ANGRYVIPER IDE User Guide

May 2019

Revision History

Revision	Description of Change	Date
	Initial, from earlier doc sources	02-2018
1.0	Revision for v1.4	09-2018
2.0	Updates for v1.5	05-2019
2.1	Converted to LaTeX, added build instructions	05-2019

Contents

Revision History i

List of Figures ii

List of Tables ii

References 1
- Assumptions . . . 1
- Reference Documents . . . 1

Introduction 1
- New Features in the 1.5 Release . . . 1

Overview 2
- AV Perspective Views . . . 2

IDE Overview and Features 4
- Overview . . . 4
- OpenCPI Projects View Features . . . 4

Operations View Features 5
- Build Status View Features . . . 8
- Eclipse Project Explorer View . . . 9
- Eclipse Console View . . . 10
- OpenCPI Asset Wizard . . . 11
- OpenCPI Asset XML Editors . . . 12
- The Execution Configuration . . . 16
- XML Editors-Modifying Existing XML Files . . . 16
- Warning about a Non-OpenCPI Project . . . 17
- The Perspective is Not in Sync with the Workspace or the File System . . . 17

OpenCPI Development Workflow Using the IDE 17
- Introduction . . . 17
- OpenCPI the Core and Assets Projects . . . 17
- Importing OpenCPI Core and Asset Projects . . . 17
- Build the Projects for the desired RCC and HDL Platforms . . . 18
- Creating New Projects and Libraries . . . 19
- More on OpenCPI Projects . . . 21
- Project Registration . . . 21
- Errors in Creating New Project . . . 22
- OpenCPI Libraries . . . 22
- Creating Components, Protocols and Workers . . . 23
- Component Example from the Getting Started Guide . . . 23
- Worker Example from the Getting Started Guide . . . 24
- Creating Applications and Assemblies . . . 25

Adding Components to an Application 26
Connecting Application Components 27
Assembly Editor 29
Creating Component Unit Tests 31

Appendices **33**

A Eclipse Basics **33**

B Eclipse Basic Concepts **35**

C Additional Plugins for the IDE **36**
TM Terminal 4.0 36

D Build The IDE **37**
Introduction 37
Instructions 37

List of Tables

1 References 1

List of Figures

1 ANGRYVIPER Perspective Views 2
2 IDE showing C/C++ Perspective 3
3 PerspectiveView 4
4 List of Features 5
5 Projects View 6
6 Build configuration setup in the Operations panel 7
7 Unit Test Panel 7
8 List panel with 2 Different Remotes 8
9 Build Status View 9
10 Successful Build Execution 9
11 Status Bar 10
12 Eclipse Project Explorer View 11
13 Eclipse Console View 11
14 Notice Console View 12
15 Execution Console 12
16 Asset Creation Wizard 13
17 Modify Attributes of HDL Worker 14
18 Add a Property to a Worker 14
19 HDL Property Element Form 15
20 Graphical Drag and Drop Editors 16
21 Procedures to use existing Projects 18
22 Importing OpenCPI Core and Assets 19
23 Build Entry 19
24 Project Built from Operations panel 20
25 Core Project Built from Ops panel 20
26 Create a New Project 21
27 Asset Wizard 22
28 Terminal Window View 22
29 Create a New OpenCPI Asset 23
30 Port Form for the Ramp In 24
31 Worker Example 24
32 Addition of In and Out Interfaces 25
33 Asset Wizard New Application 25
34 Application Editor 26
35 Properties View 26
36 Adding the Ramp Component 27
37 Third Method to Add a Component Instance 27
38 Select File Write as the New Companion 28
39 Design View of the Application 28
40 Connection Example 29
41 Pop Up Panel 30
42 Advanced Connection Views 30
43 Square HDL Worker 30
44 All Inputs are preset for the Ramp Spec 31
45 Unit Test Editor Panel 31
46 Display of Attribute Choice 32
47 Eclipse Launcher 33
48 Eclipse Welcome Screen 33
49 Eclipse Workspace 34

References

Some familiarity with the Eclipse Integrated Development Environment (IDE) or similar environment is assumed. If this is not the case, Appendix A gives basic information to get you started.

Assumptions

The document begins with the assumption that OpenCPI and ANGRYVIPER IDE are installed on a development machine and the Eclipse workspace has been previously set up.

Reference Documents

Table 1: References

Title	Link
OpenCPI Overview	`Overview.pdf`
Acronyms and Definitions	`Acronyms_and_Definitions.pdf`
Getting Started	`Getting_Started.pdf`
Installation Guide	`RPM_Installation_Guide.pdf`
OpenCPI Website	www.opencpi.org
OpenCPI Component Development Guide	`OpenCPI_Component_Development.pdf`
OpenCPI Application Development Guide	`OpenCPI_Application_Development.pdf`

Introduction

The ANGRYVIPER (AV) IDE consists of the Eclipse IDE for C/C++ developers and a custom plugin for OpenCPI development. OpenCPI users local to the project may install OpenCPI and the IDE from the yum repository (the IDE is a separate AV IDE RPM). Open source users must obtain the IDE plugin from GitHub and drop it into an existing Eclipse installation. Both methods are described in the RPM Installation Guide Section 6, addresses the RPM Installation and Appendix B addresses the plugin install. Note, if the RPM method is used, the IDE is installed in **opt/opencpi/gui** and the command **ocpigui** is provided to start the IDE.

New Features in the 1.5 Release

In the 1.5 release, the ANGRYVIPER team focused on getting the OpenCPI XML editors current with OpenCPI Framework standards. Every available component editor (excluding the Application and Assembly Editors) is updated. In addition, new editors are provided for the Properties File, Signals File, and Slot Signals File. These editors are not integrated with the OpenCPI Projects view but they will open if the respective XML file is selected via the Eclipse Project Explorer.

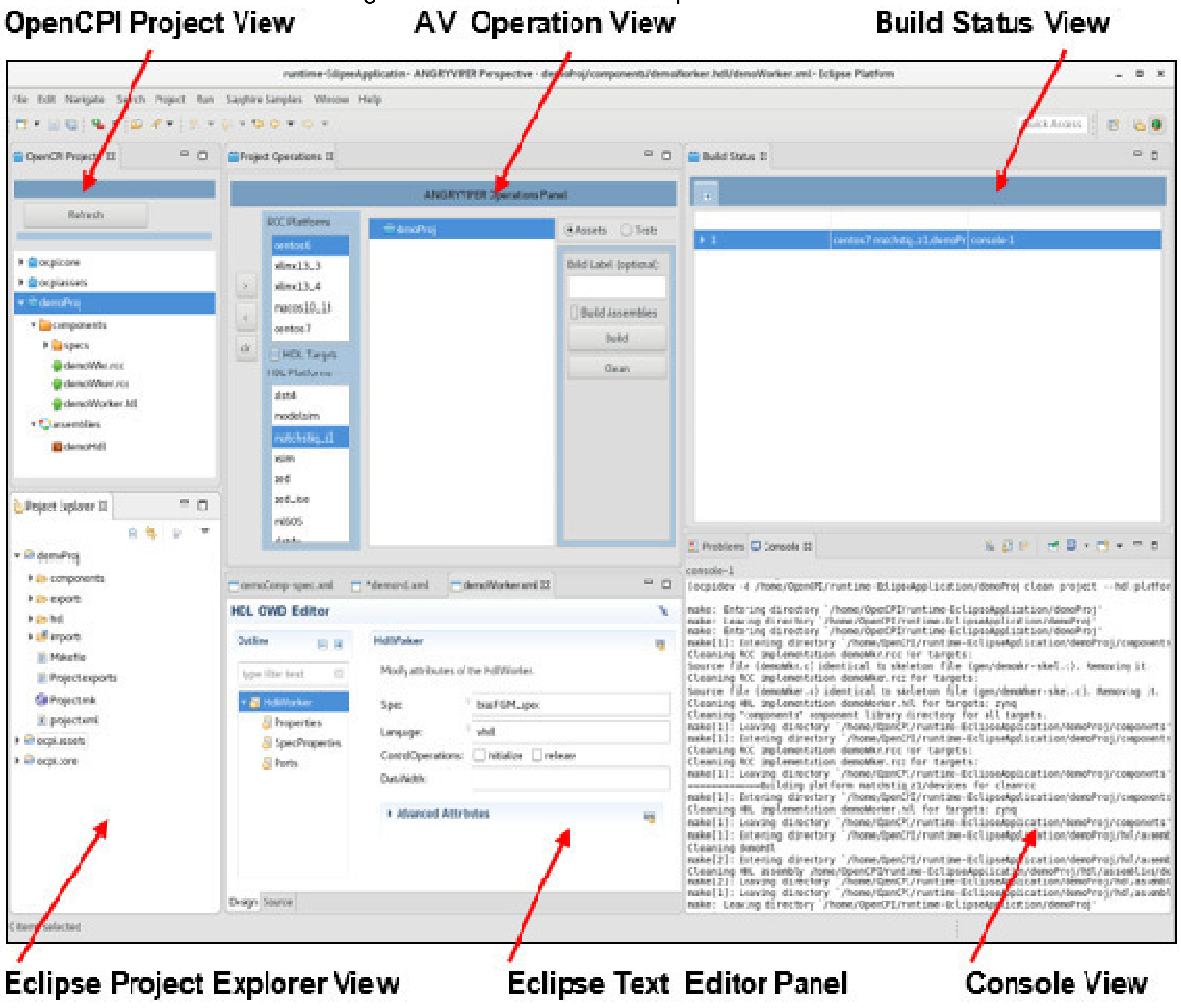
Overview

The face of the ANGRYVIPER AV IDE is now the **ANGRYVIPER (AV) Perspective**. This section provides an overview of the perspective layout and how to set it up in a new IDE installation.

AV Perspective Views

The figure below illustrates the new AV Perspective. It consists of detailed views and the Eclipse Editor panel.

Figure 1: ANGRYVIPER Perspective Views



```
console-1
locpidev -d /home/OpenCPI/runtime-EclipseApplication/demoProj clean project --hdl-platfor

make: Entering directory `/home/OpenCPI/runtime-EclipseApplication/demoProj'
make: Leaving directory `/home/OpenCPI/runtime-EclipseApplication/demoProj'
make: Entering directory `/home/OpenCPI/runtime-EclipseApplication/demoProj'
make[1]: Entering directory `/home/OpenCPI/runtime-EclipseApplication/demoProj/components
Cleaning RCC implementation demoWkr.rcc for targets:
Source file (demoWkr.c) identical to skeleton file (gen/demoWkr-skel.c). Removing it.
Cleaning RCC implementation demoWker.rcc for targets:
Source file (demoWker.c) identical to skeleton file (gen/demoWker-skel.c). Removing it.
Cleaning HDL implementation demoWorker.hdl for targets: zynq
Cleaning "components" component library directory for all targets.
make[1]: Leaving directory `/home/OpenCPI/runtime-EclipseApplication/demoProj/components'
make[1]: Entering directory `/home/OpenCPI/runtime-EclipseApplication/demoProj/components
Cleaning RCC implementation demoWkr.rcc for targets:
Cleaning RCC implementation demoWker.rcc for targets:
make[1]: Leaving directory `/home/OpenCPI/runtime-EclipseApplication/demoProj/components'
============Building platform matchstiq_z1/devices for cleanrc
make[1]: Entering directory `/home/OpenCPI/runtime-EclipseApplication/demoProj/components
Cleaning HDL implementation demoWorker.hdl for targets: zynq
make[1]: Leaving directory `/home/OpenCPI/runtime-EclipseApplication/demoProj/components'
make[1]: Entering directory `/home/OpenCPI/runtime-EclipseApplication/demoProj/hdl/assemb
Cleaning demoHdl
make[2]: Entering directory `/home/OpenCPI/runtime-EclipseApplication/demoProj/hdl/assemb
Cleaning HDL assembly /home/OpenCPI/runtime-EclipseApplication/demoProj/hdl/assemblies/de
make[2]: Leaving directory `/home/OpenCPI/runtime-EclipseApplication/demoProj/hdl/assemb
make[1]: Leaving directory `/home/OpenCPI/runtime-EclipseApplication/demoProj/hdl/assemb
make: Leaving directory `/home/OpenCPI/runtime-EclipseApplication/demoProj'
```

Below are brief summaries of each of the perspective views, full descriptions are provided in the later sections.

OpenCPI Projects View – an explorer giving a flattened view of the projects and assets. In the 1.4 release this view has become the primary tool to find and open asset XML files and its context menu has a rich set of features: asset wizard, open, build, clean, delete, and project registration. To open the menu simply right-click anywhere in this view.

AV Operations View– provides platform selection and controls to build assets and run tests. The AV Operations View layout has changed to provide a full feature application unit testing capability in addition to building and cleaning assets.

Build Status View – provides a graphical view of build/run execution configurations and execution status. A run configuration gets a color coded status bar that visually relates status. The status bar expands to provide details and execution times of selections addressed in the run. The user can also re-run builds and tests using this view.

Project Explorer View – Provides a view into a projects file system. This view primarily supports code development. It is included in the perspective for reasons provided in the Overview Section.

Eclipse Editor panel – Asset and text editors open in this panel

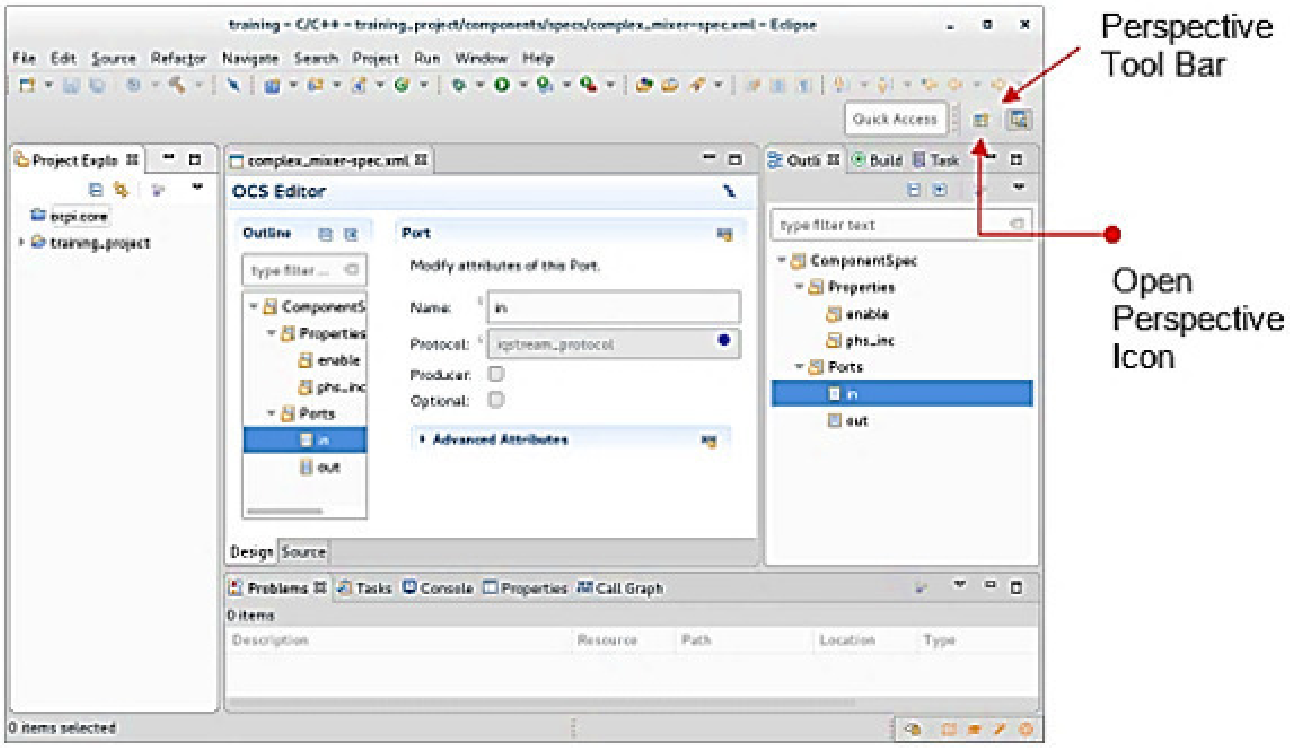
Eclipse Console View – Shows the various ocpidev command executions based on user requests as well as the result of the requests.

