Cupid

The Cupid Integrated Development Environment for Earth System Models

Feature Overview and Tutorial

Contents

Overview of Features

Cupid is a development environment for geoscience modelers who have basic programming experience and experience with geoscience model development workflows, but are new to developing with geoscience modeling frameworks. In general, a software framework provides a set of abstractions, *framework-provided concepts*, which the developer is required to instantiate and configure in code. Creating a framework-based application is called *framework completion* because the developer is filling in application behaviors not provided by the framework, or specializing existing behaviors. Cupid adds a layer of intelligence to the Eclipse Integrated Development Environment in order to facilitate model development using geoscience software frameworks such as ESMF and NUOPC.

Reverse Engineering and Compliance Verification

Most climate and weather model codebases are staggeringly large and obtaining an overview of the model, its subcomponents, and their interconnections is a cumbersome, time-consuming task. Often, this can only be accomplished by manually reading through the top-level source files to establish a mental picture of the overall structure of the model and its data flows.

Cupid's reverse engineering feature parses a model's source code and produces a high level representation of the framework concepts that are present. This representation, called the NUOPC tree viewer, is shown to the user alongside the source code in an outline form. This provides an alternative, more abstract perspective for viewing a model's source code. Some of the main concepts provided by NUOPC are Drivers, Models, Mediators, and Connectors. If a codebase contains any of these components, the reverse engineering function will automatically find them and present them in outline form. This provides an architectural overview of an entire coupled system without requiring the developer to read through hundreds of lines of source code.

NUOPC ensures interoperability of modeling components by specifying a set of technical rules that model implementers should follow. In addition to presenting a high level view of framework concepts in source code, Cupid's compliance checking feature provides feedback to the user when potential compliance issues are discovered in a reverse engineered model. For example, when developing a NUOPC Model component, certain subroutines must be implemented and registered with the framework. If a required subroutine or its registration is missing, Cupid can identify the problem and annotate the outline view with icons indicating that some required code is missing. Moreover, this feedback is provided immediately to the user during model development thereby reducing the number of runtime failures and improving efficiency of the development process.

Forward Engineering

Even if a software framework is well designed, writing framework completion code is notoriously difficult, even for seasoned developers. Often, completing a single logical task such as adding a new run initialization phase or run phase requires making several code additions at multiple places spread throughout the application source code. If one or more of the required additions are inadvertently left out, the application may not behave as expected.

In the software engineering research community, many ideas have been proposed for how to help developers write framework-based applications correctly and efficiently. For ESMF and NUOPC, guidance is provided in the form of comprehensive API documentation, system tests, and small archetypical codebases that show how to structure NUOPC applications based on the components in the modeled system (e.g., standalone atmosphere, coupled atmosphere-ocean, three-component system, etc.).

Cupid's forward engineering feature complements these static resources by generating on-the-fly NUOPC-compliant source code fragments directly in existing source files. The user initiates a code fragment generation by adding elements to the model in the NUOPC tree viewer. The source code is then synchronized with the tree viewer, generating the required code fragments. The generated code fragments can then be customized by the developer for their particular case. The following use case illustrates use of forward engineering feature:

> A developer has finished writing the initialization phases for a NUOPC Model component called ATM and now needs to add the capability to advance the model one time step. The developer right clicks on the ATM element in the NUOPC tree viewer and selects "Add Model Advance." Two things happen immediately: the tree viewer is updated with a new sub-element underneath ATM called "Model Advance" which in turn contains sub-elements "Registered in Set Services" and "Implementation." Then, source code fragments are generated inside the Fortran file for ATM including a call inside the ATM SetServices to register the Model Advance subroutine and a stub for the new Model Advance subroutine.

The use case shows some advantages of this approach compared to an approach in which code is copied-and-pasted from archetypical example code. First, the new Model Advance element added to the tree viewer included multiple sub-elements indicating that source code changes are required in at least two places: a new subroutine and a call to register this subroutine with the framework. This provides guidance to the developer to ensure that all framework requirements are met. The approach, therefore, is less error-prone than brute force copy and paste. Also, the generated code fragments are customized based on the state of the existing source code. For example, the developer may be using specialized variable names. Since these variables have already been discovered during the reverse engineering phase, the generated code can reference these variables instead of requiring the developer to modify variables in copy-pasted code.

Cloud Integration

Cupid simplifies the process of configuring a computational environment capable of compiling and executing high-performance geoscience models such as NASA's ModelE and archetypical NUOPC applications.

IDEs package a lot of development tools into a single application to help manage and simplify the software development workflow. Although IDEs aim to increase developer productivity, they can still introduce a steep learning curve. Some challenges with using IDEs for geoscience model development include:

- Understanding the basic steps involved in moving from source code to a running model
- Making sense of the many development tools and features available in the IDE
- Setting up a high-performance computational environment capable of configuring, compiling and executing model code
- Configuring the IDE to connect to remote computational environments

Cupid’s cloud integration feature allows a developer to select a training scenario and, within a few minutes, configure, compile, execute, and view the output of both skeleton models and realistic models. This feature relies on a set of pre-configured machine images that can be instantiated on Amazon EC2 cloud infrastructure. The machine images contain all of the necessary software dependencies for the selected scenario. Furthermore, Cupid automatically configures the IDE to connect to and synchronize source code with cloud-based virtual machines.

More information about Cupid can be found at: https://earthsystemcog.org/projects/cupid/

Tutorial

This tutorial will take you through the process of installing the Cupid Plugin for the Eclipse Integrated Development Environment (IDE) and using its features.

Install Eclipse and the Cupid Plugin

1. **Download and install Eclipse for Parallel Application Developers, version 4.3.1 SR1 (Kepler).**

   The main download page is: http://www.eclipse.org/downloads/.

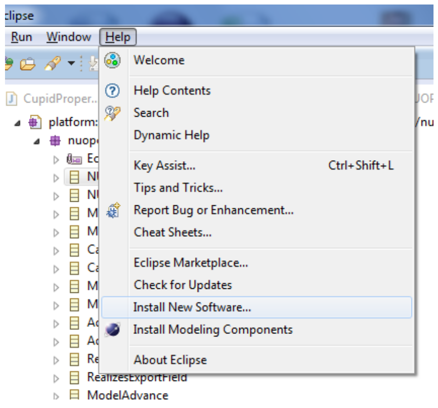
   There is a list of available Eclipse packages. Be sure to choose "Eclipse for Parallel Application Developers" as it will come pre-bundled with the necessary plugins for working with remote systems.

2. **Unpack the downloaded file into a local directory and run Eclipse by double clicking on the Eclipse executable.**

   The first time you start Eclipse, you will be prompted to select a location for your workspace. Choose an empty folder.

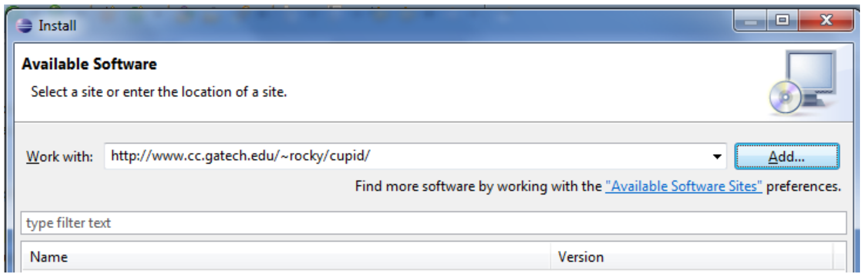
3. **Install the Cupid Plugin from the Cupid Update Site.**

   a. Click Help→Install New Software



   b. Put the Cupid Update Site URL into "Work with..." You will be prompted to give the update site a name of your choosing.

   The URL is: `http://www.cc.gatech.edu/~rocky/cupid/`


Install

Available Software

Select a site or enter the location of a site.

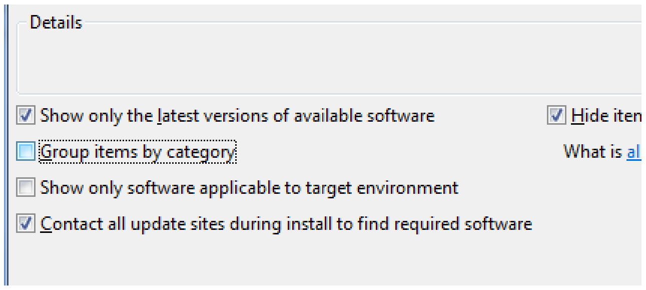
Work with: http://www.cc.gatech.edu/~rocky/cupid/

Find more software by working with the "Available Software Sites" preferences.

type filter text

Name	Version

c. Uncheck the “Group items by category” option.


Details

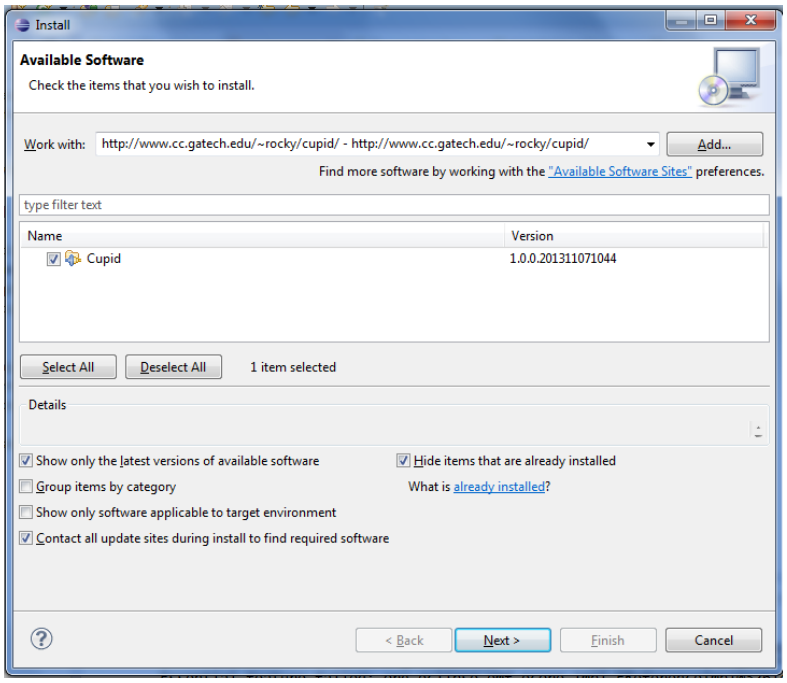
☑ Show only the latest versions of available software ☑ Hide iten

☐ Group items by category What is al

☐ Show only software applicable to target environment

☑ Contact all update sites during install to find required software

d. Select “Cupid” from the list and click Next.


