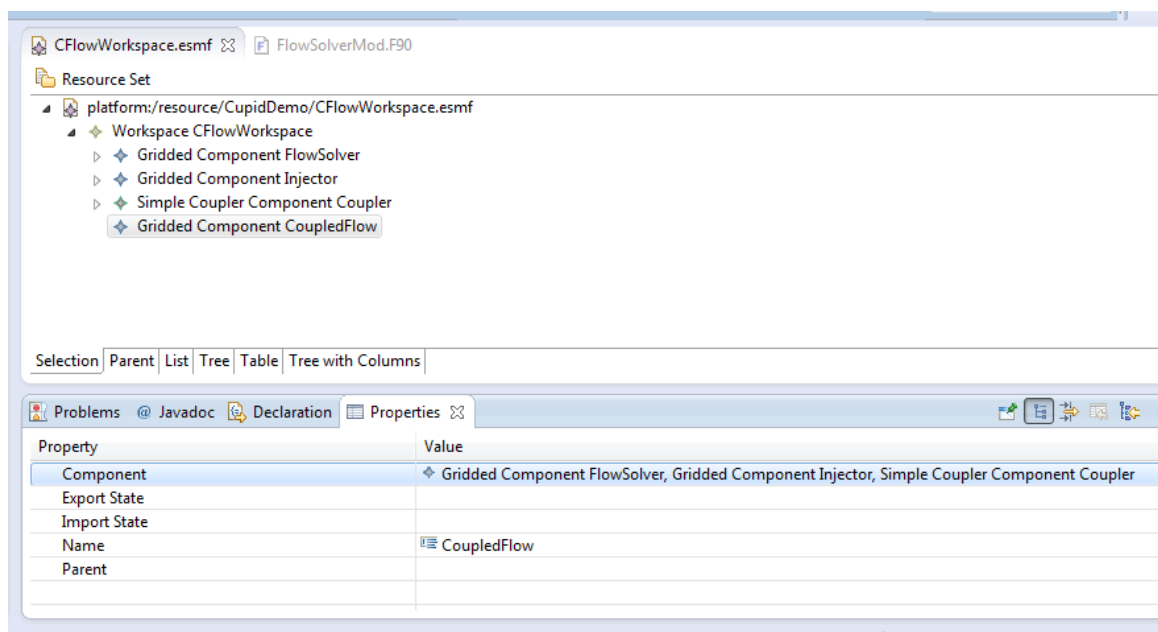
- In the ESMF model editor, add a new Simple Coupler Component to the workspace. Set the following properties:
 - Name: Coupler
 - Src Component: FlowSolver
 - Dst Component: Injector

Simple Coupler Components support two-way communication between Gridded Components, so the source and destination components will alternate during model execution.

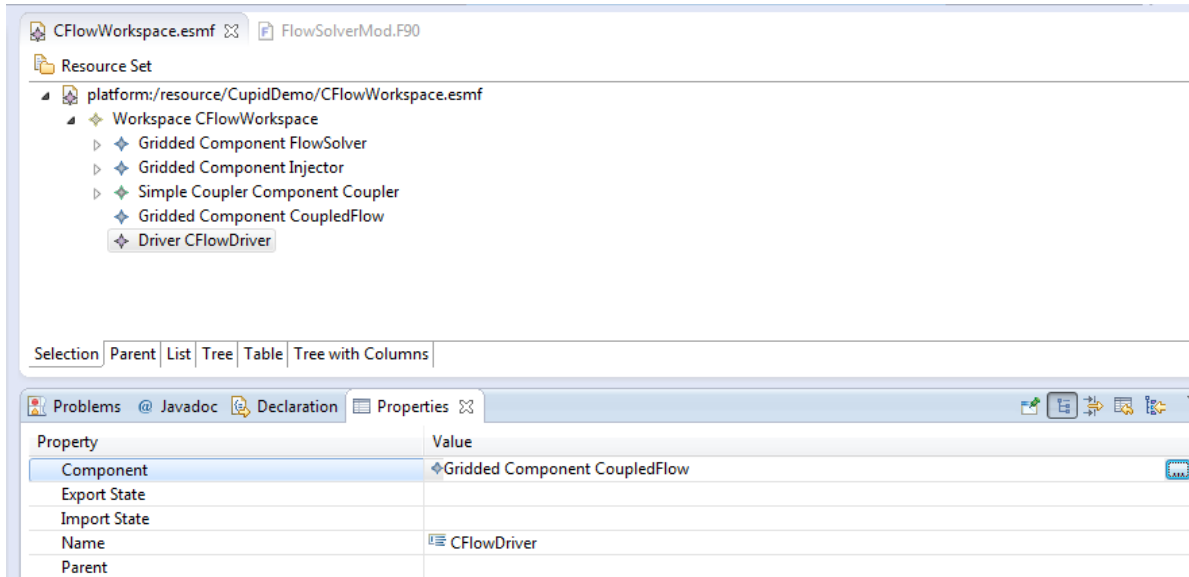
- Simple Coupler Components are based on Field Connections which establish a relationship between two ESMF Fields in different Gridded Components. Add a Field Connection child element to the Coupler. Set its *Src Field* property to *fs_sie* and its *Dst Field* property to *inj_sie*.
- Create a Field Connection element for each pair of fields between the FlowSolver and the Injector components. For now we'll assume that all fields need to be communicated between both models. When you are done you should have a total of 12 field connections.