Open the AV Perspective

In a new Eclipse installation, the current default behavior is that Eclipse will open in the selected workspace and will display the C/C++ Perspective shown below.

Note: The AV Perspective must be added to the Eclipse perspectives toolbar for regular use.

Figure 2: IDE showing C/C++ Perspective


training - C/C++ - training_project/components/specs/complex_mixer-spec.xml - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Project Explorer
ocpi.core
training_project
complex_mixer-spec.xml
OCS Editor
Outline
Port
Modify attributes of this Port.
Name: in
Protocol: iqstream_protocol
Producer:
Optional:
Advanced Attributes
Design Source
Outline Build Task
ComponentSpec
Properties
enable
phs_inc
Ports
in
out
Problems Tasks Console Properties Call Graph
0 items
Description Resource Path Location Type
0 items selected
Perspective Tool Bar
Open Perspective Icon

Adding the Perspective to the Toolbar

To open the Perspective and add it to the Perspectives Toolbar:

- Click the Open Perspective Icon
- Select it from the Perspective List

  (Eclipse will switch to the perspective and add the AV Icon to the perspectives toolbar)
- Click through the toolbar selections to change between perspectives

  Note: If the AV Perspective is not shown in this Open Perspective window, make sure the av.proj.ide. plugin is in your Eclipse installation

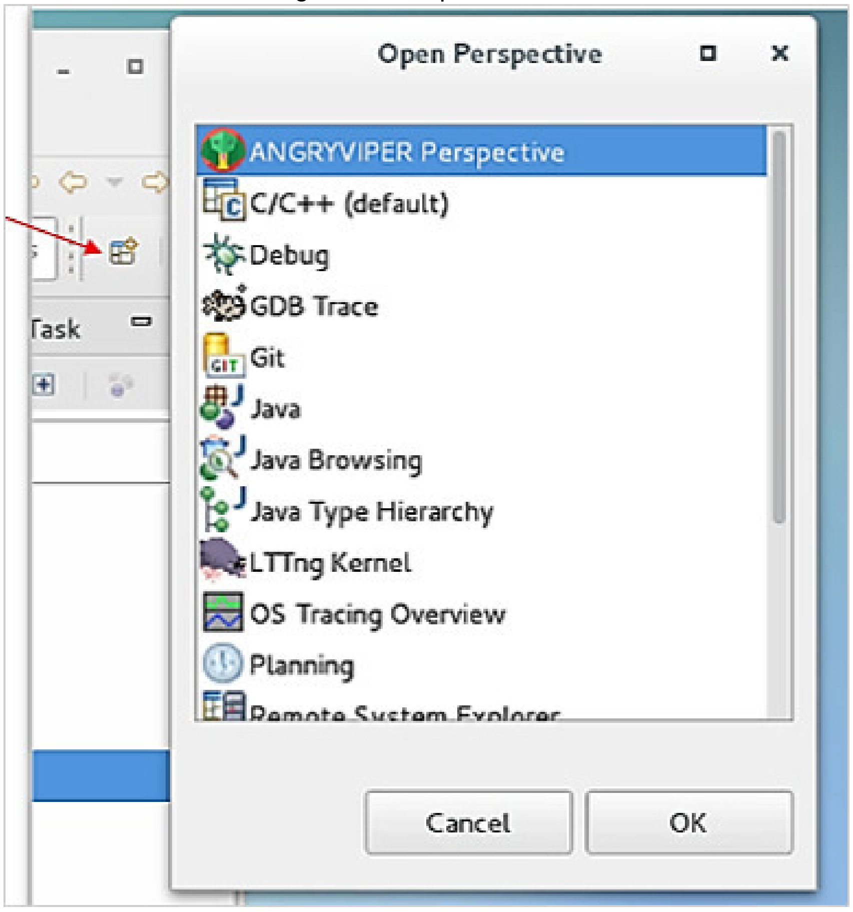
  For personalization, Eclipse allows the user to rearrange the views in the perspective and also to add views. It will then save the rearranged perspective and will display it from this way forward, if the workspace is preserved.

To see a listing of Eclipse perspectives:

- Navigate to: Window
- Navigate to: Perspective
- Open Perspective

  (A listing of views can be seen by navigating to: Window and then Show View)

Figure 3: PerspectiveView



IDE Overview and Features

Overview

The AV Perspective default layout was selected because it provides a core complement of tools to accomplish OpenCPI operations and it provides the most OpenCPI project features for the user. It is now the main display for the IDE. This section provides feature details for each of the Eclipse views that make up the perspective.

OpenCPI Projects View Features

The OpenCPI Projects View provides navigation into a flattened view of OpenCPI projects. It currently displays OpenCPI projects and a subset of the OpenCPI assets that it supports. A right-click context menu provides features appropriate to the selected asset.

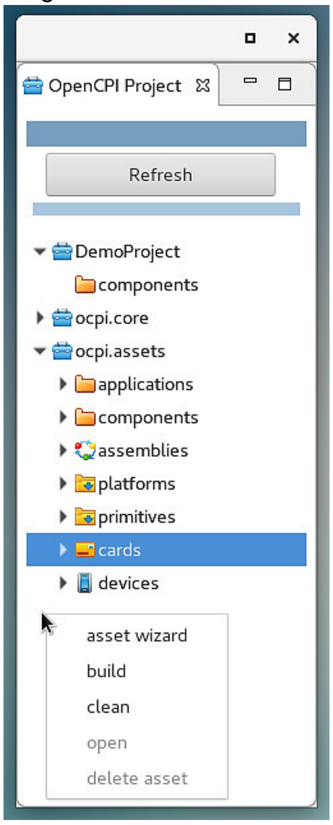
The user has the following features to choose from:

- Asset Wizard – Opens the Asset Wizard
- Build
- Clean
- Open - When enabled, opens the assets XML file in the respective editor (double-clicking on an individual asset it will open it as well)
- Delete Asset – Removes the asset and its respective artifacts
- Build or clean can be executed from any level in the tree from entire projects to individual assets like workers and applications.

Context features are added to the menu based on the current [single] selection:

- Project selected – Opens Register or Unregister depending on the registration state of the project

Figure 4: List of Features



- Components selected – New Component, New Protocol, New Worker, New Unit Test
- Applications, assemblies, primitives Selected – Creates a new respective asset. It is also used to select assets to be added to the Operations View

Building from the OpenCPI Projects View

- Select one or more assets in the view and right-click
- Select Build or Clean from the context menu

An execution configuration is constructed from the selections, the platform selections and other inputs from the Operations panel The figure below displays a launch from the OpenCPI Projects View and the Status Bar for the build

Operations View Features

The Operations View is used to build OpenCPI assets and to build and run application component unit tests

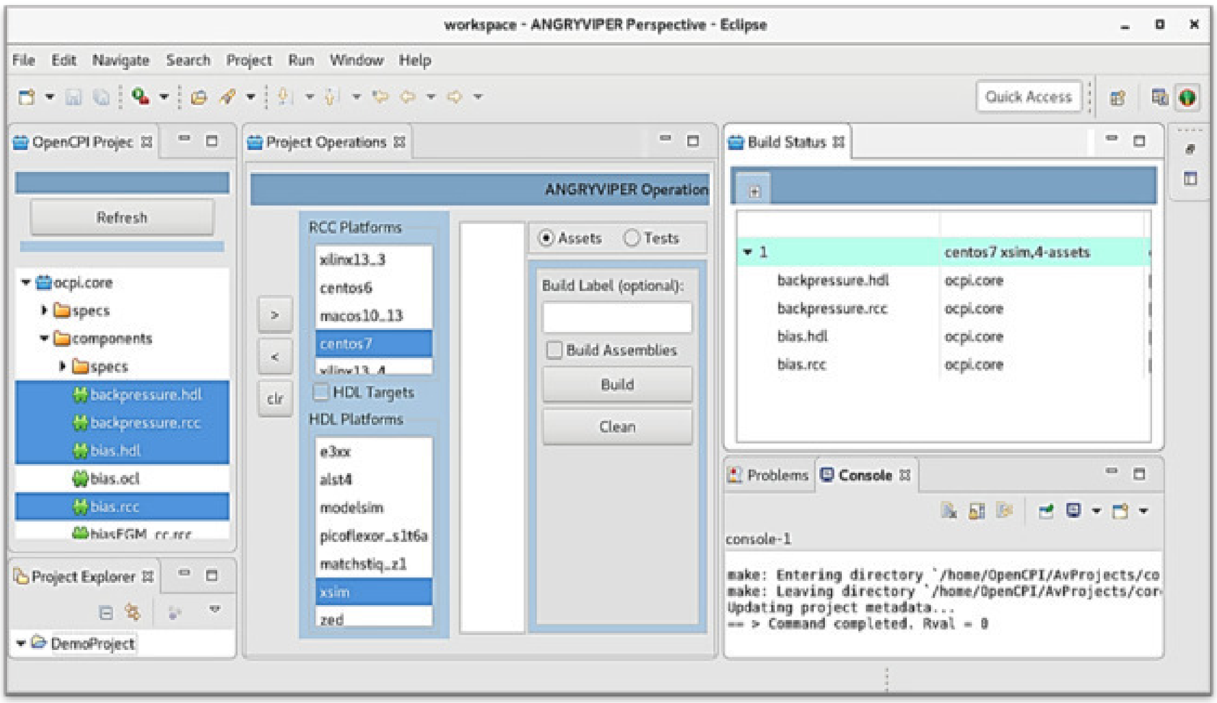
This panel has two modes of operation:

1. Assets Mode- supports assets build and clean operations
2. Test Mode- supports unit test operations

Core features of the panel are:

- Add selections in the Projects View to the Operations panel (> button)
- Remove selected assets from the panel (< button)
- Clear the panel (clr button)
- Make RCC/HDL platform build selections or HDL target selections
- Execute build or clean on the assets in the Operations panel in the order they appear

Figure 5: Projects View



```
make: Entering directory `/home/OpenCPI/AvProjects/co
make: Leaving directory `/home/OpenCPI/AvProjects/cor
Updating project metadata...
== > Command completed. Rval = 0
```

- Build and run unit tests

To add assets to the Operations Panel:

- Select one or more assets in the Projects panel (use the CTRL or SHIFT keys similarly to working with an email app)
- Press the add button
- Select the platforms for the build

  The radio buttons on the top of the Build Controls section toggle controls for building assets and running unit tests.

Assets Mode

A build or clean execution is initiated by the Build/Clean buttons. Once it is initiated, an execution configuration is established. The user may repeat a build or clean execution on this configuration from this panel or the Status Monitor View.

The figure below demonstrates a build executed from the Operations panel. A status bar for the completed build is expanded to show the build order. The console view lists each ocpidev build command and output from the command.

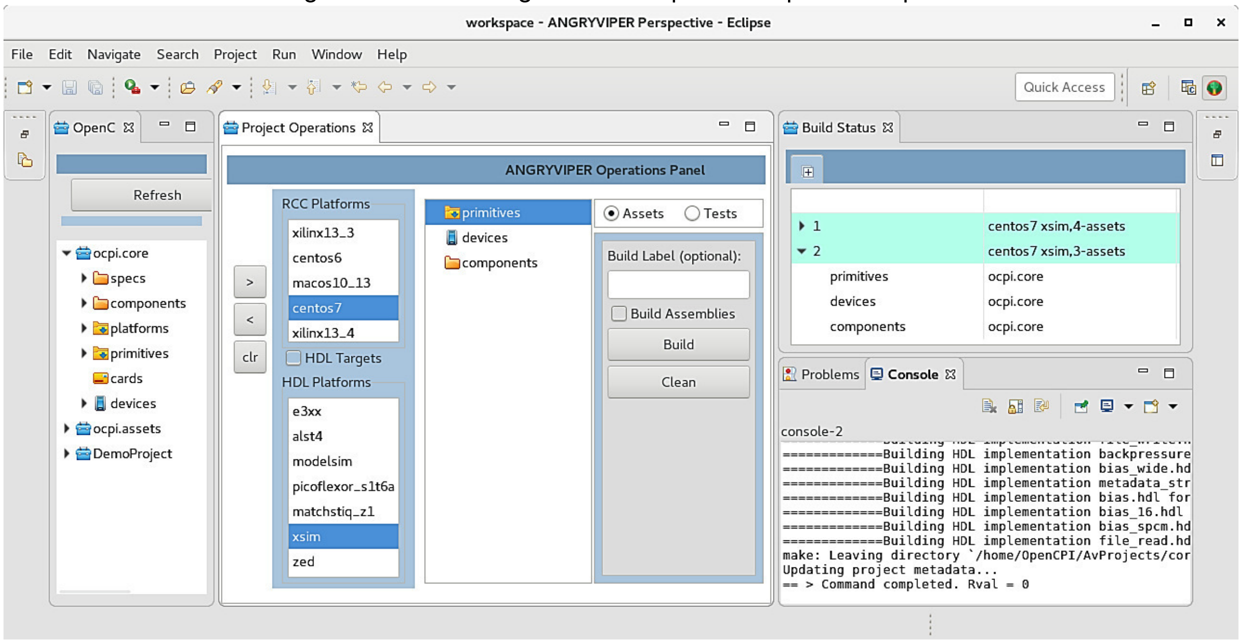
If the execution configuration is rerun, the corresponding status bar and console will show the progress of the execution. Building assets can occur at all levels of an OpenCPI project:

- Project
- Second-Level Asset folders
- Individual Assets

Unit Test panel

The Unit Tests Panel (shown in the below figure) supports most of the features for the unit testing of workers which is described in the OpenCPI Component Development Guide. Reference the Component Development Guide for a complete description of the unit test of workers and details of the five phases of unit testing. The five phases of unit testing are:

Figure 6: Build configuration setup in the Operations panel



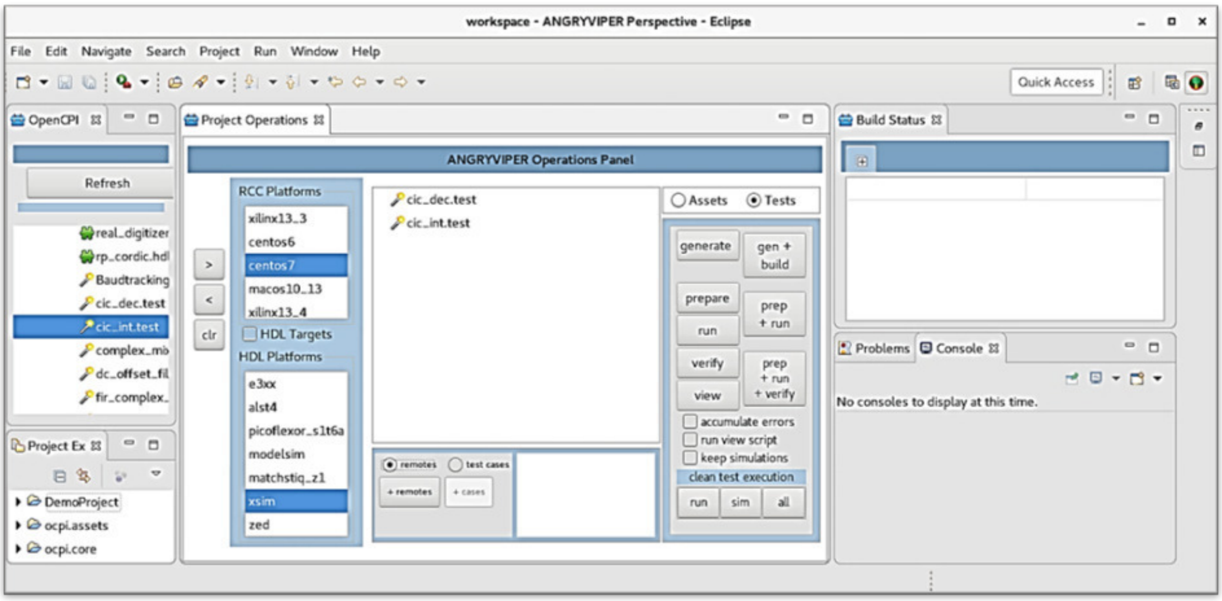
1	centos7 xsim,4-assets
2	centos7 xsim,3-assets
primitives	ocpi.core
devices	ocpi.core
components	ocpi.core

```
console-2
============Building HDL implementation backpressure
============Building HDL implementation bias_wide.hd
============Building HDL implementation metadata_str
============Building HDL implementation bias.hdl for
============Building HDL implementation bias_16.hdl
============Building HDL implementation bias_spcm.hd
============Building HDL implementation file_read.hd
make: Leaving directory `/home/OpenCPI/AvProjects/cor
Updating project metadata...
== > Command completed. Rval = 0
```

1. Generate
2. Build
3. Prepare
4. Run
5. Verify

The tests panel provides control for these 5 phases, *(currently, there is no discrete support for the Build Phase)* and it provides three buttons that combine unit test phases that typically support development and debugging of a unit test.