Install

Available Software

Check the items that you wish to install.

Work with: http://www.cc.gatech.edu/~rocky/cupid/ - http://www.cc.gatech.edu/~rocky/cupid/

Find more software by working with the "Available Software Sites" preferences.

type filter text

Name	Version
☑ Cupid	1.0.0.201311071044

1 item selected

Details

☑ Show only the latest versions of available software ☑ Hide items that are already installed

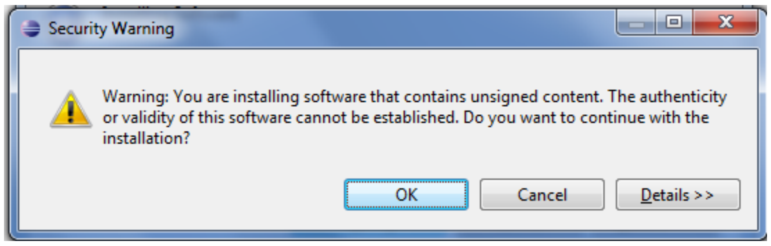
☐ Group items by category What is already installed?

☐ Show only software applicable to target environment

☑ Contact all update sites during install to find required software

e. You will need to click Next a couple more times and accept the license agreements. Then click Finish. The Cupid plugin and its dependencies will be downloaded and installed.

During the process, you may receive a message that the software contains unsigned content. Click OK.


Security Warning

Warning: You are installing software that contains unsigned content. The authenticity or validity of this software cannot be established. Do you want to continue with the installation?

f. After installation, you will be prompted to restart Eclipse. Click Yes.

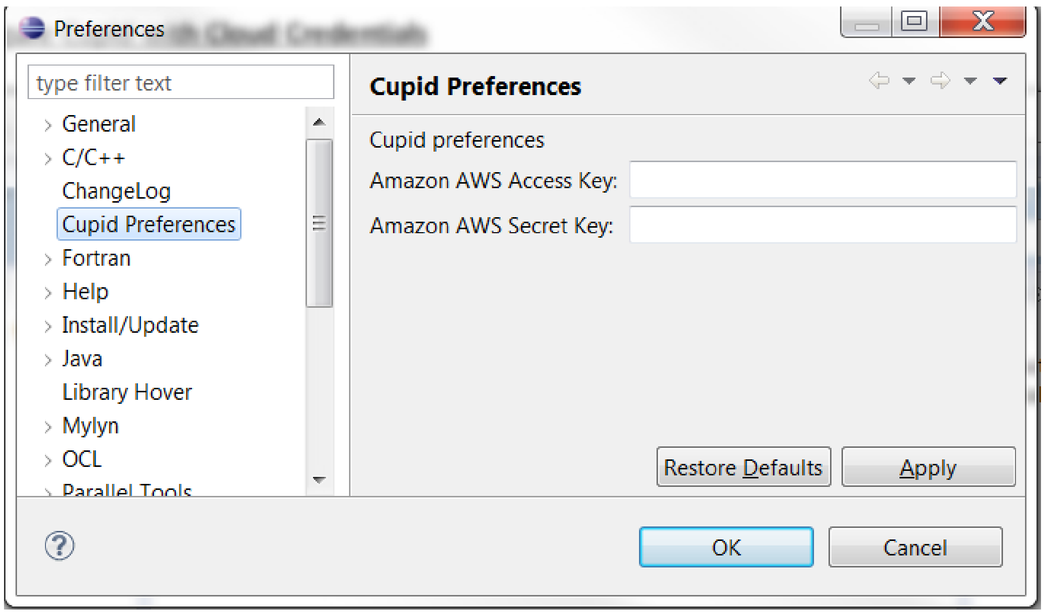
Configure Cupid with Cloud Credentials

Cupid creates cloud-based virtual machine instances to serve as the computational environment for the training scenarios. Currently, the only supported cloud provider is Amazon EC2. The first time you run Eclipse, you will need to set the Amazon EC2 credentials. You may use your own credentials or request access to the NESII cloud.

(Note: Cupid automatically launches Amazon machine instances when you create a new project. Alarms are set up to automatically kill instances after 50 minutes of idle CPU utilization. *However, it is your responsibility to ensure that any unused instances are terminated to avoid unnecessary cloud computing charges to your account*.)

1. In the Eclipse menu, select Window→Preferences. Then select Cupid Preferences in the list on the left.

2. Enter your Amazon Web Services (AWS) access key and secret key.



Cupid preferences

Amazon AWS Access Key:

Amazon AWS Secret Key:

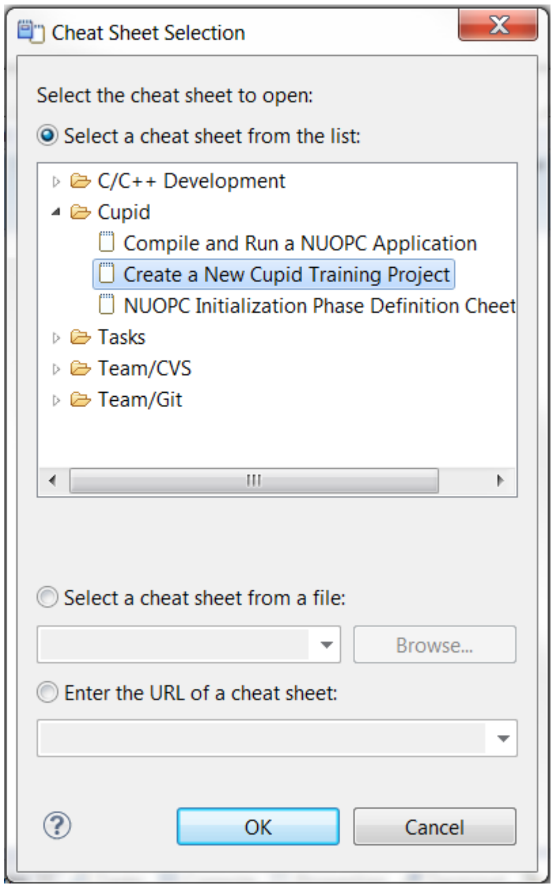
3. Click OK.

Using Cheat Sheets

An Eclipse Cheat Sheet is a step-by-step guide for performing some task in the IDE. Several Cheat Sheets have been provided that will help you get started using Cupid.

1. Select Help→Cheat Sheets from the Eclipse menu.

2. In the list at the top, choose the Cheat Sheet called "Create a New Cupid Training Project" in the Cupid folder.



Cheat Sheet Selection

Select the cheat sheet to open:

[✓ Select a cheat sheet from the list:]

- C/C++ Development
- Cupid
  - Compile and Run a NUOPC Application
  - [✓ Create a New Cupid Training Project]
  - NUOPC Initialization Phase Definition Cheet
- Tasks
- Team/CVS
- Team/Git

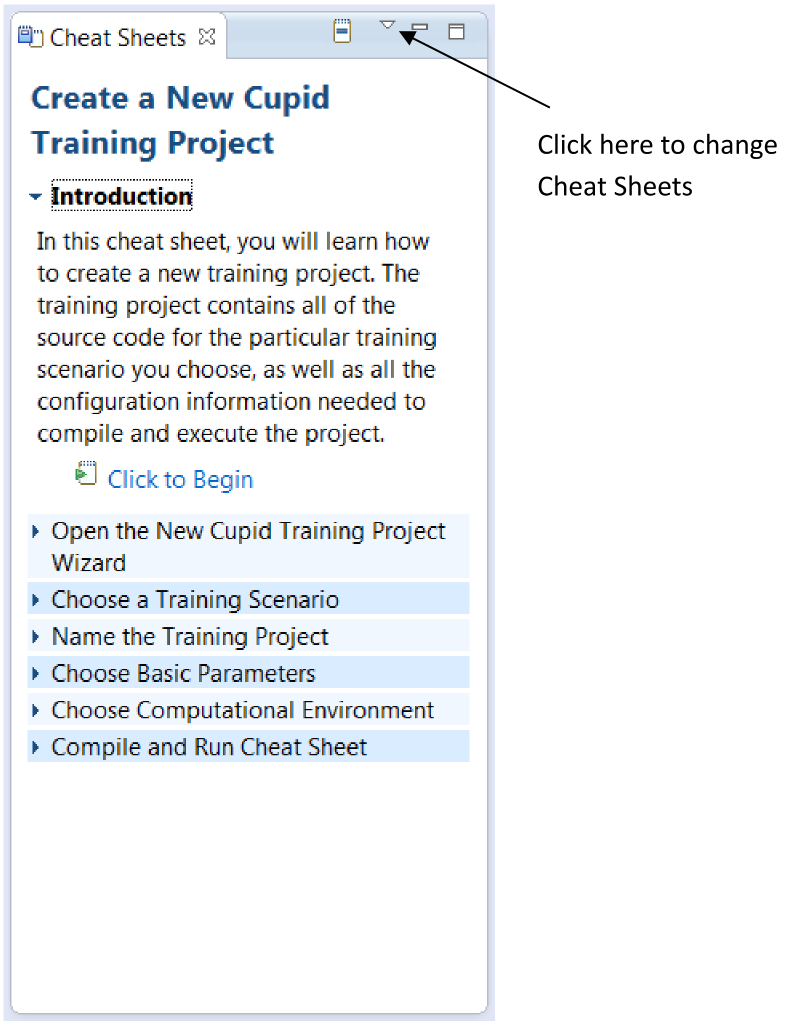
Select a cheat sheet from a file:

Browse...