- Add another Gridded Component to the workspace with the name CoupledFlow. This Gridded Component will be a parent of the FlowSolver, Injector, and Coupler components. To indicate that these three are children, set the *component* property of CoupledFlow Gridded Component to include the FlowSolver, Injector, and Coupler components.



- You should now have three Gridded Components (CoupledFlow, FlowSolver, and Injector) and one Simple Coupler Component (Coupler) in the workspace. Add a Driver to the workspace by right clicking on the Workspace and choosing New Child→Driver. Set the *name* of the Driver to CFlowDriver. Set the child *component* of the driver to the CoupledFlow Gridded Component.



- Congratulations! You have now specified the complete architecture that matches that of the Coupled Flow demo. Now we can generate code for the application and test it.

6. Generate skeleton code and a Makefile for the Coupled Flow demo

- Right click on the CFlowDemo Workspace in the model editor and choose Validate. Correct any validation errors.
- Right click on the CFlowDemo Workspace and choose Cupid→Generate Fortran. You should see five generated Fortran 90 files in the src-gen/CFlowWorkspace folder.
- Examine the code in CouplerMod.F90. A few things to notice:
 - There are six subroutines: init1_, init2_, run1_, run2_, finalize1_, finalize2_. The phase one subroutines (init1_, run1_, finalize1_) deal with the source→destination communication direction and the phase two subroutines deal with the destination→source direction.
 - A redistribution operator is calculated in the init subroutines. The choice of redistribution was determined automatically by Cupid because the two DistGrids have the same number of cells (100 x 100). (Try changing one DistGrid to 200 x 200 and re-generating the code. A regrid operator will be selected instead.)
 - The run routines execute the redistribution operator for each pair of connected fields. Note that Cupid currently assumes that all fields can share a Route Handle. This will not be the case in general.

- Examine the code in CoupledFlowMod.F90.
 - The init_ subroutine implicitly instantiates each child Gridded Component followed by each child Coupler Component. Order of execution dependencies are currently not taken into account.
 - Each Gridded Component receives its own import and export ESMF State objects. These are also implicitly instantiated in the init_ subroutine.
 - All child components are initialized in the CoupledFlow init_ subroutine. Both phases of the coupler initialization are called with different import and export states.
 - The run_ method executes the child Gridded and Coupler components' run methods. Order of execution dependencies between Gridded and Coupler Components are currently not taken into account.
- Examine the code in CFlowDriver.F90.
 - This code is a Fortran 90 program instead of a module.
 - The only child component (CoupledFlow) is instantiated, initialized, run, finalized, and destroyed.
 - Empty clock and state objects are passed to the component.

To test the code, let's first generate a Makefile that will compile the source files into an executable program.

- Right click on the CFlowDemo workspace and choose Cupid→Generate Makefile.

7. Execute the generated Coupled Flow application

Cupid is currently not integrated with an execution environment. To test the code, copy the five .F90 source files and the Makefile to a machine with an installation of ESMF 5.2 or later.

- Set the ESMFMKFILE environment variable to point to the Makefile for your installation of ESMF.

```
$ export ESMFMKFILE="/home/rocky/esmf/lib/libO/Linux.intel.64.openmpi.default/esmf.mk"
```

- Build the executable

```
$ make
```

- Execute using 4 processors

```
$ mpirun -np 4 ./CFlowDriver
```