Figure 7: Unit Test Panel



No consoles to display at this time.

Unit Test Features

The following is a list of test features and the equivalent development guide operation or argument:

- View Button = View Operation
- Run View Script = View =1
- Accumulate Errors = Test Accumulate Errors =1
- Keep Simulations = KeepSimulations =1
- Test Cases = Cases
- Remotes = OCPI_REMOTE_TEST_SYSTEMS

The content of a unit test is provided in the form of: *run* content, *sim* (simulations data) or *all* (gen/ and run/). Use the buttons in the clean test execution section to remove this content.

Caution! *Simulation directories may become quite large and consume an alarming amount of storage*.

Similar to the Assets Mode, the Unit Test Mode constructs and executes *ocpidev* command strings to perform the various phases of unit testing.

Building of the unit tests is possible via the Assets or Tests panel. However only the Test panel supports the other phases of unit test. Any test listed in the Operations panel is executed in sequence.

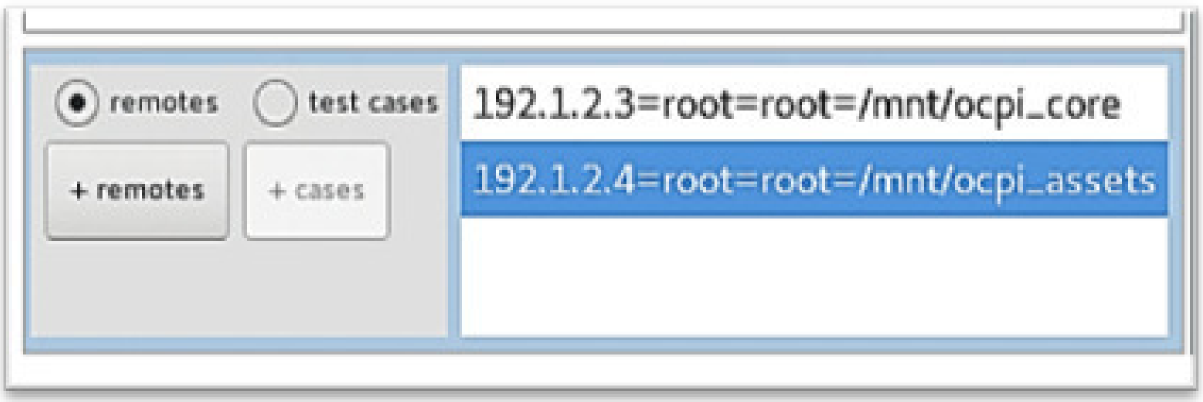
Prior to execution of a test phase, the user must have selected desired RCC/HDL platforms, test cases and remotes. Multiple platforms, test cases and remotes may be highlighted, but only one project for a remote system can be active. For example:

- `Valid :192.168.2.9=root=root=/mnt/ocpi_core:192.168.2.10=root=root=/mnt/ocpi_assets`
- `Invalid :192.168.2.9=root=root=/mnt/ocpi_core:192.168.2.9=root=root=/mnt/ocpi_assets`

The bottom panel is used to enter remote systems and test cases. The remote/test cases radio button toggles the two operations. Click the Remotes button to add a remote system via a pop-up dialog.

In the example below, two remotes are available; the remote selected is placed in the next run. Test cases are added and selected in the same manner. Multiple entries may be selected for a run. The list panel has a right-click menu with edit and delete options for selected entries.

Figure 8: List panel with 2 Different Remotes



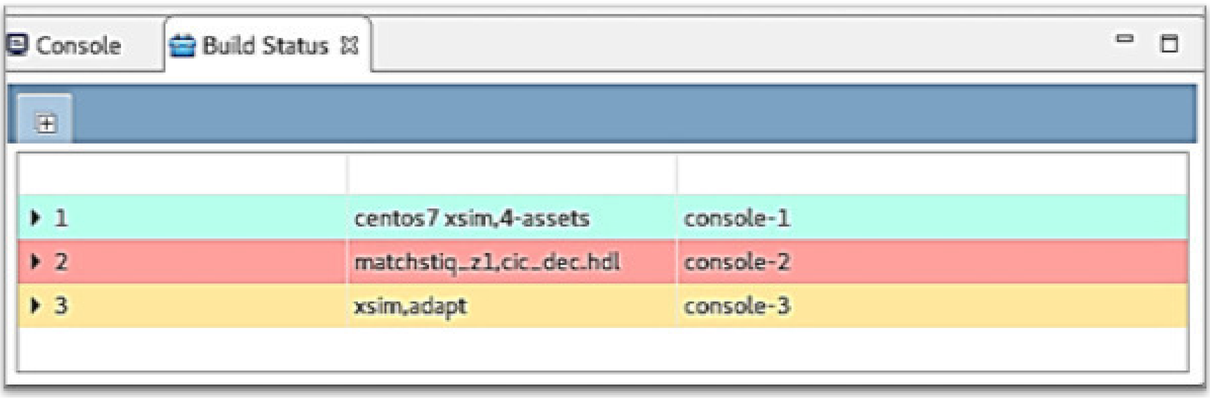
192.1.2.3=root=root=/mnt/ocpi_core
192.1.2.4=root=root=/mnt/ocpi_assets

Build Status View Features

Each execution configuration has a corresponding status bar in the Build Status View. The color of the bar represents the build status:

- White- Indicates an active execution
- Green- Indicates a successful completion
- Red- Indicates a failure

Figure 9: Build Status View

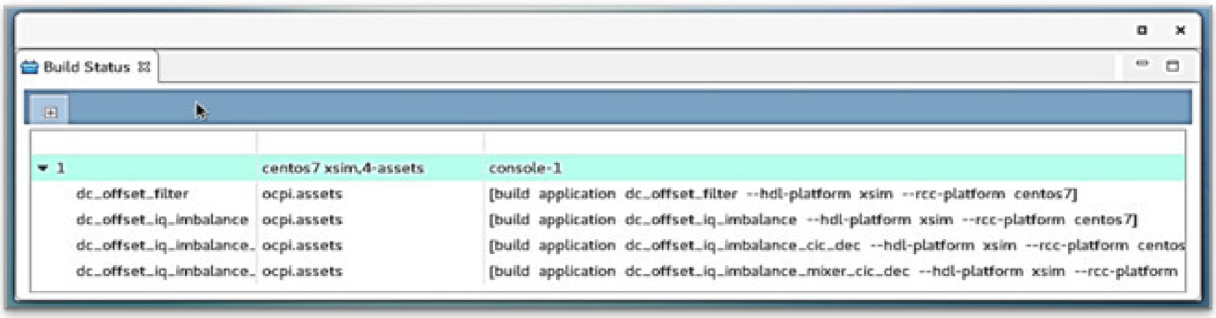


1	centos7 xsim,4-assets	console-1
2	matchstiq_z1,cic_dec_hdl	console-2
3	xsim,adapt	console-3

- Yellow- Indicates an execution stopped by the user

The Status Bar expands to provide details about the execution and the sequence in which they occurred. In an active execution this list is dynamic; rows are added as the execution proceeds.The figure below demonstrates the Status Bar for a build that has completed successfully and the expanded view.

Figure 10: Successful Build Execution



1	centos7 xsim,4-assets	console-1
dc_offset_filter	ocpi.assets	[build application dc_offset_filter --hdl-platform xsim --rcc-platform centos7]
dc_offset_iq_imbalance	ocpi.assets	[build application dc_offset_iq_imbalance --hdl-platform xsim --rcc-platform centos7]
dc_offset_iq_imbalance_	ocpi.assets	[build application dc_offset_iq_imbalance_cic_dec --hdl-platform xsim --rcc-platform centos
dc_offset_iq_imbalance_	ocpi.assets	[build application dc_offset_iq_imbalance_mixer_cic_dec --hdl-platform xsim --rcc-platform

By using right-click on the status bar, several actions can be selected: **Build, Clean, Run, Stop, Delete**. As an example: an inactive build execution can be rerun as a build or a clean.

Similarly, the user may **stop** an active execution. Finally, an inactive status bar can be deleted and when selecting a build status bar, the console for that build is brought into view.

Eclipse Project Explorer View

The Eclipse Project Explorer is provided in the Perspective because it provides a *file system view* of the projects. A right-click context menu provides features that are appropriate to the selected asset.

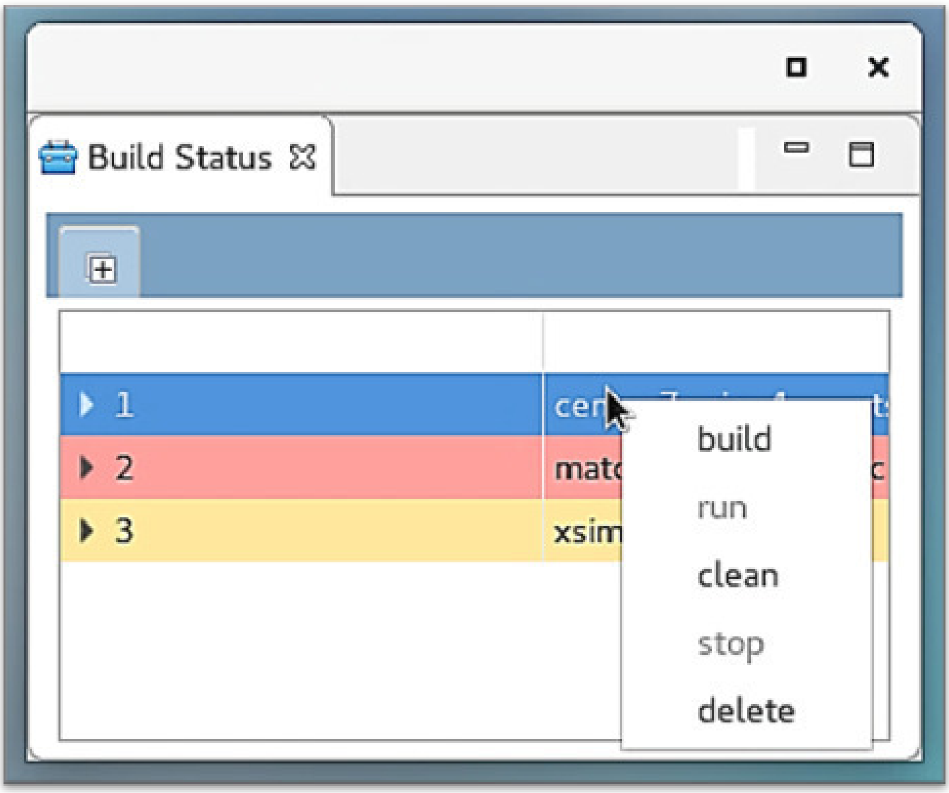
The user has the following features to choose from:

This view has a right-click context menu that provides workspace features such as:

- **Project Import**
- **Workspace/Project Refresh**
- **Access to the Assets Wizard (via the New Selection)**

It is the view Eclipse provides in the C/C++ Perspective and it is best suited to support code development. However, it can be used to open OpenCPI XML files in an XML Editor and it can open supported assets in the respective graphical editor.

Figure 11: Status Bar



Unfortunately, it is the only view that currently supports drag and drop into the Application and Assembly Editors. Most of the core OpenCPI asset management features are now implemented in the OpenCPI Projects View.

Caution! Do not delete OpenCPI assets by deleting their top-level folders in this view.

Use the **Asset Delete** provided in the OpenCPI Projects View - it executes the underlying ocpidev delete command that completely removes the asset from the framework infrastructure.

Eclipse Console View

The IDE leverages the Eclipse Console View to allow the user to readily see execution output for multiple build and test runs. The IDE supports a notice console and up to fifty execution consoles.

There are useful features in the view toolbar: **Clear, Scroll Lock, Display Selected Console**.

Note: Execution configurations will clear the respective console and bring them into view when they are rerun.

Notice Console

The **Notice Console** is used to communicate messages typically seen in a log. These are messages about issues in reading the environment, output of ocpidev create, delete messages, and resource issues.

View the Notice Console for notices if something unexpected happens and the IDE does not indicate a problem via a pop-up dialog.

Execution Consoles

Execution consoles provide **build and run** commands and their output.The IDE supports up to fifty (50) active execution configurations at a time and each configuration gets its own console. The figure below shows a console for a successful build.