Enter the URL of a cheat sheet:

OK Cancel

3. Click OK. You should see a new view in Eclipse showing the Cheat Sheet. At any time, to change Cheat Sheets, click on the small downward-facing triangle in the Cheat Sheet toolbar.



Create a New Cupid Training Project

Introduction

In this cheat sheet, you will learn how to create a new training project. The training project contains all of the source code for the particular training scenario you choose, as well as all the configuration information needed to compile and execute the project.

Click to Begin

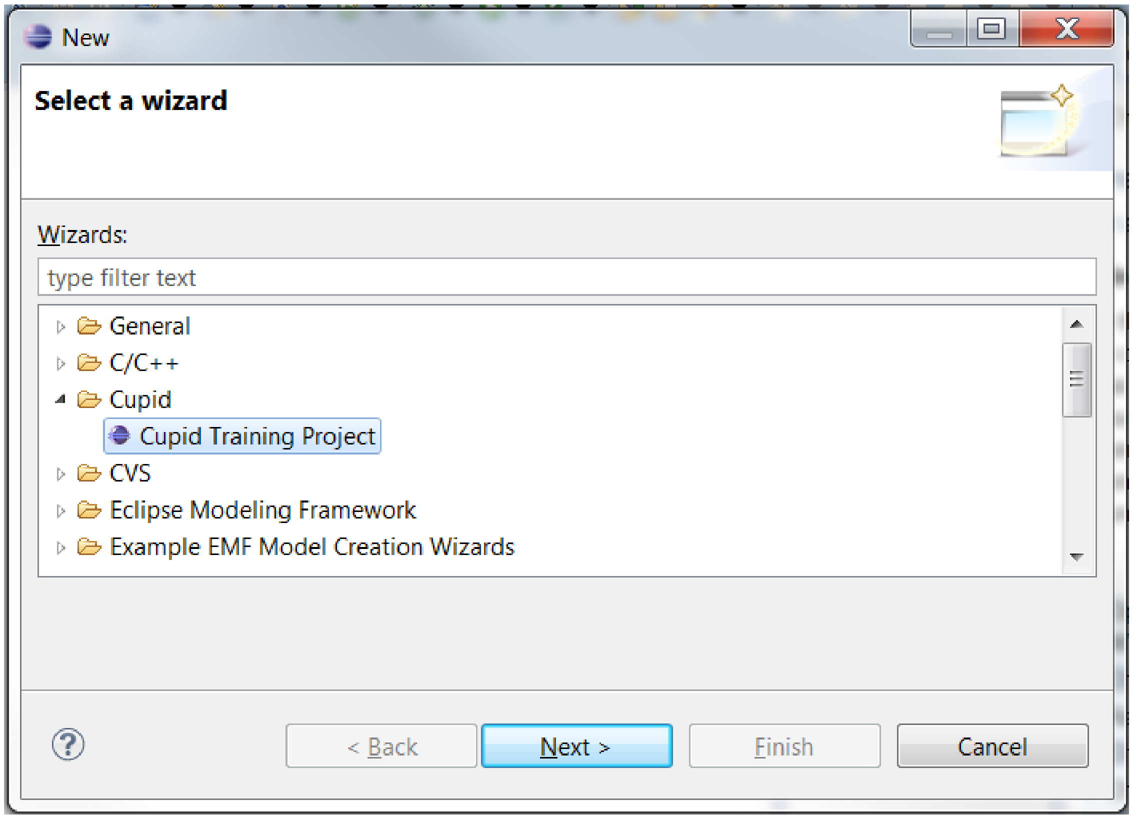
- Open the New Cupid Training Project Wizard
- Choose a Training Scenario
- Name the Training Project
- Choose Basic Parameters
- Choose Computational Environment
- Compile and Run Cheat Sheet

4. Follow the steps in the Cheat Sheet to create a new Cupid Training Project. The steps for creating a new project are also included in the next section of this tutorial.

Create a New Cupid Training Project

This section of the tutorial describes how to create a new Cupid Training Project. It is the same steps as outlined in the Cheat Sheet mentioned in the previous section.

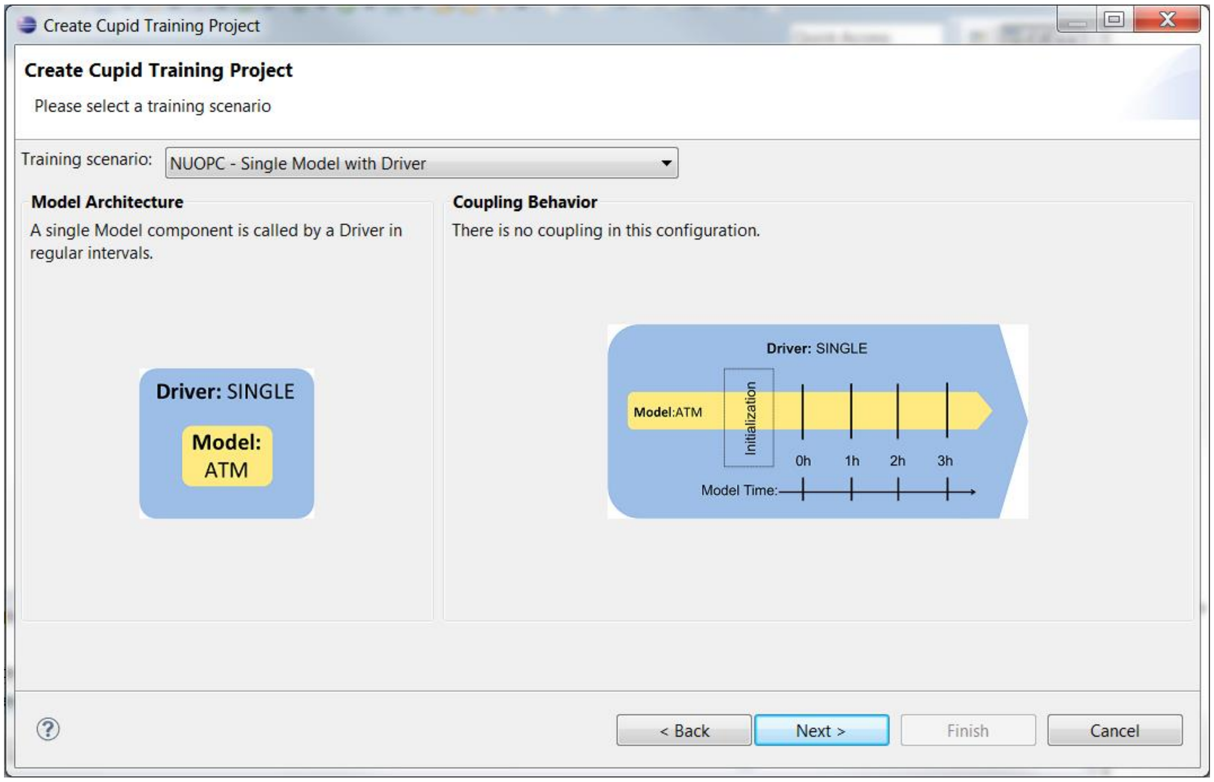
1. Click File→New→Other… and choose Cupid Training Project in the Cupid folder. Click Next.



2. On the first page of the wizard, choose a training scenario. Several options are available, but currently on the first option is supported:

    - NUOPC – Single Model with Driver
    - NUOPC – Coupled Atmosphere-Ocean Driver ***(coming soon)***
    - NUOPC – Coupled Atmosphere-Ocean with Mediator and Driver ***(coming soon)***
    - ModelE – Basic Configuration (EM20 rundeck) ***(coming soon)***

    Changing the training scenario in the list will update the screen to show the model architecture and coupling behavior of the selected scenario. Choose the scenario "NUOPC – Single Model with Driver" and click Next.


Create Cupid Training Project

Create Cupid Training Project

Please select a training scenario

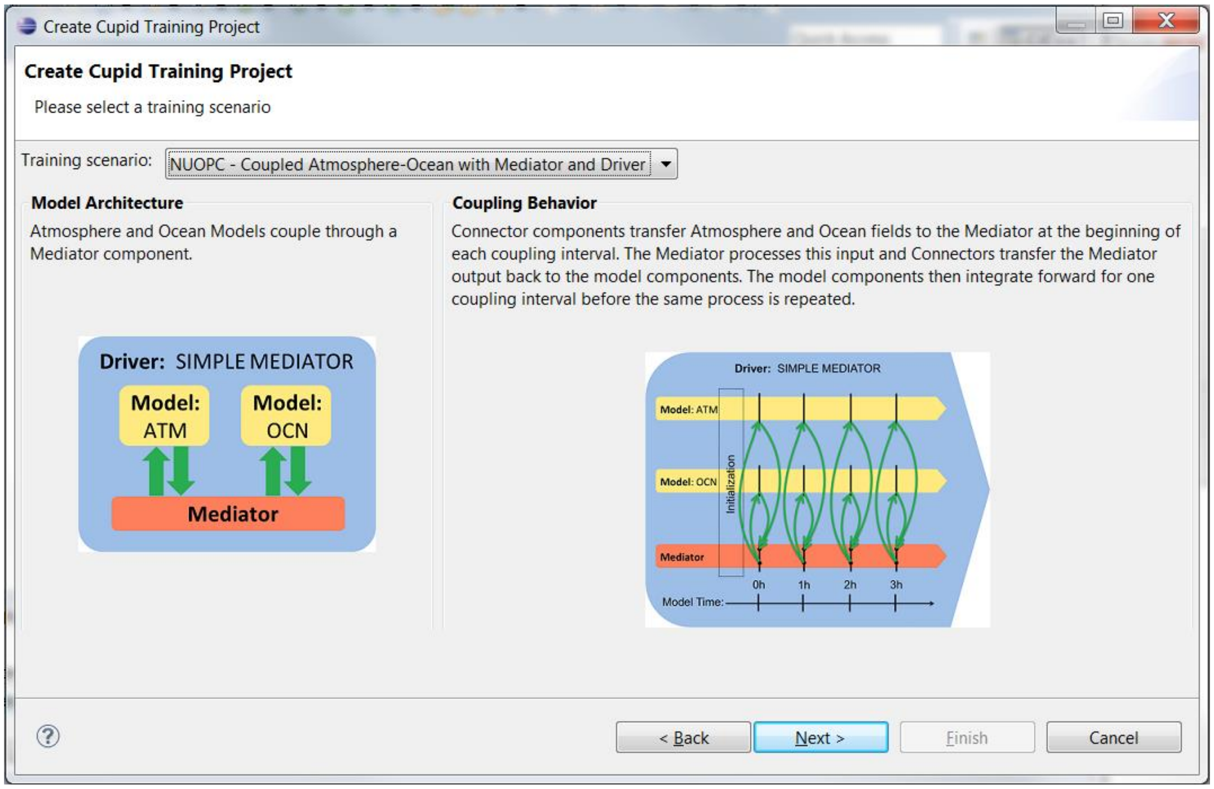
Training scenario: NUOPC - Single Model with Driver

Model Architecture

A single Model component is called by a Driver in regular intervals.

Coupling Behavior

There is no coupling in this configuration.


Create Cupid Training Project

Create Cupid Training Project

Please select a training scenario

Training scenario: NUOPC - Coupled Atmosphere-Ocean with Mediator and Driver

Model Architecture

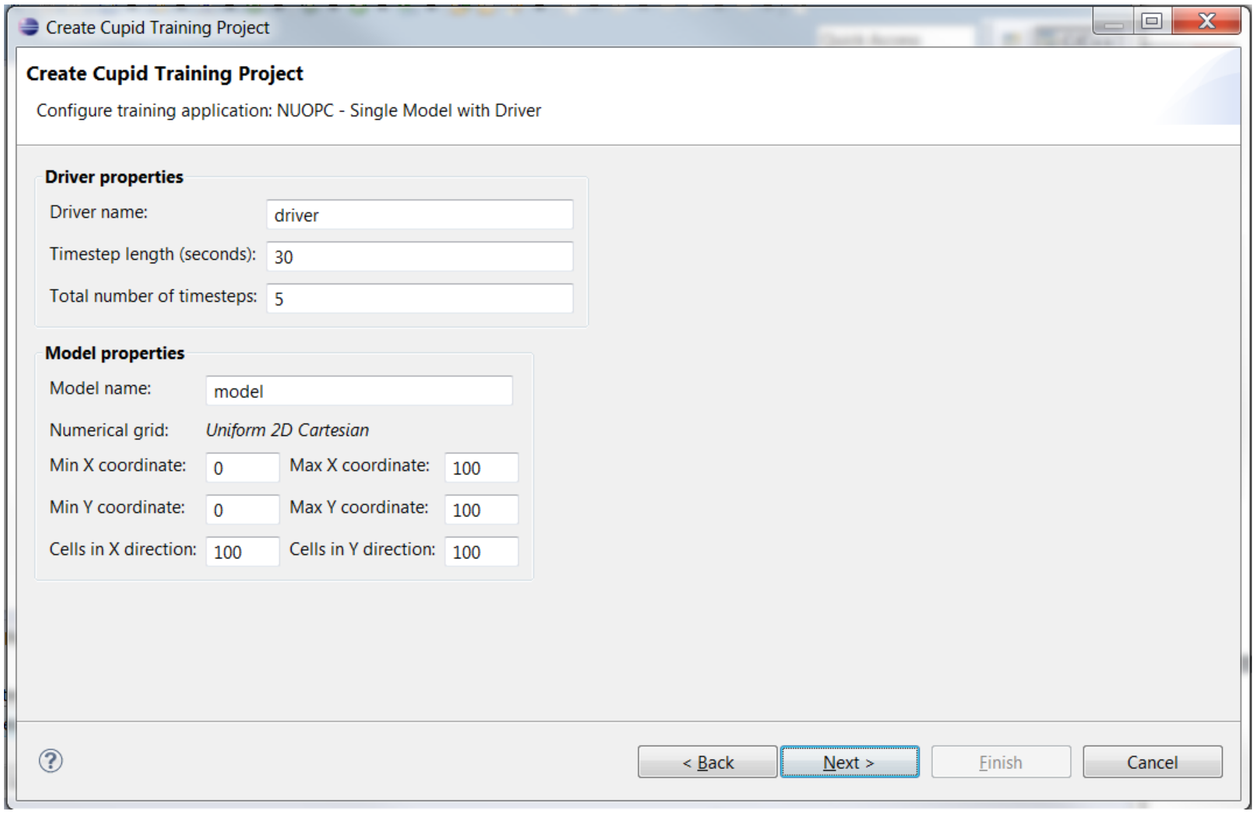
Atmosphere and Ocean Models couple through a Mediator component.

Coupling Behavior

Connector components transfer Atmosphere and Ocean fields to the Mediator at the beginning of each coupling interval. The Mediator processes this input and Connectors transfer the Mediator output back to the model components. The model components then integrate forward for one coupling interval before the same process is repeated.

3. On this page, choose a name for your project or accept the default name. Note that project names must be unique, so you must choose a name that does not already exist in your workspace. Click Next.

4. On this page you can optionally set a few parameters supported by the training scenario. It is okay to leave all the default values. Click Next.



Create Cupid Training Project

Configure training application: NUOPC - Single Model with Driver

Driver properties

Driver name:	driver
Timestep length (seconds):	30
Total number of timesteps:	5

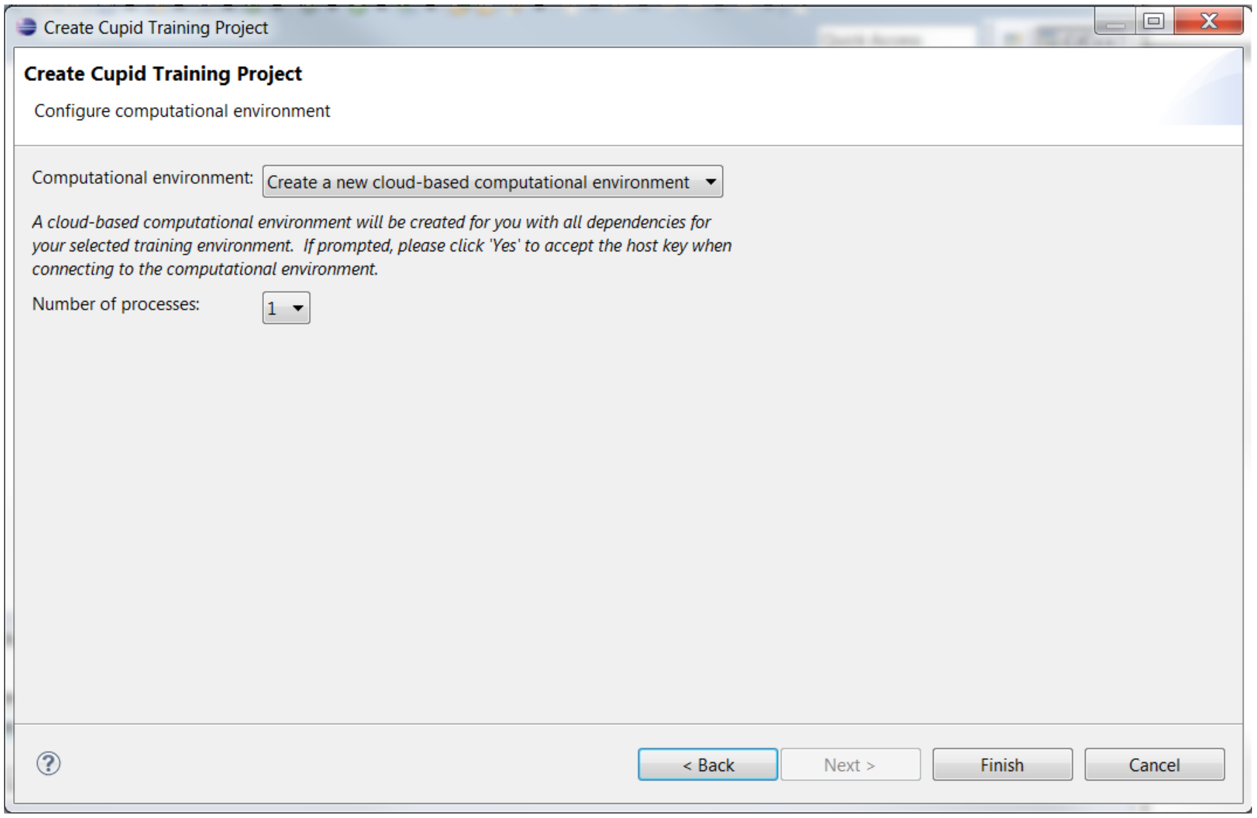
Model properties

Model name:	model		
Numerical grid:	*Uniform 2D Cartesian*		
Min X coordinate:	0	Max X coordinate:	100
Min Y coordinate:	0	Max Y coordinate:	100
Cells in X direction:	100	Cells in Y direction:	100

5. On this page you will choose a computational environment in which to compile and run the training scenario. The two options are:

   - Create a cloud-based computational environment
   - Use my local machine

   The first option will create a preconfigured computational environment for you with all dependent software (model source code, Fortran compiler, MPI, NetCDF, ESMF, etc.). This is the recommended option unless you know you have a supported environment already set up locally. Choose the first option and leave the number of processes at 1. Click Finish.



Create Cupid Training Project

Create Cupid Training Project

Configure computational environment

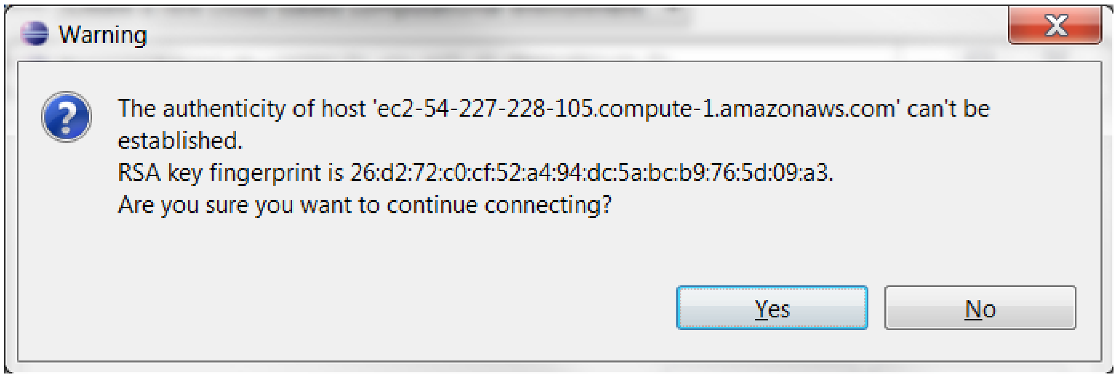
Computational environment: Create a new cloud-based computational environment

A cloud-based computational environment will be created for you with all dependencies for your selected training environment. If prompted, please click 'Yes' to accept the host key when connecting to the computational environment.