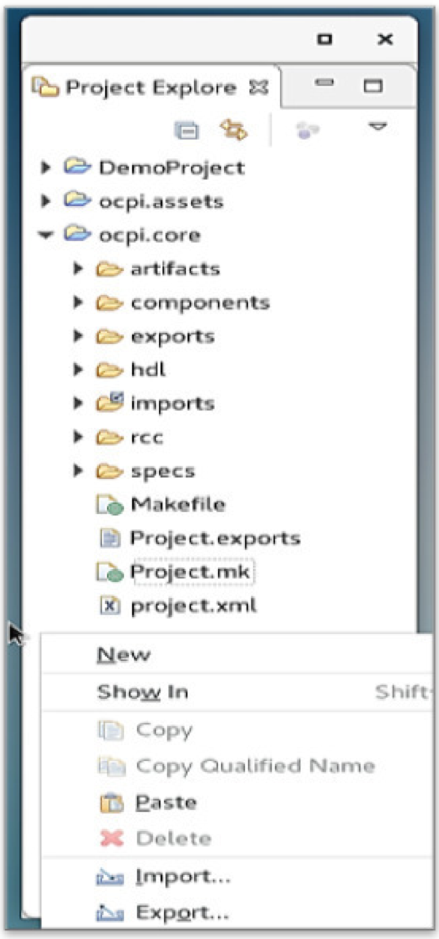
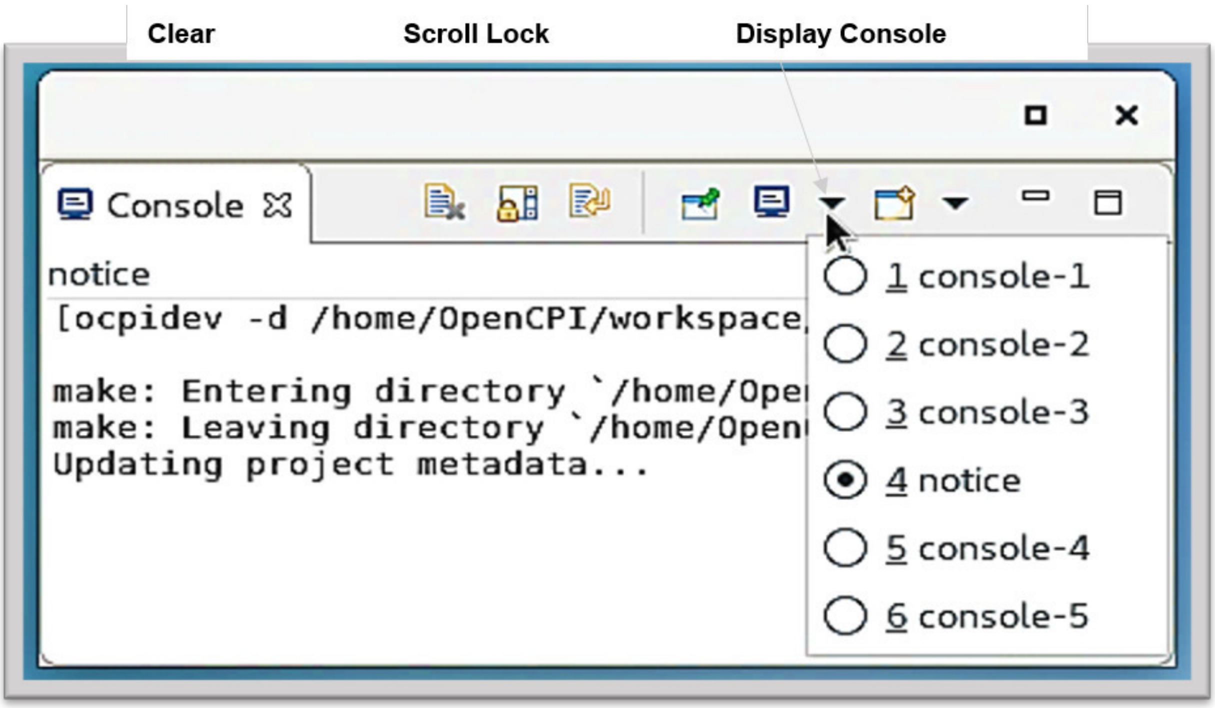


Figure 12: Eclipse Project Explorer View



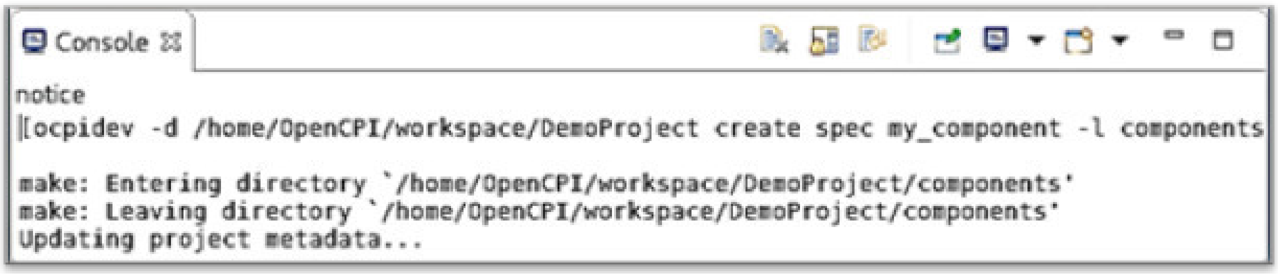
```
notice
[ocpidev -d /home/OpenCPI/workspace,

make: Entering directory `/home/Ope
make: Leaving directory `/home/Open
Updating project metadata...
```

Figure 13: Eclipse Console View

OpenCPI Asset Wizard

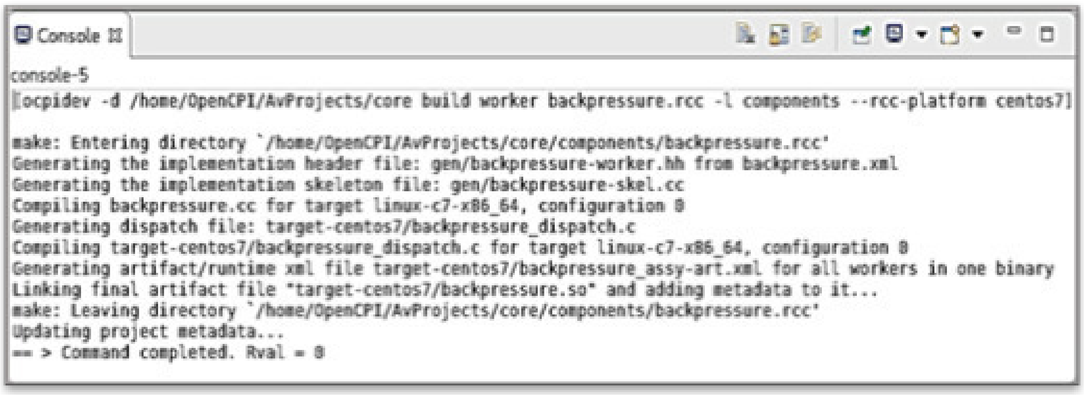
The **Asset Wizard** provides a tool to create a number of OpenCPI Assets. It provides a simple form for required and optional parameters to create the asset (accomplished by ocpidev create). Upon completion, the asset and

Figure 14: Notice Console View



```
notice
[ocpidev -d /home/OpenCPI/workspace/DemoProject create spec my_component -l components

make: Entering directory `/home/OpenCPI/workspace/DemoProject/components'
make: Leaving directory `/home/OpenCPI/workspace/DemoProject/components'
Updating project metadata...
```

Figure 15: Execution Console



```
console-5
[ocpidev -d /home/OpenCPI/AvProjects/core build worker backpressure.rcc -l components --rcc-platform centos7]

make: Entering directory `/home/OpenCPI/AvProjects/core/components/backpressure.rcc'
Generating the implementation header file: gen/backpressure-worker.hh from backpressure.xml
Generating the implementation skeleton file: gen/backpressure-skel.cc
Compiling backpressure.cc for target linux-c7-x86_64, configuration 0
Generating dispatch file: target-centos7/backpressure_dispatch.c
Compiling target-centos7/backpressure_dispatch.c for target linux-c7-x86_64, configuration 0
Generating artifact/runtime xml file target-centos7/backpressure_assy-art.xml for all workers in one binary
Linking final artifact file "target-centos7/backpressure.so" and adding metadata to it...
make: Leaving directory `/home/OpenCPI/AvProjects/core/components/backpressure.rcc'
Updating project metadata...
== > Command completed. Rval = 0
```

follow-on OpenCPI framework folders and files will be created.

The new asset is displayed in the project views. If an XML Editor is provided in the IDE, then the respective Editor is opened to further populate the XML specification. The figure below shows all of the available assets in the drop-down list that the wizard supports. This OpenCPI **Asset Creation Wizard** may be selected from the context menu obtained by right-clicking anywhere in OpenCPI Projects View.

It is also found under the New selection (or New Other) Eclipse Project Explorer View context menu or under the File Tab in the top Eclipse menu bar.

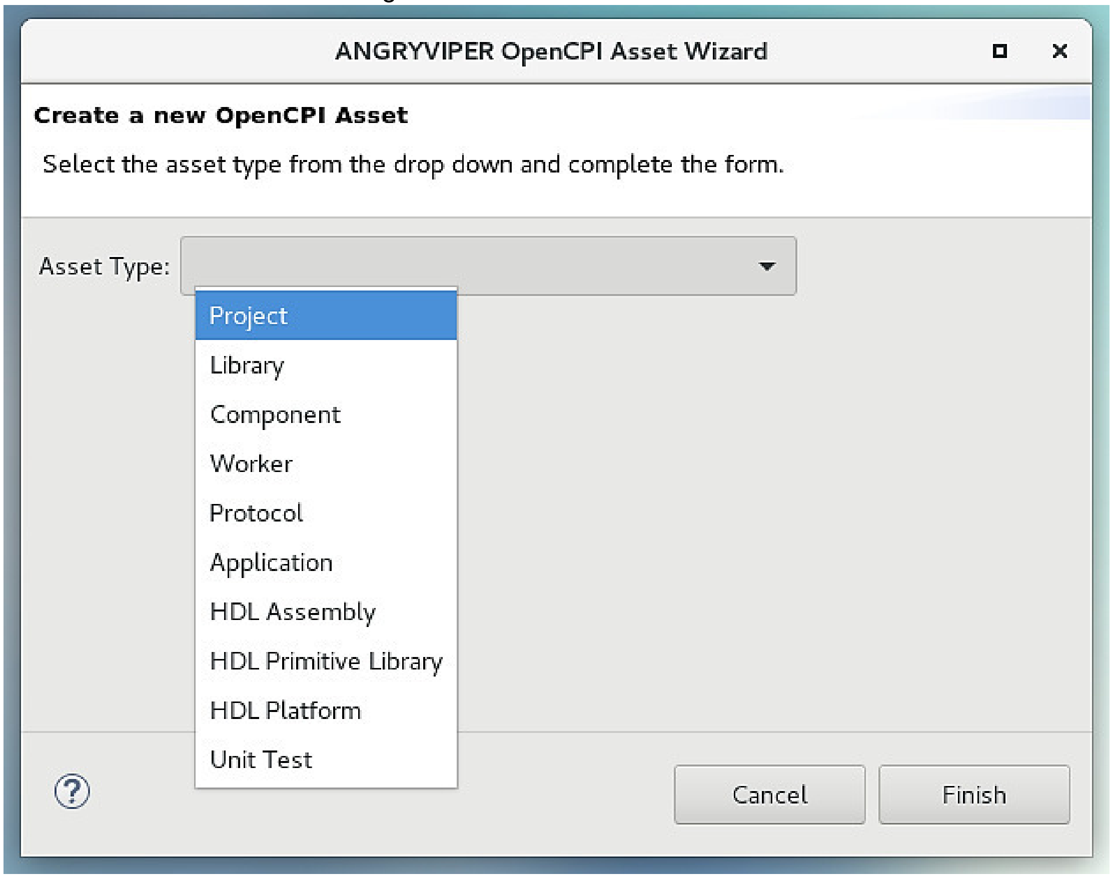
Caution! There are some dependencies to create assets. An OpenCPI project must be open in the workspace before any other assets can be created. Also, a components library must be established before components and workers can be created.

OpenCPI Asset XML Editors

The IDE supports the creation of the following OpenCPI Assets and provides graphical editors to populate their XML files:

- **Component Spec** – OCS Editor
- **Protocol Spec** – OPS Editor
- **Component Properties File** – Properties File Editor
- **RCC Application Worker** – OWD RCC Editor
- **HDL Application Worker**– OWD HDL Editor
- **HDL Assembly** – OHAD Editor
- **HDL Signals File** – Signals File Editor
- **HDL Slot File** – Slot File Editor
- **OpenCPI Application** – OAS Editor

Figure 16: Asset Creation Wizard



- **HDL Platform Worker** – HDL Platform Editor
- **Component Unit Tests** – Unit Test Editor

There are two implementations of the XML editors in the plugin. The **Component** editors are *form-based* while the **Application and Assembly** editors are *diagram-based* and support the form-based feature.

Form Based Graphical Editors

The Graphical Editor has two panels:

1. **Outline** panel
2. **Form-Input** panel

In the example below, the Outline Panel is used to navigate the XML elements for this worker. These are the Top-Level HdlWorker Elements and Property, SpecProperty and Port (StreamInterface) child elements. Select the element in the outline to add a new one or select an existing one to see or and modify it. The figure below is an example of adding a property element to the worker. Select the Properties element in the outline, then Right Click and Add Property. The form to populate the new element appears as shown in the Form Input panel image below.

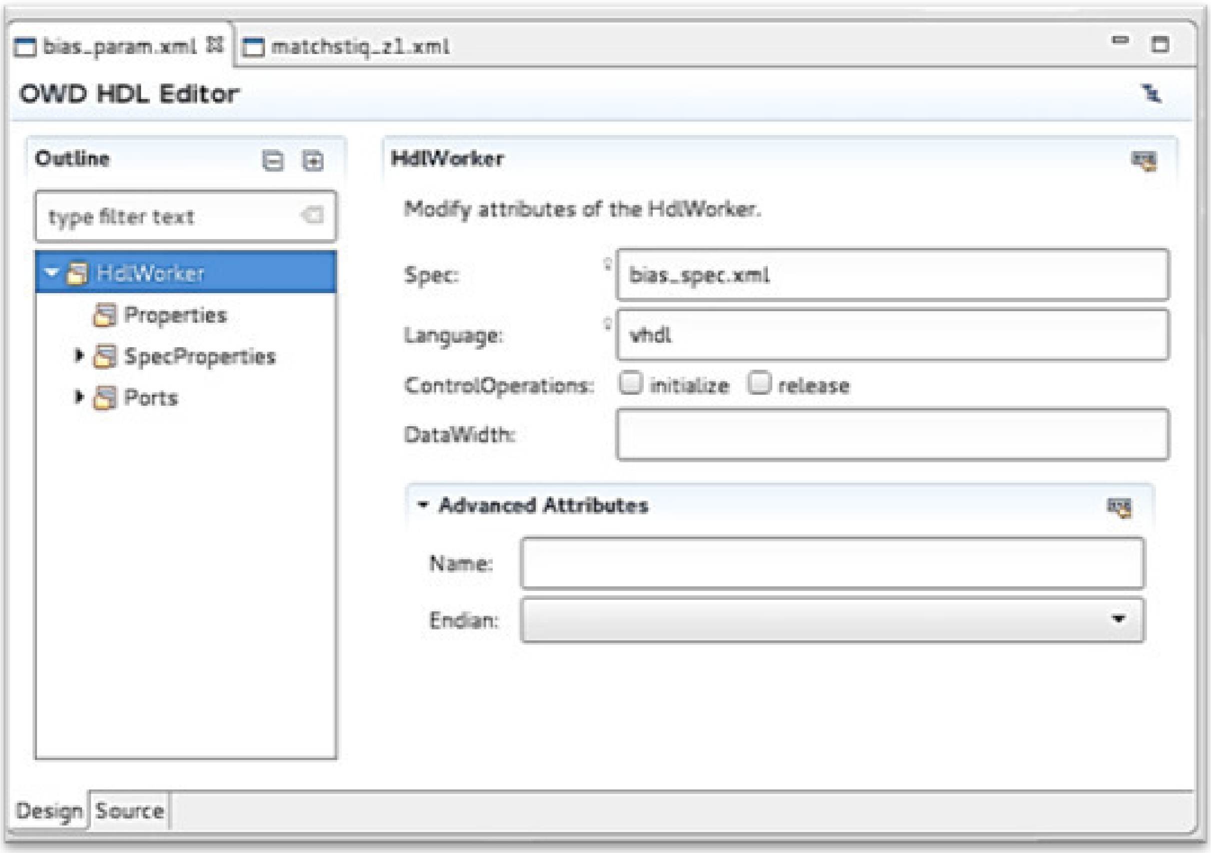
Select the Source Tab to see how the new element is added to the XML file.
To remove an element:

- Select
- Right Click
- Select Delete

The source file may also be edited and those changes will appear in the design view.