Number of processes: 1

< Back | Next > | Finish | Cancel

6. It will take up to several minutes for the new computational environment to start up. During this process you will be asked to accept the host SSH key. Click Yes.

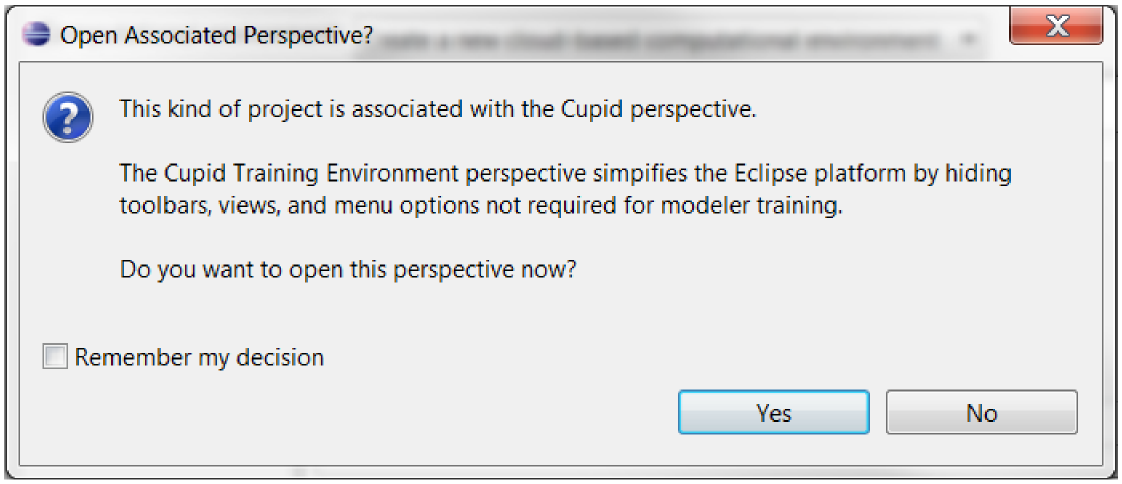


Warning

The authenticity of host 'ec2-54-227-228-105.compute-1.amazonaws.com' can't be established.
RSA key fingerprint is 26:d2:72:c0:cf:52:a4:94:dc:5a:bc:b9:76:5d:09:a3.
Are you sure you want to continue connecting?

Yes | No

You may also be asked if you would like to switch to the Cupid Perspective. An Eclipse perspective is a particular screen layout customized for specific tasks. The Cupid Perspective hides a number of Eclipse tools and commands that are not required for the training. Click Yes to switch to the Cupid Perspective.



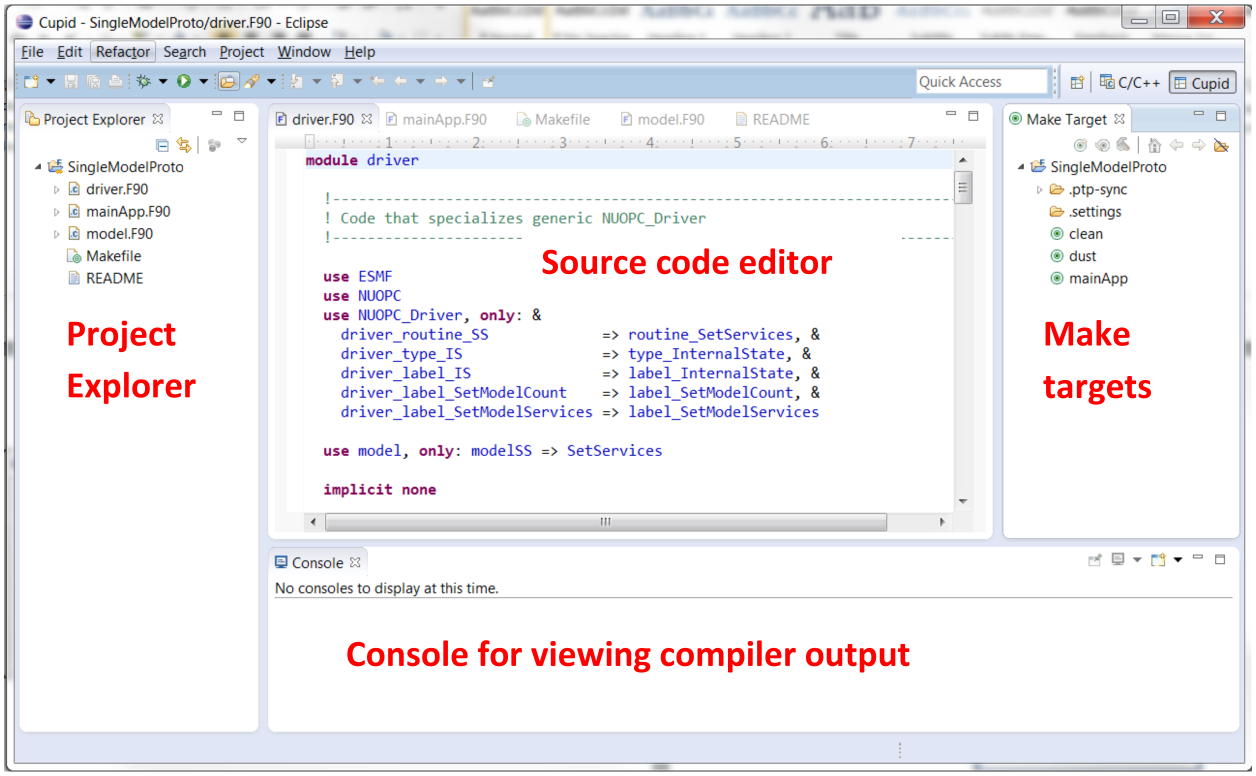
Open Associated Perspective?

This kind of project is associated with the Cupid perspective.

The Cupid Training Environment perspective simpifies the Eclipse platform by hiding toolbars, views, and menu options not required for modeler training.

Do you want to open this perspective now?

7. You should now see your project in the Project Explorer. The Cupid Perspective also exposes several other views including a Fortran source code editor, a Make Target view on the right for compiling your code and a Console at the bottom for viewing output from the compiler.



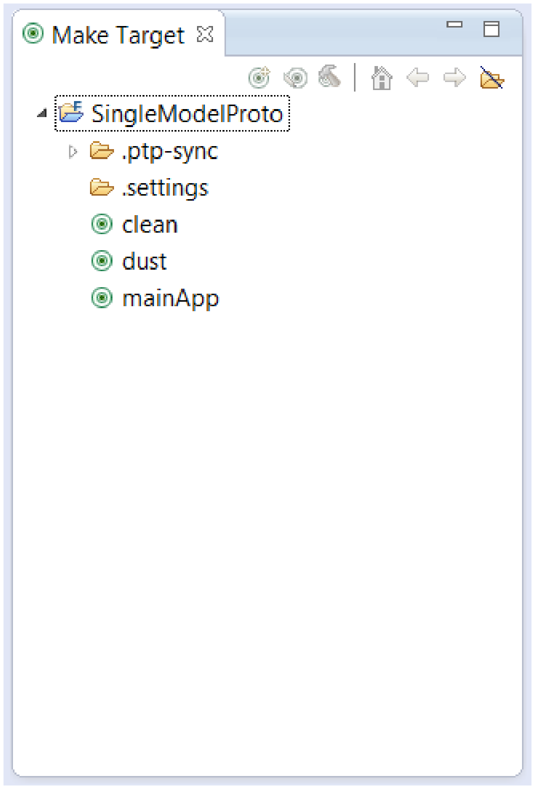
```
module driver

  !-----------------------------------------------------------------------------
  ! Code that specializes generic NUOPC_Driver
  !-----------------------------------------------------------------------------

  use ESMF
  use NUOPC
  use NUOPC_Driver, only: &
    driver_routine_SS             => routine_SetServices, &
    driver_type_IS                => type_InternalState, &
    driver_label_IS               => label_InternalState, &
    driver_label_SetModelCount    => label_SetModelCount, &
    driver_label_SetModelServices => label_SetModelServices

  use model, only: modelSS => SetServices

  implicit none
```

```
No consoles to display at this time.
```

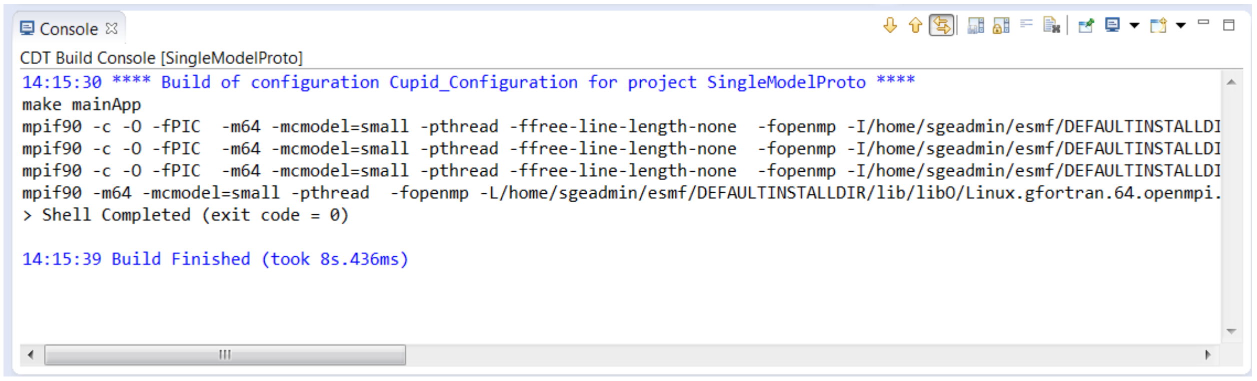
Compile and Run the NUOPC Single Model with Driver training scenario

In this section of the tutorial, you will learn how to compile and execute the source code provided in the NUOPC Single Model with Driver training scenario.   A Cheat Sheet is available for this task. (Click Help→Cheat Sheets and choose “Compile and Run a NUOPC Application” in the Cupid folder.)