See the Component Development Guide for more information about the XML files these editors support.

Figure 17: Modify Attributes of HDL Worker



bias_param.xml matchstiq_z1.xml

OWD HDL Editor

Outline

type filter text

- HdlWorker
  - Properties
  - SpecProperties
  - Ports

HdlWorker

Modify attributes of the HdlWorker.

Spec: bias_spec.xml

Language: vhdl

ControlOperations: initialize release

DataWidth:

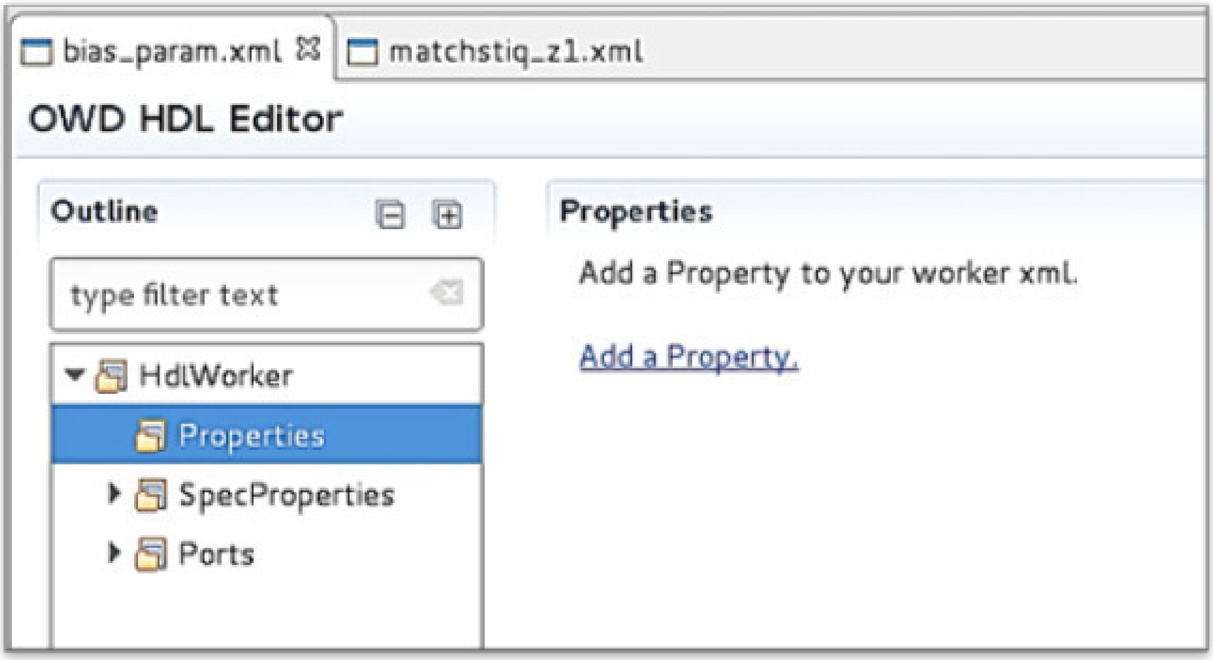
Advanced Attributes

Name:

Endian:

Design Source

Figure 18: Add a Property to a Worker



bias_param.xml matchstiq_z1.xml

OWD HDL Editor

Outline

type filter text

- HdlWorker
  - Properties
  - SpecProperties
  - Ports

Properties

Add a Property to your worker xml.

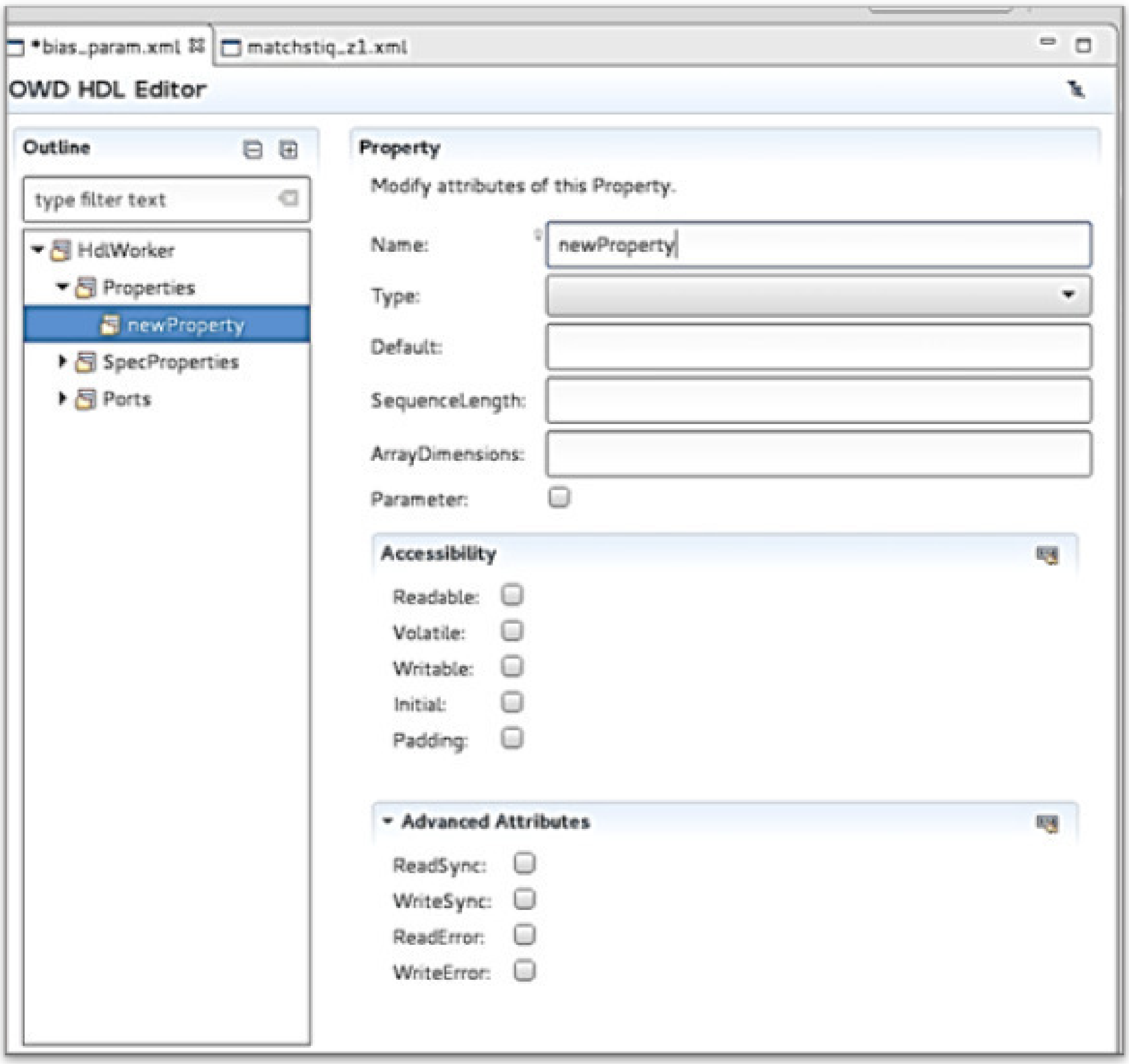
Add a Property.

Note: In Release 1.5 these editors were updated to current OpenCPI standards.

Graphical Drag-and-Drop Editors

The Application and Assembly Editors provide a **diagram interface** to generate the respective XML files.

Figure 19: HDL Property Element Form



Name: newProperty
Type:
Default:
SequenceLength:
ArrayDimensions:
Parameter: ☐

Accessibility

Readable: ☐
Volatile: ☐
Writable: ☐
Initial: ☐
Padding: ☐

Advanced Attributes

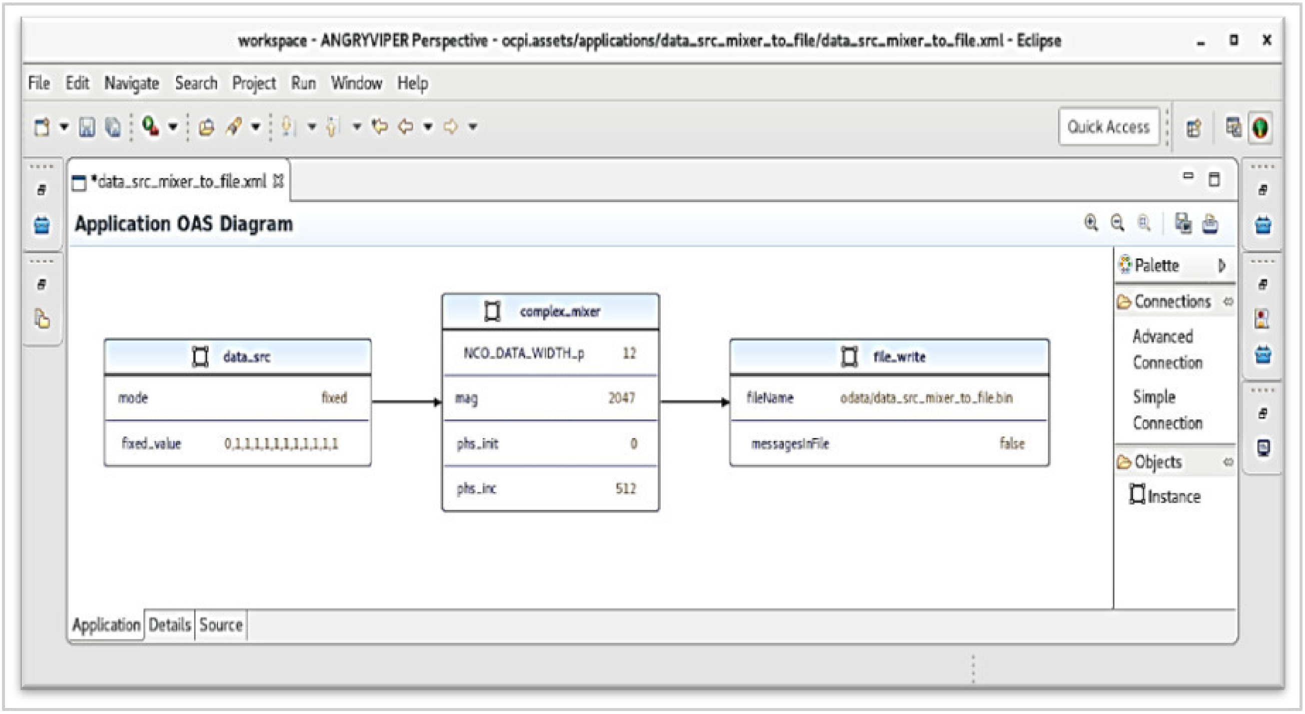
ReadSync: ☐
WriteSync: ☐
ReadError: ☐
WriteError: ☐

Asset XML Examples

The following list provides OpenCPI asset XML to examine in its respective Editor. Use the OpenCPI Projects View to navigate into the OpenCPI Core and Assets projects and open the various asset XML files by doubling-clicking the asset.

1. **OPS Editor-** core-specs-iqstream_protocol.xml
2. **OCS Editor-** core-components-specs-bias_spec.xml
3. **OWD RCC Editor-** core-components-bias_cc.rcc
4. **OWD HDL Editor-** core-components-bias_param.hdl
5. **HDL Assemble Editor-** assets-assemblies-cic_int_dc_offset_iq_imbalance_mixer_cic_dec
6. **OAS Editor-** assets-applications-data_src_mixer_to_file
7. **Platform Worker Editor-** assets-platforms- matchstiq_z1
8. **Unit Test Editor-** core-components-metadata_stressor.test

Figure 20: Graphical Drag and Drop Editors


workspace - ANGRYVIPER Perspective - ocpi.assets/applications/data_src_mixer_to_file/data_src_mixer_to_file.xml - Eclipse
File Edit Navigate Search Project Run Window Help
Quick Access
*data_src_mixer_to_file.xml
Application OAS Diagram
data_src
mode fixed
fixed_value 0,1,1,1,1,1,1,1,1,1,1,1
complex_mixer
NCO_DATA_WIDTH_p 12
mag 2047
phs_init 0
phs_inc 512
file_write
fileName odata/data_src_mixer_to_file.bin
messagesInFile false
Palette
Connections
Advanced Connection
Simple Connection
Objects
Instance
Application Details Source

Additional Notes About the Perspective

The ANGRYVIPER Team hopes to continue to add capabilities to the *OpenCPI Projects View* to further support OpenCPI operations. The goal for Version 1.4 was to give complementary features. This section provides a brief discussion to make the user aware of a number of issues that may occur when using the IDE.

The Execution Configuration

The **Execution Configuration** provides a means to associate and manage resources tied to the execution. These are:

- Selected Assets and Platforms
- Execution Number
- Status Bar
- Console

The perspective allows limitless execution configurations, however, it only allows so many to exist at once since they tie together Eclipse resources such as the consoles. The current limit is fifty(50).

When the limit is met, the user will get a pop-up dialog that instructs the user to delete some status bars to free up resources. This is the only way to free up resources other than closing and re-launching Eclipse. The Status View allows multiple delections for this reason. Once done, execution numbers will continue to increase but consoles will be reused.

XML Editors-Modifying Existing XML Files

The following Editors will modify existing XML files when they are opened with the Editor. The IDE will indicate this is occurring and why; however, it will only do it once per Eclipse session.
The Editors are:

- **OAS Editor** - The Application Editor adds a name attribute to the instance element to better support presentations
- **HDL Platform Editor** - This Editor will update slot signal definitions to the current standard where signal name and direction are discretely stated. Also note that if signal definitions were updated by the Version 1.4 GUI, these will have an unnecessary extension tag. If present, these may be removed manually.

The signal and slots file editors will similarly update old signal definition formats (used a direction=<name> attribute).

The modifications do not have to be saved.

Note: Signal definition changes represent the current OpenCPI standard.

Warning about a Non-OpenCPI Project

The perspective obtains asset information from the top level project.xml files. If an Eclipse workspace project is encountered that does not have a project.xml file, it is assumed the project is not an OpenCPI project. This is communicated in the Notice Console. If the project is an OpenCPI project, then something is wrong in the project file structure that does not allow project.xml to be generated.

The Perspective is Not in Sync with the Workspace or the File System

The OpenCPI Projects View does not track changes to the Workspace or the files system. This means changes made to a project outside of the IDE will not be apparent in the project explorers nor seen in current asset lists within the IDE. This means assets created or removed via command line will not be presented in either the Eclipse Project Explorer or OpenCPI Projects Views until both views are refreshed. Refresh the Eclipse Project view first then the OpenCPI Projects View.

Note: Do not delete a project using Project Explorer or Shell Command. Use the ocpidev command to delete a project. This will ensure project registration information remains up to date.

OpenCPI Development Workflow Using the IDE

Introduction

The IDE supports a number of OpenCPI workflow concepts; beginning with setting up the OpenCPI core and assets projects and getting them built to support new project development. This section demonstrates a number of the fundamentals discussed in Sections 4 and 5 of the Getting Started Guide. This includes creating the DemoProject and then a number of assets that go in the project as described in the document. Additionally, the IDE provides an effective way to look at existing assets OpenCPI XML specifications.