1. Ensure that the Make Target view is showing (see below). If not, choose Window→Show View→Other… and select “Make Target” under the Make folder.

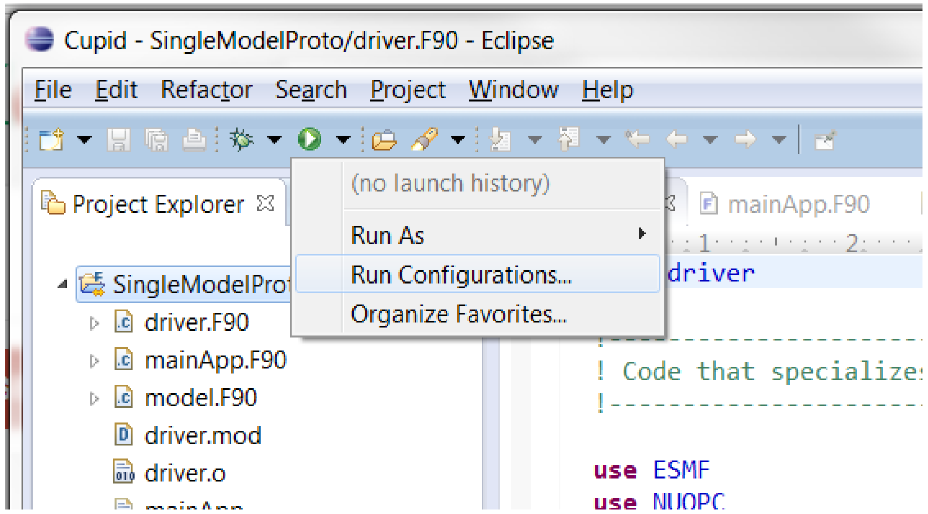


2. If you used the New Cupid Training Project wizard to set up the training scenario, then the correct make targets have already been set up. To compile the NUOPC application, double click the “mainApp” target. You should be able to see the compiler output in the Console.



```
14:15:30 **** Build of configuration Cupid_Configuration for project SingleModelProto ****
make mainApp
mpif90 -c -O -fPIC  -m64 -mcmodel=small -pthread -ffree-line-length-none  -fopenmp -I/home/sgeadmin/esmf/DEFAULTINSTALLDI
mpif90 -c -O -fPIC  -m64 -mcmodel=small -pthread -ffree-line-length-none  -fopenmp -I/home/sgeadmin/esmf/DEFAULTINSTALLDI
mpif90 -c -O -fPIC  -m64 -mcmodel=small -pthread -ffree-line-length-none  -fopenmp -I/home/sgeadmin/esmf/DEFAULTINSTALLDI
mpif90 -m64 -mcmodel=small -pthread  -fopenmp -L/home/sgeadmin/esmf/DEFAULTINSTALLDIR/lib/libO/Linux.gfortran.64.openmpi.
> Shell Completed (exit code = 0)

14:15:39 Build Finished (took 8s.436ms)
```

3. To run the compiled code, you must set up a run configuration. *(Note: This step will be automated in a future release of Cupid.)* Click the down arrow next to the Run As... button in the toolbar (green circle with a while arrow) and choose Run Configurations... from the popup menu.

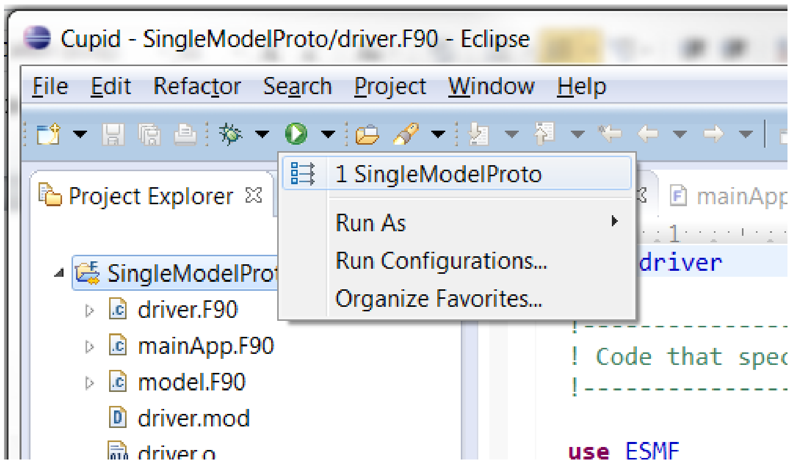


```
driver

!------------------------
! Code that specialize
!------------------------

use ESMF
use NUOPC
```

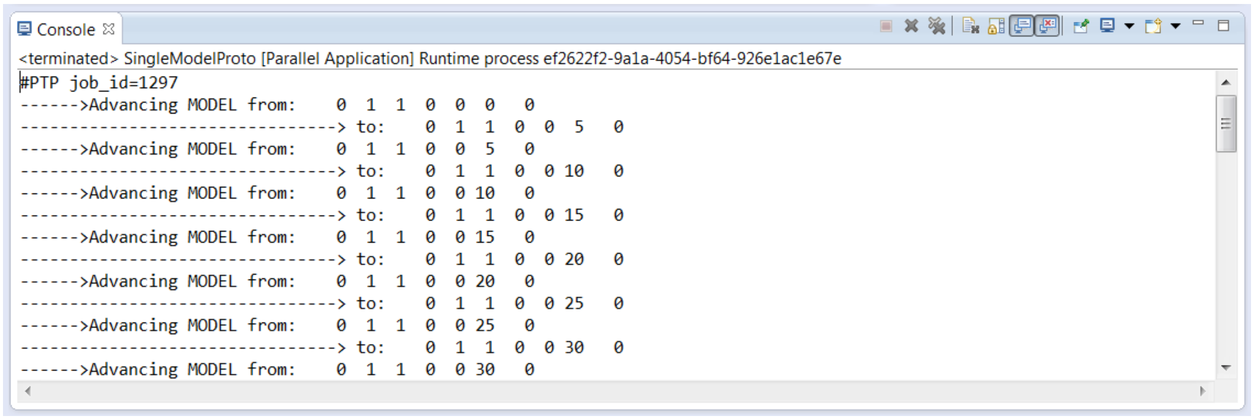
4. In the Run Configurations dialog, right click on Parallel Application and choose "New." Configure the run configuration as follows:

- Resources tab
  - Target System Configuration: **Open MPI-Generic-Interactive**
  - Connection type: **Remote**
  - Connection: **Cupid Environment (Amazon EC2 <ip address>)**
- Application tab
  - Project: should default to your project name
  - Application program: Click Browse and select "mainApp". The final path should be **/home/sgeadmin/SingleModelProto/mainApp**
- Click Apply then Run

5. Once the Run Configuration has been created, you do not need to set it up again unless you need to change some configuration settings. After running it the first time, the run configuration should be available in the Run Configurations dropdown list on the toolbar.



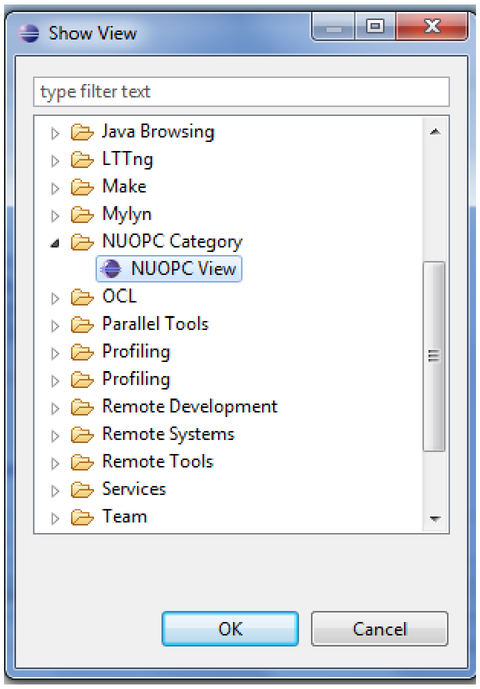
6. Output from the run will be shown in the Console view.



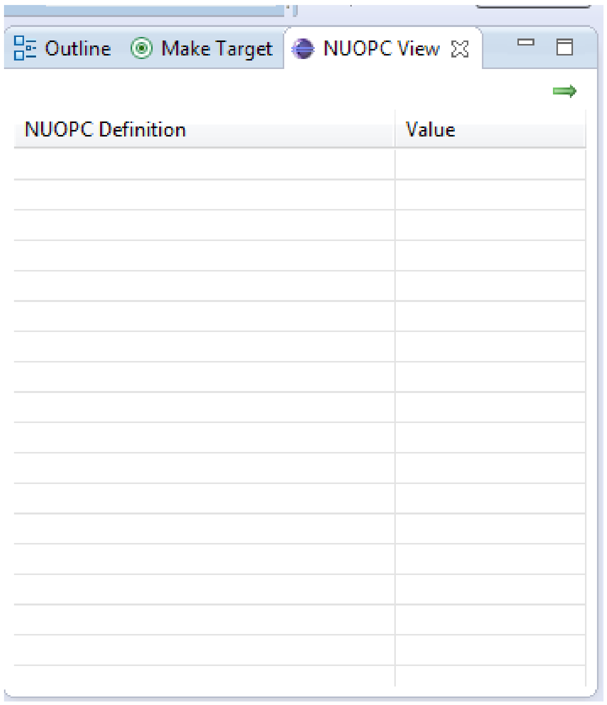
```
<terminated> SingleModelProto [Parallel Application] Runtime process ef2622f2-9a1a-4054-bf64-926e1ac1e67e
#PTP job_id=1297
------>Advancing MODEL from:    0  1  1  0  0  0   0
-------------------------------> to:     0  1  1  0  0  5   0
------>Advancing MODEL from:    0  1  1  0  0  5   0
-------------------------------> to:     0  1  1  0  0 10   0
------>Advancing MODEL from:    0  1  1  0  0 10   0
-------------------------------> to:     0  1  1  0  0 15   0
------>Advancing MODEL from:    0  1  1  0  0 15   0
-------------------------------> to:     0  1  1  0  0 20   0
------>Advancing MODEL from:    0  1  1  0  0 20   0
-------------------------------> to:     0  1  1  0  0 25   0
------>Advancing MODEL from:    0  1  1  0  0 25   0
-------------------------------> to:     0  1  1  0  0 30   0
------>Advancing MODEL from:    0  1  1  0  0 30   0
```

Show the NUOPC tree viewer.

Click the Window menu→Show View→Other and then select the NUOPC View in the NUOPC Category.



The NUOPC view will appear and will initially be empty. The view can be moved by dragging the "NUOPC View" tab to another part of the screen.

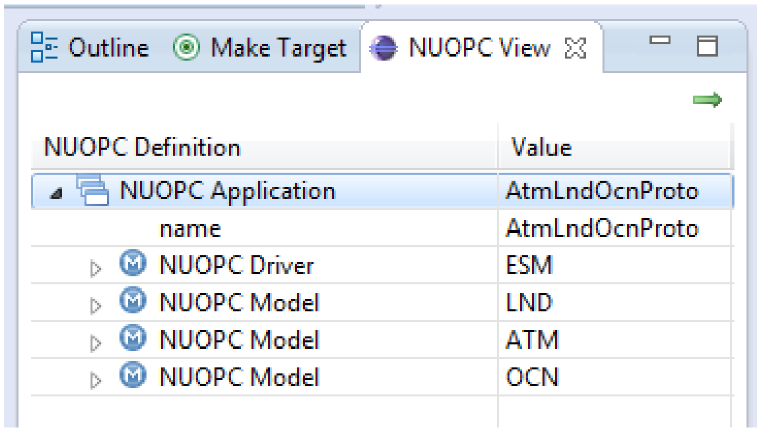


NUOPC Definition	Value

4. Reverse engineer the AtmOcnLndProto project.

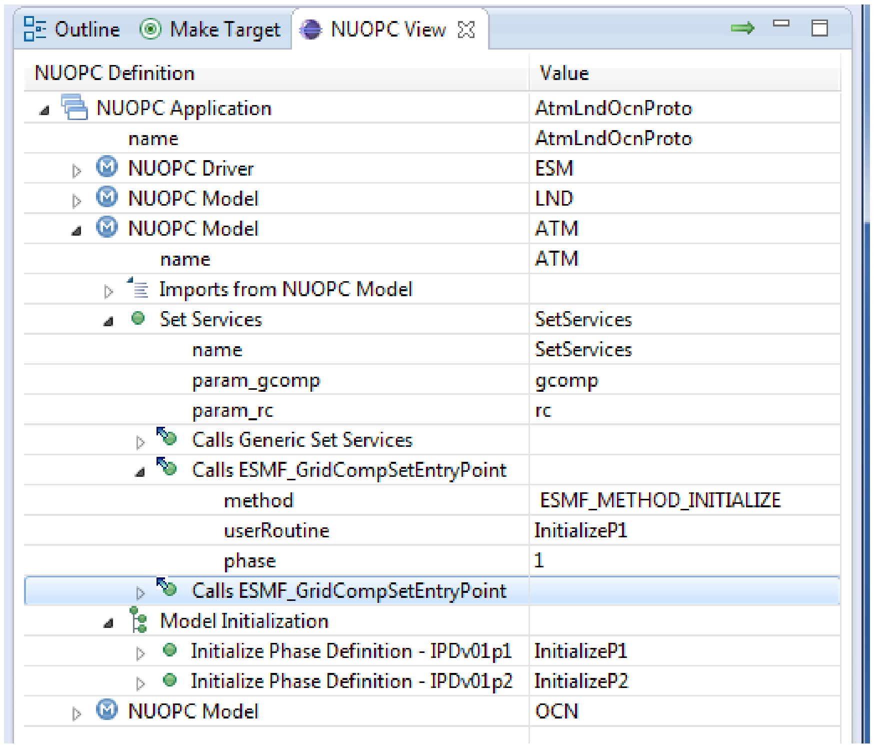
a. Open any of the .F90 files in the AtmOcnLndProto project by double clicking on the file name. The selected file tells Cupid which project to reverse engineer. (It is possible to have multiple NUOPC projects in the same workspace.)

b. Click the green arrow in the top-right corner of the NUOPC View or select Cupid→Reverse Engineer from the menu. The NUOPC View will update with a tree structure representing the reverse engineered model.



NUOPC Definition	Value
NUOPC Application	AtmLndOcnProto
name	AtmLndOcnProto
NUOPC Driver	ESM
NUOPC Model	LND
NUOPC Model	ATM
NUOPC Model	OCN

c. Expand the tree to see what NUOPC code patterns were discovered by Cupid.

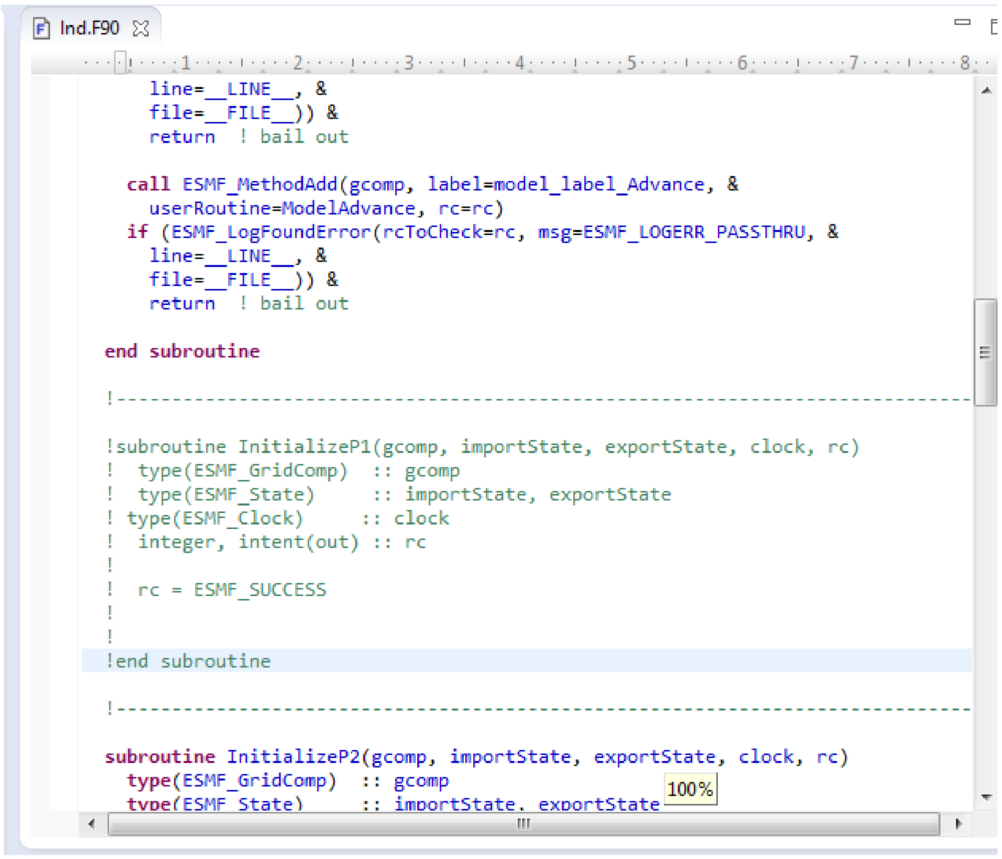


NUOPC Definition	Value
NUOPC Application	AtmLndOcnProto
name	AtmLndOcnProto
NUOPC Driver	ESM
NUOPC Model	LND
NUOPC Model	ATM
name	ATM
Imports from NUOPC Model	
Set Services	SetServices
name	SetServices
param_gcomp	gcomp
param_rc	rc
Calls Generic Set Services	
Calls ESMF_GridCompSetEntryPoint	
method	ESMF_METHOD_INITIALIZE
userRoutine	InitializeP1
phase	1
Calls ESMF_GridCompSetEntryPoint	
Model Initialization	
Initialize Phase Definition - IPDv01p1	InitializeP1
Initialize Phase Definition - IPDv01p2	InitializeP2
NUOPC Model	OCN

5. Show a NUOPC compliance issue using Cupid.

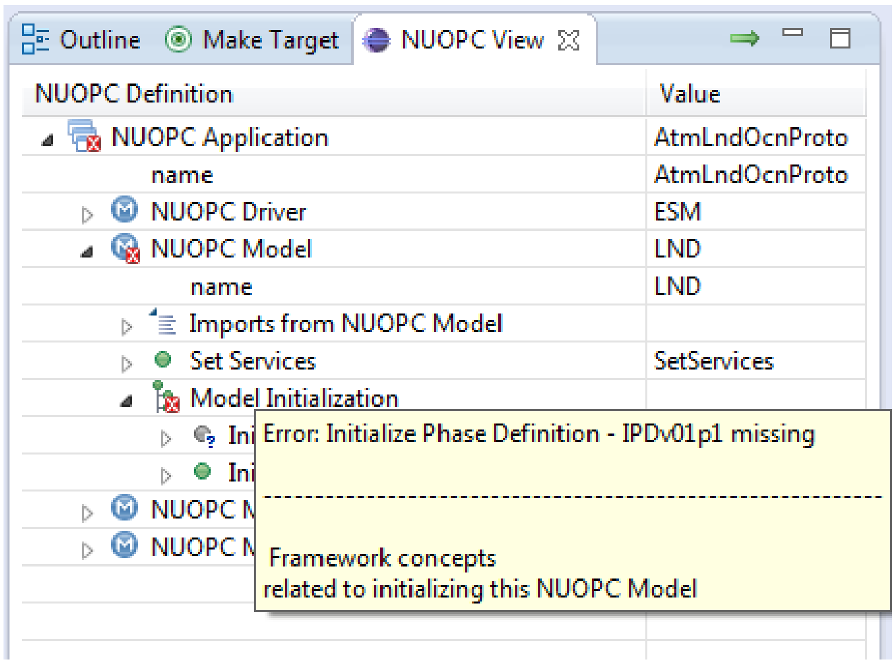
Cupid can automatically indicate NUOPC compliance problems that would hinder interoperability of a codebase with other NUOPC components. The AtmOcnLndProto code is already NUOPC compliant, but we will introduce a problem in order to show how Cupid reports validation errors.