OpenCPI the Core and Assets Projects

A number of existing projects are provided in OpenCPI Release for use in development. If they are to be used, they need to be set up in the development environment, registered, and imported into the IDE and built for the desired platforms to further support new development. Procedures to set these projects up:

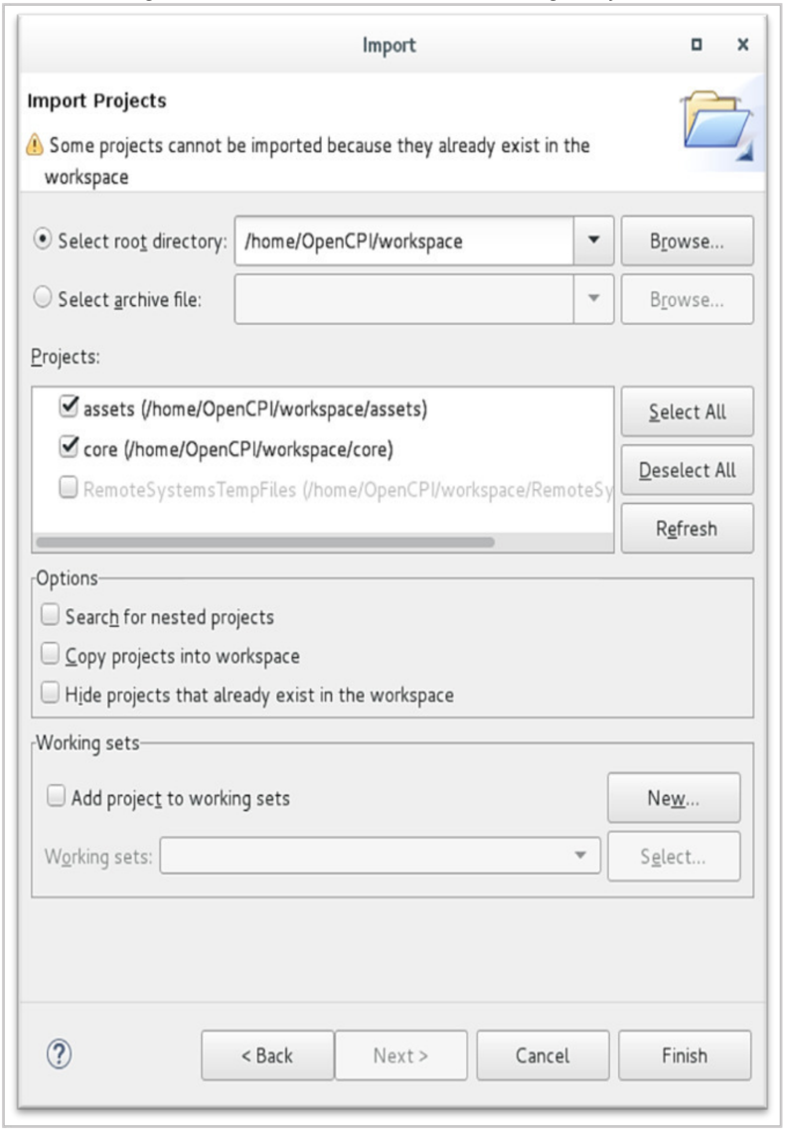
1. Select an area in the file system to put these projects and create the folder if necessary
2. Execute the project copy script ocpi-copy-projects to load these projects in desired folder and register them in that location (this script is interactive or arguments may be supplied)
3. Import the desired projects into Eclipse

Importing OpenCPI Core and Asset Projects

This demonstration describes the procedure to import the core and assets projects.

1. Place the cursor in the Eclipse Project Explorer panel
2. Right Click -Import-General-Existing projects into workspace - (This opens the import wizard)
3. Click Browse for the select root directory input
4. Navigate to and select the folder holding these projects
5. Select the checkboxes on the desired projects to import as shown in the preceding figure
6. Click Finish and the projects will be brought into the Eclipse workspace - (they will appear in the Eclipse Project Explore View)
7. Click Refresh in the OpenCPI Projects View to update that view with the imported projects

Figure 21: Procedures to use existing Projects


Import

Import Projects

⚠ Some projects cannot be imported because they already exist in the workspace

(●) Select root directory: /home/OpenCPI/workspace

( ) Select archive file:

Projects:

- [✓] assets (/home/OpenCPI/workspace/assets)
- [✓] core (/home/OpenCPI/workspace/core)
- [ ] RemoteSystemsTempFiles (/home/OpenCPI/workspace/RemoteSy

Options

- [ ] Search for nested projects
- [ ] Copy projects into workspace
- [ ] Hide projects that already exist in the workspace

Working sets

- [ ] Add project to working sets

Working sets:

Build the Projects for the desired RCC and HDL Platforms

There are a number of ways to build existing projects in the IDE because builds can be kicked off concurrently. If time is not an issue, put the core and assets projects in the Operations panel (core first followed by assets), Select the platforms to which to build to and launch the build (press build). Core will be built followed by assets. Concurrent building saves time-key things to know:

- HDL libraries in the core project must be built before starting a build on the assets project primitives
- Project primitives must be built first
- HDL card and device libraries (cards and devices) can be built concurrently after primitives. Core components and asset primitives can be built concurrently after core HDL libraries are built
- Follow this sequence to build HDL libraries in the assets project
- Once projects components are built, it is recommended that a top-level project build is executed to ensure all build artifacts and exported properly for dependent projects
- Assemblies may be built concurrently

The following examples demonstrate executing builds using the IDE:

1. Select the platforms for the build. In the example, RCC platform centos7 and HDL platform xsim are Selected

Figure 22: Importing OpenCPI Core and Assets

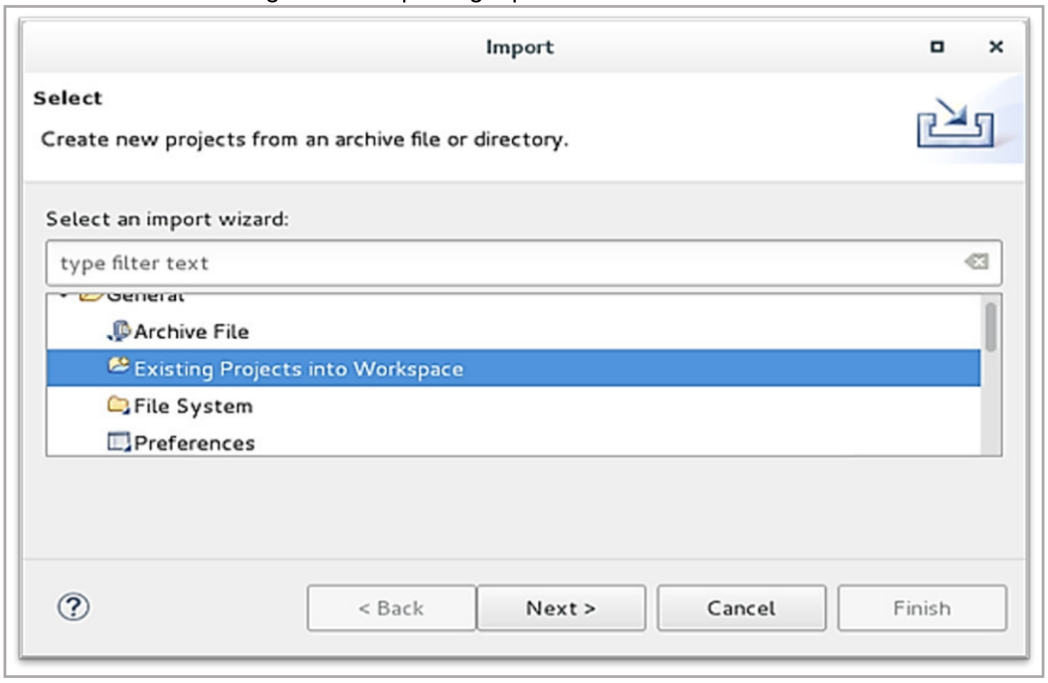
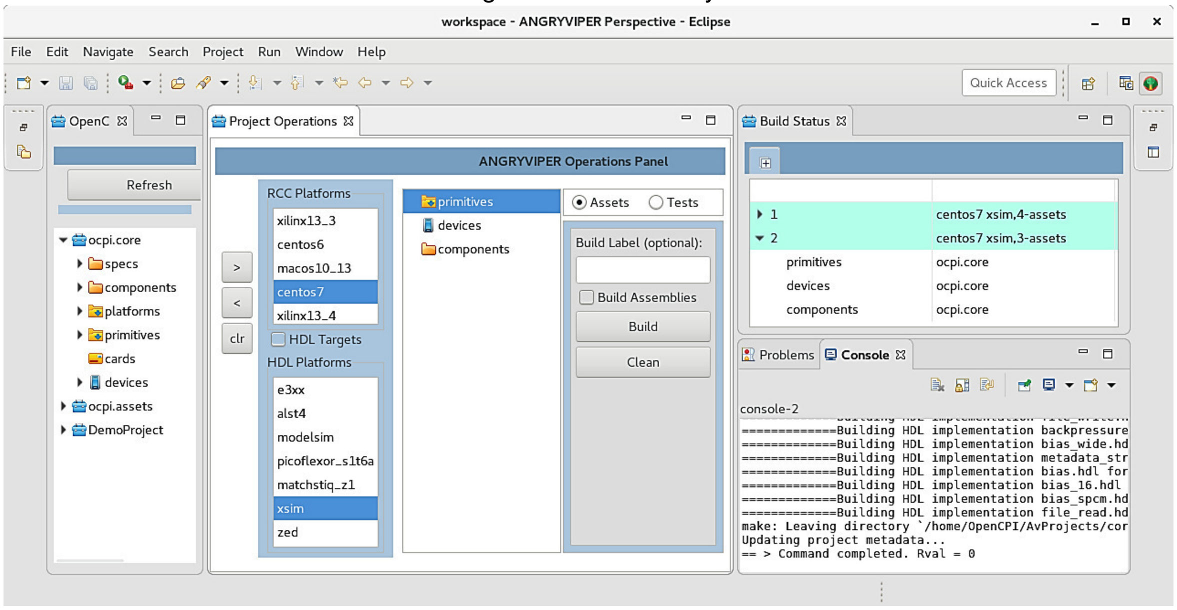


Figure 23: Build Entry



1	centos7 xsim,4-assets
2	centos7 xsim,3-assets
primitives	ocpi.core
devices	ocpi.core
components	ocpi.core

```
=============Building HDL implementation backpressure
=============Building HDL implementation bias_wide.hd
=============Building HDL implementation metadata_str
=============Building HDL implementation bias.hdl for
=============Building HDL implementation bias_16.hdl
=============Building HDL implementation bias_spcm.hd
=============Building HDL implementation file_read.hd
make: Leaving directory `/home/OpenCPI/AvProjects/cor
Updating project metadata...
== > Command completed. Rval = 0
```

2. Expand the ocpi. core project in the OpenCPI Projects View, select Primitives, right click, select Build. The build status entry appears and turns green when the build successfully completes.

3. Select Devices (the core projects have no cards currently), right click, select Build

4. Similarly start builds for the core projects components and the assets project primitives

5. Complete a top-level build for the core project. In this example, the core project is built from the Operations panel.

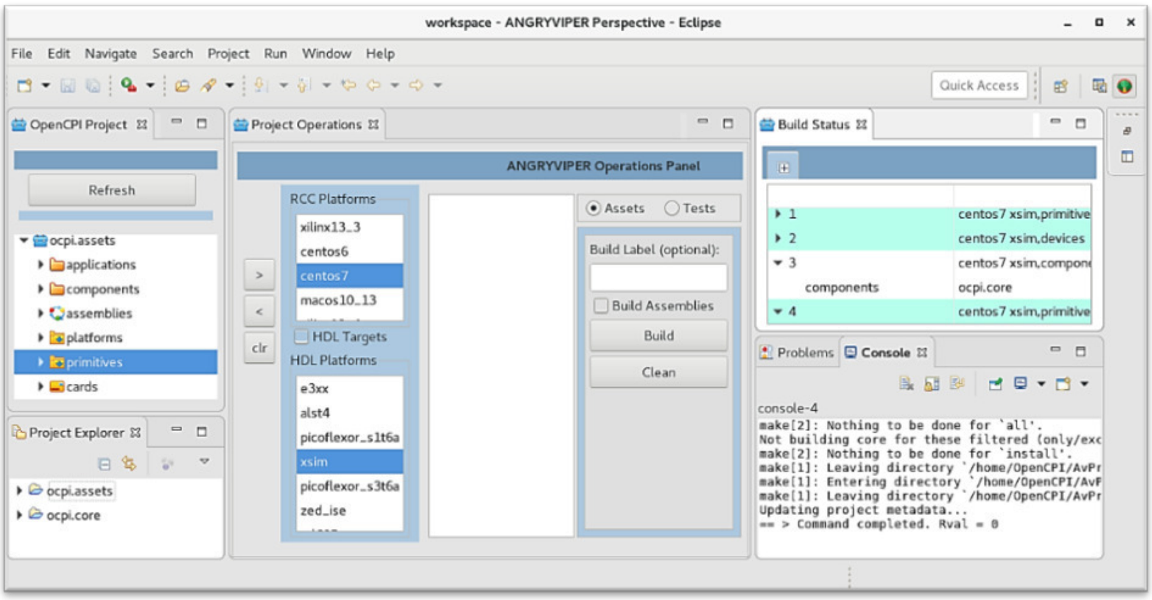
6. Complete building the assets project following a similar process.

Creating New Projects and Libraries

To create a new project:

- Place the cursor in the OpenCPI Projects panel

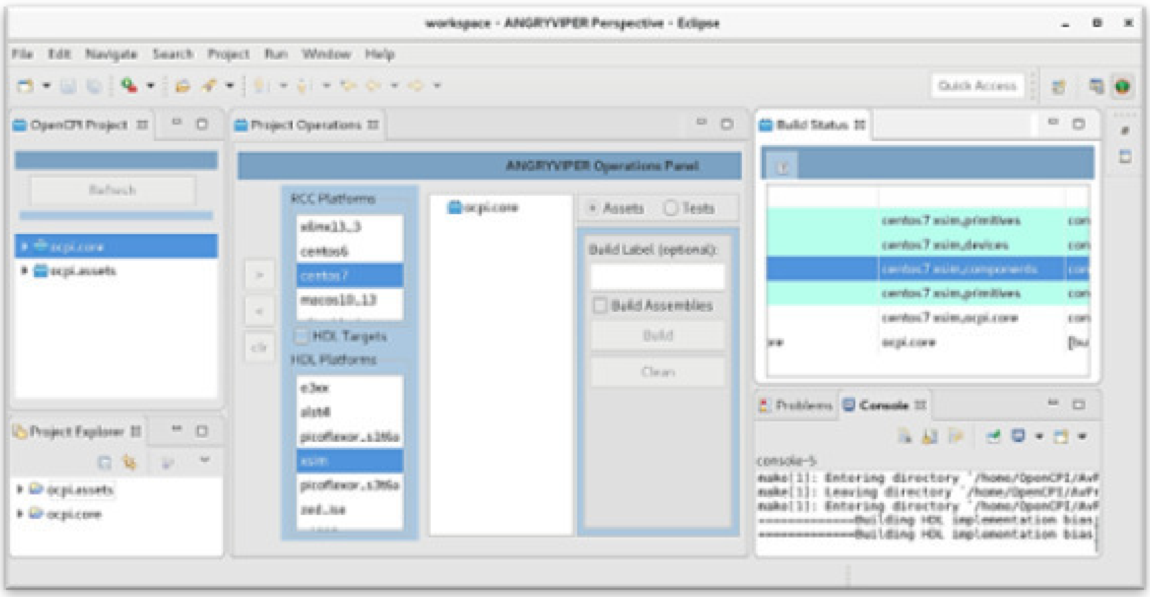
Figure 24: Project Built from Operations panel