a. Open the file **lnd.F90** and comment out the subroutine `InitializeP1`. We will assume that the developer forgot to implement this subroutine, or knows about it but simply hasn't written it yet.



```
      line=__LINE__, &
      file=__FILE__)) &
      return  ! bail out

    call ESMF_MethodAdd(gcomp, label=model_label_Advance, &
      userRoutine=ModelAdvance, rc=rc)
    if (ESMF_LogFoundError(rcToCheck=rc, msg=ESMF_LOGERR_PASSTHRU, &
      line=__LINE__, &
      file=__FILE__)) &
      return  ! bail out

  end subroutine

  !-----------------------------------------------------------------------------

  !subroutine InitializeP1(gcomp, importState, exportState, clock, rc)
  !  type(ESMF_GridComp)  :: gcomp
  !  type(ESMF_State)     :: importState, exportState
  ! type(ESMF_Clock)     :: clock
  !  integer, intent(out) :: rc
  !
  !  rc = ESMF_SUCCESS
  !
  !
  !end subroutine

  !-----------------------------------------------------------------------------

  subroutine InitializeP2(gcomp, importState, exportState, clock, rc)
    type(ESMF_GridComp)  :: gcomp
    type(ESMF_State)     :: importState, exportState
```

b. Save the modified **lnd.F90** file. Then, click the reverse engineer button again to re-analyze the code and update the NUOPC View. The view should now indicate an error with a small red X over the invalid node(s) in the tree. Hovering over the invalid node shows a description of the validation error.



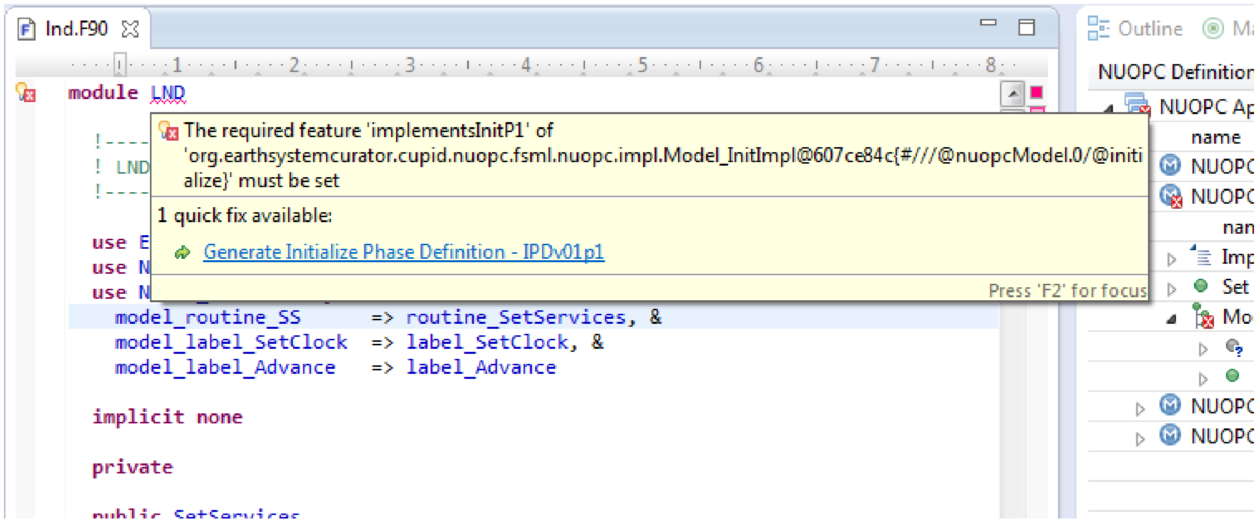
NUOPC Definition	Value
NUOPC Application	AtmLndOcnProto
name	AtmLndOcnProto
NUOPC Driver	ESM
NUOPC Model	LND
name	LND
Imports from NUOPC Model	
Set Services	SetServices
Model Initialization	
Ini	
Ini	
NUOPC M	
NUOPC M	

```
Error: Initialize Phase Definition - IPDv01p1 missing

--------------------------------------------------------

Framework concepts
related to initializing this NUOPC Model
```

6. Use Cupid's Quick Fix feature to generate missing code.

a. Scroll to the top of the **lnd.F90** file. The module name `LND` has been underlined in red indicating that the module has failed validation and is not NUOPC compliant.

b. Hover over the module name to show available quick fixes.

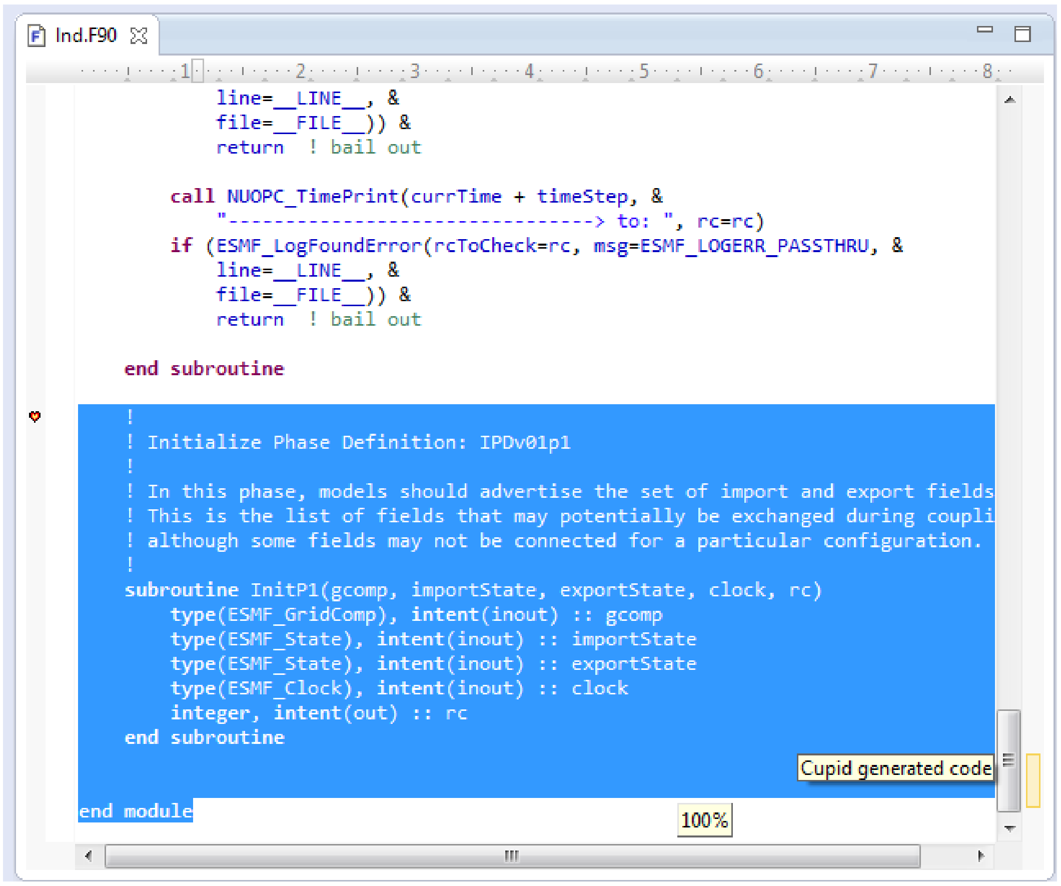


```
module LND

  !-----
  ! LND
  !-----

  use E
  use N
  use N
    model_routine_SS      => routine_SetServices, &
    model_label_SetClock  => label_SetClock, &
    model_label_Advance   => label_Advance

  implicit none

  private

  public SetServices
```

```
The required feature 'implementsInitP1' of
'org.earthsystemcurator.cupid.nuopc.fsml.nuopc.impl.Model_InitImpl@607ce84c{#///@nuopcModel.0/@initi
alize}' must be set
1 quick fix available:
  Generate Initialize Phase Definition - IPDv01p1
                                                              Press 'F2' for focus
```

c. Click on the one quick fix available: “Generate Initialize Phase Definition - IPDv01p1”

d. A new subroutine skeleton named InitP1 is generated automatically and inserted at the end of the module. The generated code is bookmarked and highlighted in yellow on the right vertical ruler to make it easy to distinquish the generated code from the existing code.



```
          line=__LINE__, &
          file=__FILE__)) &
          return  ! bail out

      call NUOPC_TimePrint(currTime + timeStep, &
          "--------------------------------> to: ", rc=rc)
      if (ESMF_LogFoundError(rcToCheck=rc, msg=ESMF_LOGERR_PASSTHRU, &
          line=__LINE__, &
          file=__FILE__)) &
          return  ! bail out

  end subroutine

  !
  ! Initialize Phase Definition: IPDv01p1
  !
  ! In this phase, models should advertise the set of import and export fields
  ! This is the list of fields that may potentially be exchanged during coupli
  ! although some fields may not be connected for a particular configuration.
  !
  subroutine InitP1(gcomp, importState, exportState, clock, rc)
      type(ESMF_GridComp), intent(inout) :: gcomp
      type(ESMF_State), intent(inout) :: importState
      type(ESMF_State), intent(inout) :: exportState
      type(ESMF_Clock), intent(inout) :: clock
      integer, intent(out) :: rc
  end subroutine

end module
```