▸ 1	centos7 xsim,primitive
▸ 2	centos7 xsim,devices
▾ 3	centos7 xsim,compon
components	ocpi.core
▾ 4	centos7 xsim,primitive

console-4
```
make[2]: Nothing to be done for `all'.
Not building core for these filtered (only/exc
make[2]: Nothing to be done for `install'.
make[1]: Leaving directory `/home/OpenCPI/AvPr
make[1]: Entering directory `/home/OpenCPI/AvP
make[1]: Leaving directory `/home/OpenCPI/AvPr
Updating project metadata...
== > Command completed. Rval = 0
```

Figure 25: Core Project Built from Ops panel



	centos7 xsim,primitives	con
	centos7 xsim,devices	con
	centos7 xsim,components	con
	centos7 xsim,primitives	con
	centos7 xsim,ocpi.core	con
re	ocpi.core	[bu

console-5
```
make[1]: Entering directory `/home/OpenCPI/AvP
make[1]: Leaving directory `/home/OpenCPI/AvPr
make[1]: Entering directory `/home/OpenCPI/AvP
================Building HDL implementation bias
================Building HDL implementation bias
```

- Right click
- Select-Asset Wizard
- Select- Asset Type Project from the drop-down
- Click Finish

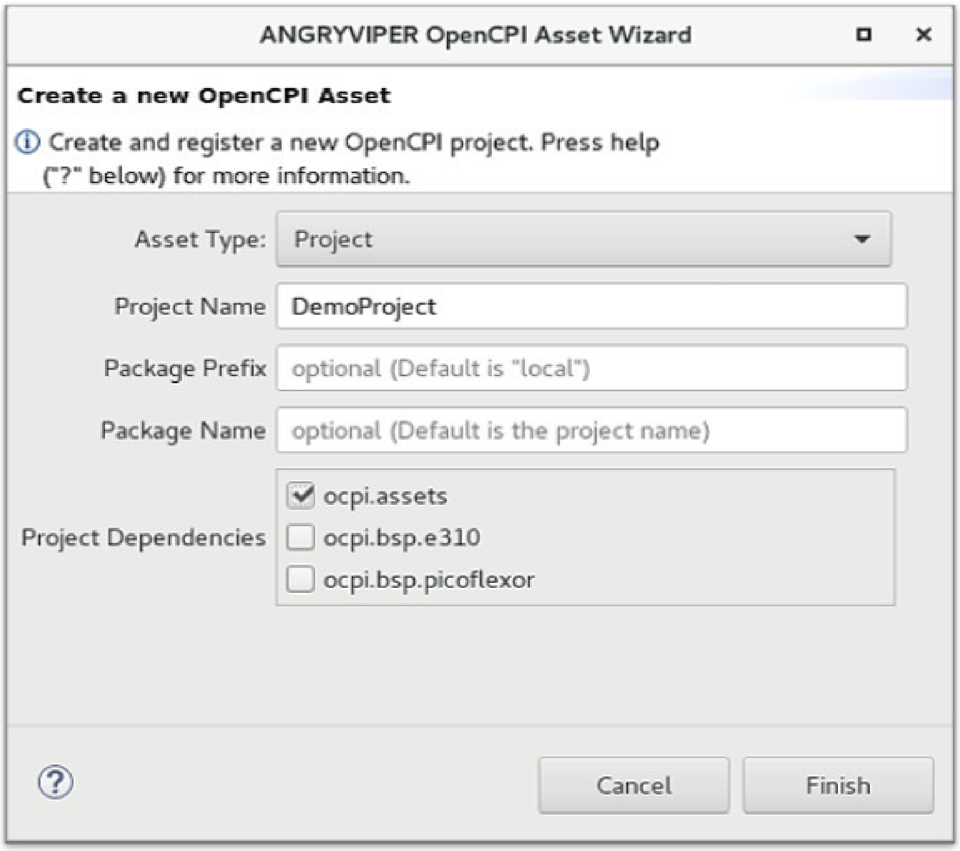
Note: All registered OpenCPI projects have the core project dependency by default.

When this completes, a new project (DemoProject) will be created, registered and is displayed in the OpenCPI Projects and the Eclipse Project Explorer views. It will have the assets project as a dependency, so it can use the components contained in it.

Next, a components library is added to DemoProject

- Select the project in the OpenCPI Projects View
- Right Click, Open the Assets Wizard

Figure 26: Create a New Project


ANGRYVIPER OpenCPI Asset Wizard

Create a new OpenCPI Asset

Create and register a new OpenCPI project. Press help ("?" below) for more information.

Asset Type: Project

Project Name: DemoProject

Package Prefix: optional (Default is "local")

Package Name: optional (Default is the project name)

Project Dependencies:
- [✓] ocpi.assets
- [ ] ocpi.bsp.e310
- [ ] ocpi.bsp.picoflexor

- Select- Asset Type Library

The wizard will create the components library by default. If multiple libraries are anticipated, then this library should be named accordingly, and it will be placed in a top-level components folder. This example uses one components library and when complete, select Finish.
The Components Library is now created and appears in the Project Explorers as shown below:

More on OpenCPI Projects

In OpenCPI, package-ids are assigned to projects and other shared assets in the project, such as its libraries. The project wizard provides inputs for the project package-id: package-prefix and package-name. The project package-id is *package-prefix.package-name*. This becomes a unique identifier to the project and its shared assets. The Project Wizard screen goes more into detail.

In the new project example above, no inputs were provided for package-prefix and package-name, the defaults are used and the assigned package-id is: local. DemoProject. This can be seen in the Project Registry.
To view project registry in a terminal window, **use the command: ocpidev show registry** and the result is shown below:

Project Registration

By registering a project, a user is publishing his/her project so that it can be referenced/searched by any user or project using that same project registry. The default project registry is found at : **/opt/opencpi/project-registry**. The registration features are provided to support bringing in an external OpenCPI project or a project location needs to change. Examples:

- To rename a project: unregister it, and change its name with Project Explorer, refresh OpenCPI Projects, re-register it.
- To move a project: unregister it then delete it from the workspace (use Eclipse Project Explorer, do not delete it from the file system). Move it, import it back into the workspace, then refresh OpenCPI Projects. Now re-register it.

Figure 27: Asset Wizard

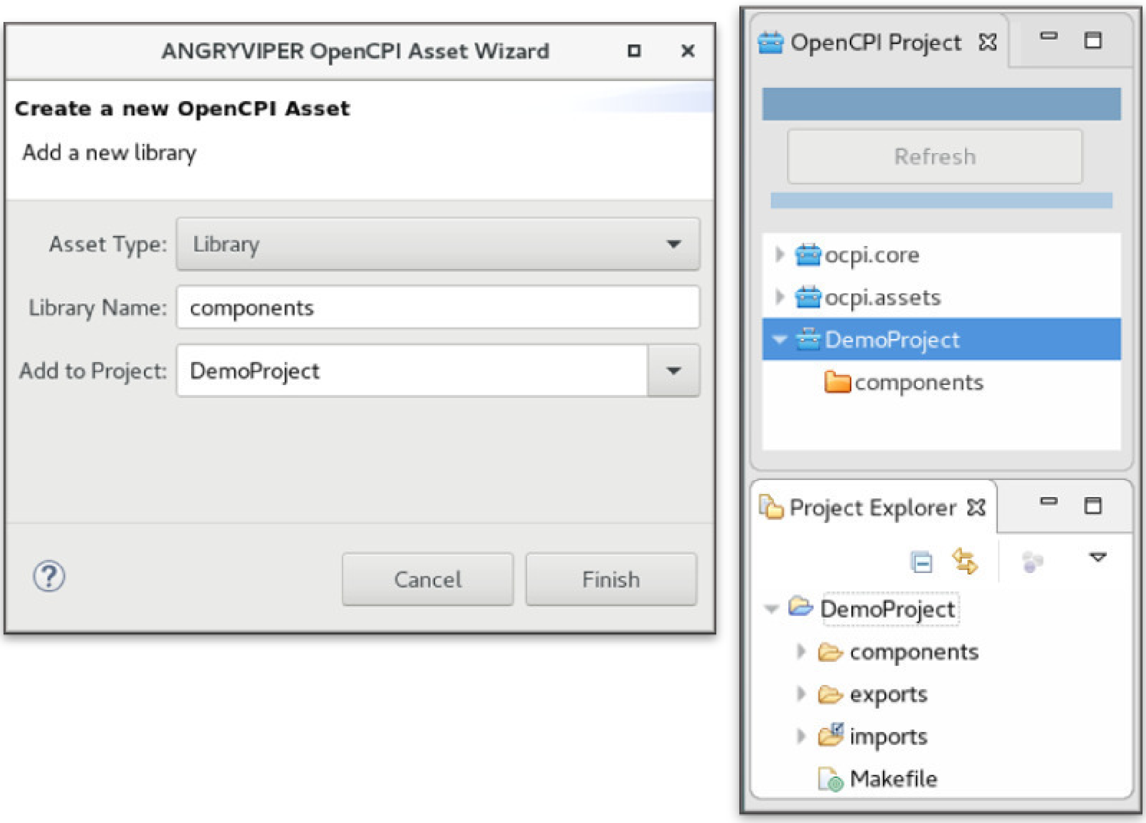
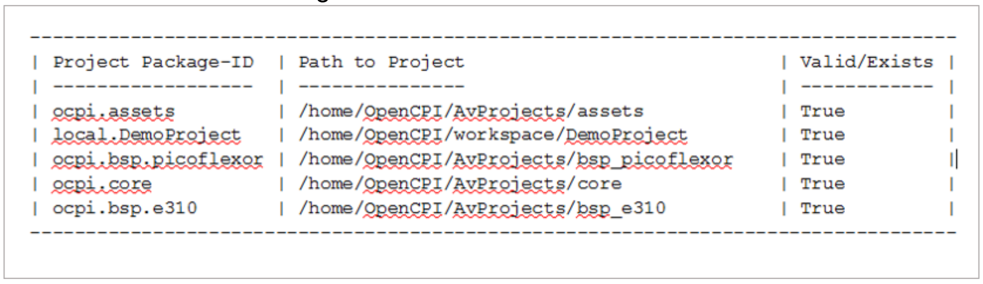


Figure 28: Terminal Window View



```
 ---------------------------------------------------------------------------------
 | Project Package-ID  | Path to Project                                | Valid/Exists |
 | ------------------  | ---------------                                | ------------ |
 | ocpi.assets         | /home/OpenCPI/AvProjects/assets                | True         |
 | local.DemoProject   | /home/OpenCPI/workspace/DemoProject            | True         |
 | ocpi.bsp.picoflexor | /home/OpenCPI/AvProjects/bsp_picoflexor        | True         |
 | ocpi.core           | /home/OpenCPI/AvProjects/core                  | True         |
 | ocpi.bsp.e310       | /home/OpenCPI/AvProjects/bsp_e310              | True         |
 ---------------------------------------------------------------------------------
```

Errors in Creating New Project

If an error occurs, the wizard will present a dialog panel explaining the problem. Errors will occur when the framework cannot support the asset creation or there are file system issues.

Click OK to close the dialog. If the framework fails to create an asset it automatically cleans remnant artifacts off the file system.

OpenCPI Libraries

An OpenCPI project can have one or more libraries. The typical guidelines are as follows:

- A project anticipated to have a single component library, name the library components. All component and worker assets will reside in the top-level components directory. When using the OpenCPI asset wizard to create a library the default name provided is components
- A Project anticipated to have multiple libraries, give these libraries a name other than components. These libraries will be placed in top-level components as sub-directories named after the library. The user will be given the option to create component specifications and workers in these sub-directory libraries
- Once an option is Selected it cannot be changed without a lot manipulation

Creating Components, Protocols and Workers

Some fundamentals regarding creating components, protocols, and workers:

- Protocols specify expected data for component ports
- Protocols and components can exist in in either the top-level specs folder of the project or in a library specs folder
- The top-level specs folder is intended to hold more global protocols and components that can be reused in library component specifications
- Workers may only reside in a library

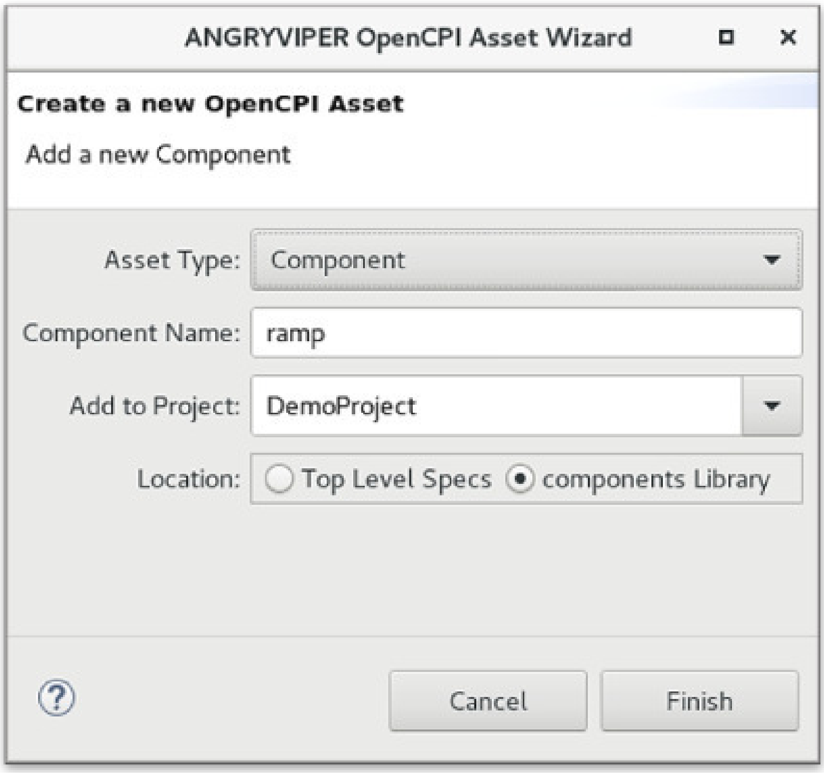
The IDE's Asset Wizard provides an easy way to create this class of assets and locate them in their proper place. Protocols and components simply need a name and where to put them. Workers need a name, component specification, and the implementation language.

Component Example from the Getting Started Guide

In this example, the ramp component is created using the IDE.

- Open the Asset Wizard
- OpenCPI Projects
- Select DemoProject/Components and right-click
- Select Component from the context menu
- Fill in the name as shown below

Figure 29: Create a New OpenCPI Asset



ANGRYVIPER OpenCPI Asset Wizard

Create a new OpenCPI Asset

Add a new Component

Asset Type: Component

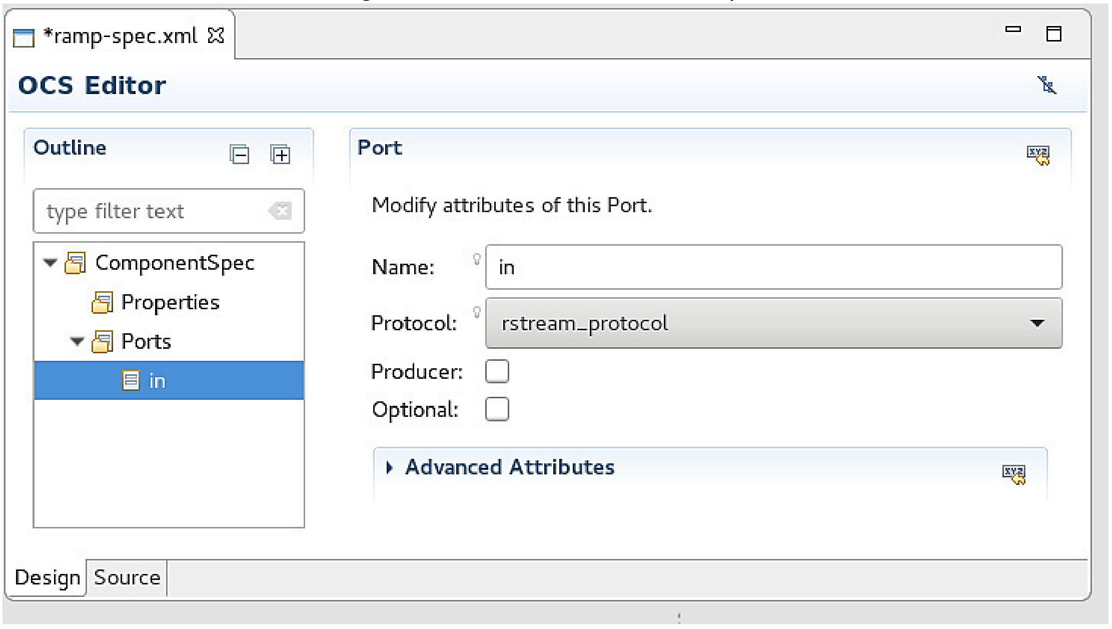
Component Name: ramp

Add to Project: DemoProject

Location: ◯ Top Level Specs ● components Library

- Once the OCS Editor opens, select Ports in the outline
- Click the Add a Port link and the Port Form appears and name the Port **in**
- Select **restream protocol** from the drop-down menu
- Repeat above and add the **out** port
- The figure below shows the Port form for the ramp **in**

Figure 30: Port Form for the Ramp In


*ramp-spec.xml

OCS Editor

Outline

type filter text

- ComponentSpec
  - Properties
  - Ports
    - in

Port

Modify attributes of this Port.

Name:	in
Protocol:	rstream_protocol
Producer:	☐
Optional:	☐

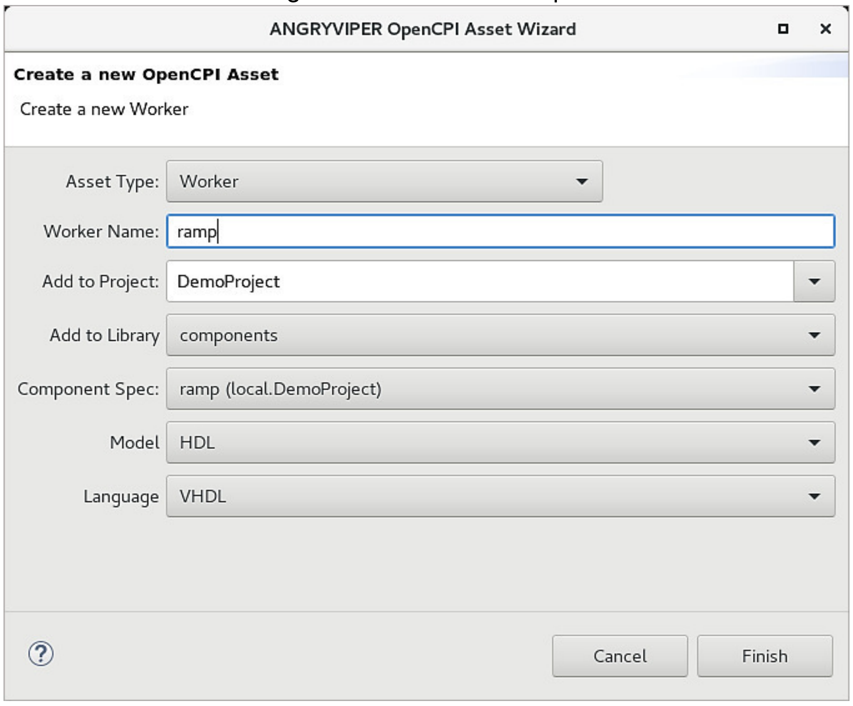
Advanced Attributes

Design | Source

Worker Example from the Getting Started Guide

- Open the Asset Wizard
- OpenCPI Projects, Select **ramp-spec.xml** located in DemoProject/Components/specs
- Right-click, Select Worker from the context menu, fill in the name
- Select the VHDL language as shown below
- When the HDL OWD Editor opens, verify the spec and language entries
- Save both files (Eclipse floppy disks icon in the toolbar at the top)

Figure 31: Worker Example


ANGRYVIPER OpenCPI Asset Wizard

Create a new OpenCPI Asset

Create a new Worker

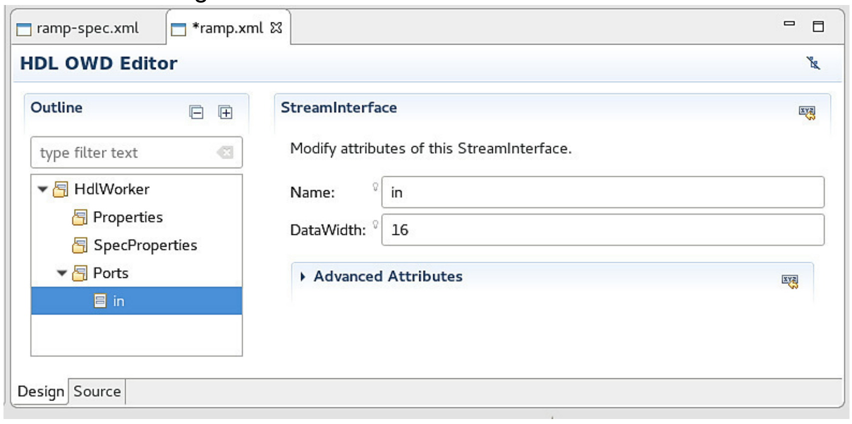
Asset Type:	Worker
Worker Name:	ramp
Add to Project:	DemoProject
Add to Library	components
Component Spec:	ramp (local.DemoProject)
Model	HDL
Language	VHDL

Cancel | Finish

The next step is to add the **in and out** stream interfaces:

- Select Ports in the HDL OWD Editor outline
- Click the Add a StreamInterface link and the StreamInterface form will display
- Set the Port name and data width as shown in the figure below
- As above, create the **out** Port - the ramp worker is now complete

Figure 32: Addition of In and Out Interfaces



Modify attributes of this StreamInterface.

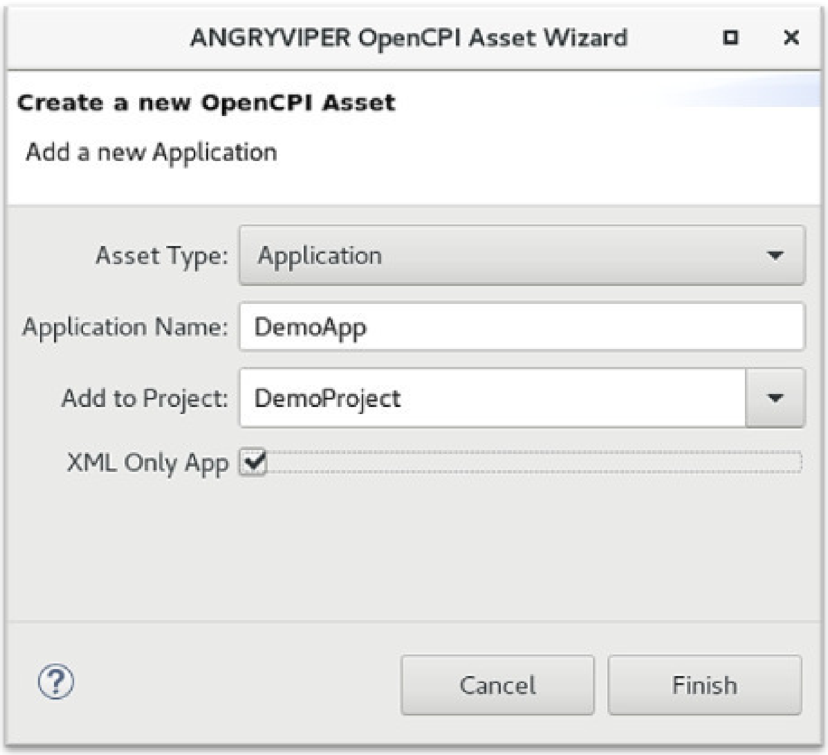
Name: in

DataWidth: 16

Creating Applications and Assemblies

The AV IDE Application and Assembly Editors provides a drag-and-drop interface to generate the respective XML files. As an example, the Application Editor is used to create the DemoApp application described in the Getting Started Guide.To start creating, use the Asset Wizard to create a new application:

Figure 33: Asset Wizard New Application



Create a new OpenCPI Asset

Add a new Application

Asset Type: Application

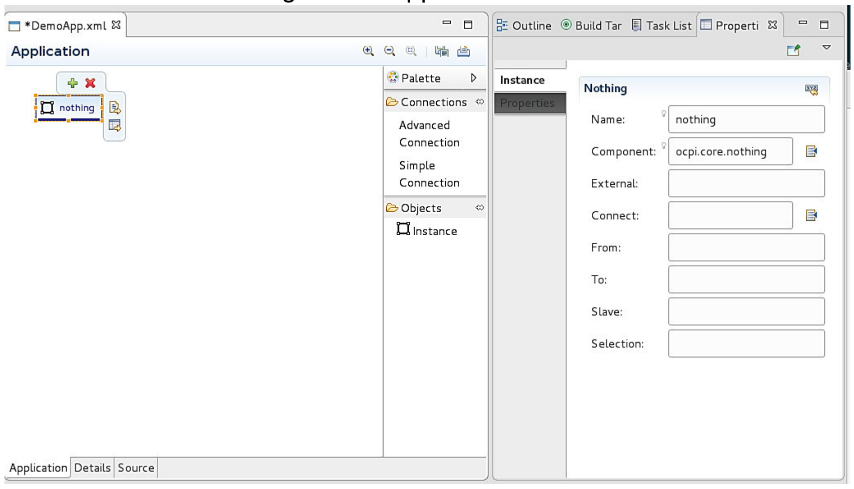
Application Name: DemoApp

Add to Project: DemoProject

XML Only App ✓

Click Finish and the Application Editor opens:

Figure 34: Application Editor



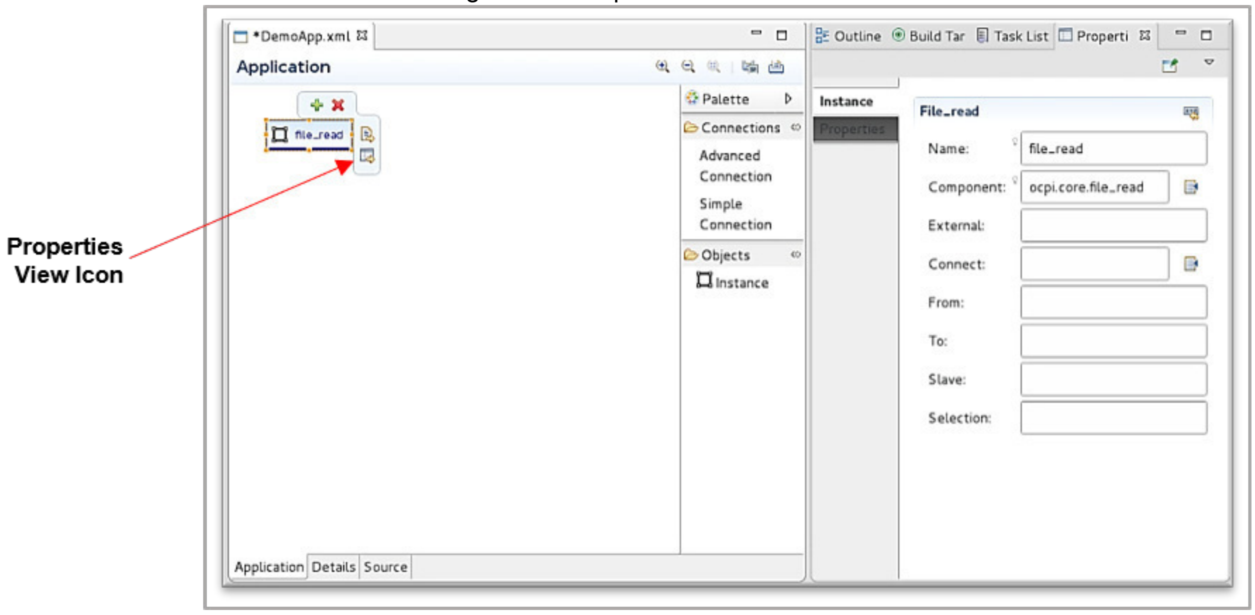
Nothing

Name:	nothing
Component:	ocpi.core.nothing
External:	
Connect:	
From:	
To:	
Slave:	
Selection:	

Adding Components to an Application

Applications are constructed using OpenCPI Component Specifications (OCS) referred to as components. When the Editor opens, the **Application Tab** is presented with the **nothing** instance in the panel.
To select a OCS for this instance, click the ***nothing*** instance and more controls appear.

Figure 35: Properties View



File_read

Name:	file_read
Component:	ocpi.core.file_read
External:	
Connect:	
From:	
To:	
Slave:	
Selection:	

Select the bottom right icon to see the Properties View Form to further populate the instance. To find the desired OCS, go to the Properties View and click on the List icon next to the component textbox.

A listing of all component specifications found in the environment is displayed. The **ocpi.core.file_read** component was selected and now the display appears as shown above.

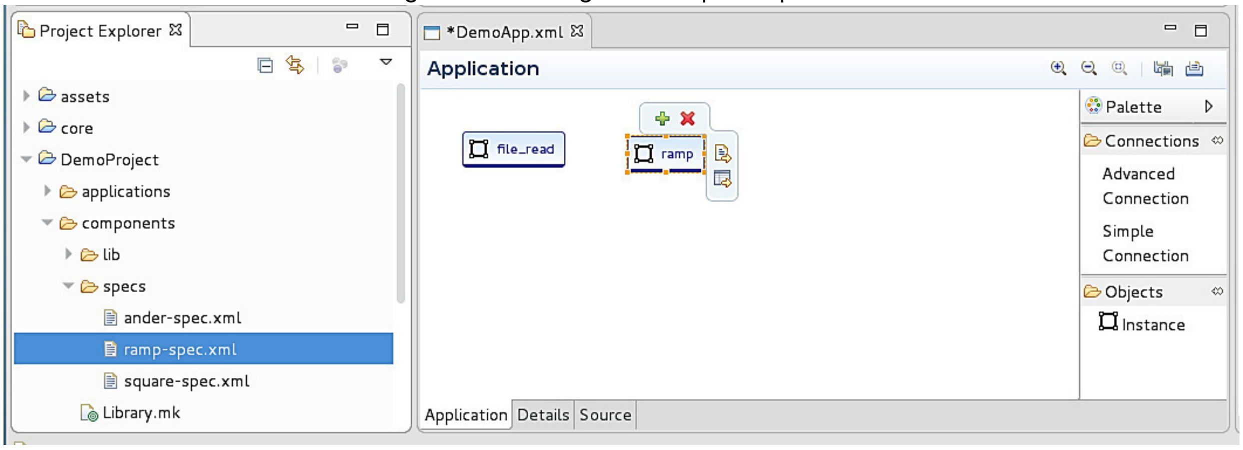
The next example adds the local DemoProject.ramp component to the application using drag-and-drop:

- Use the Eclipse Project Explorer to Select a component OCS XML file
- Left click/hold and drag the file to the Application Panel
- Release the mouse button

  As shown below, the ramp component is now in the application. Another method to do this is:

- Drag-and-drop the instance object from the Palette into the application panel
- Select the component via the Properties View as demonstrated in the example above.

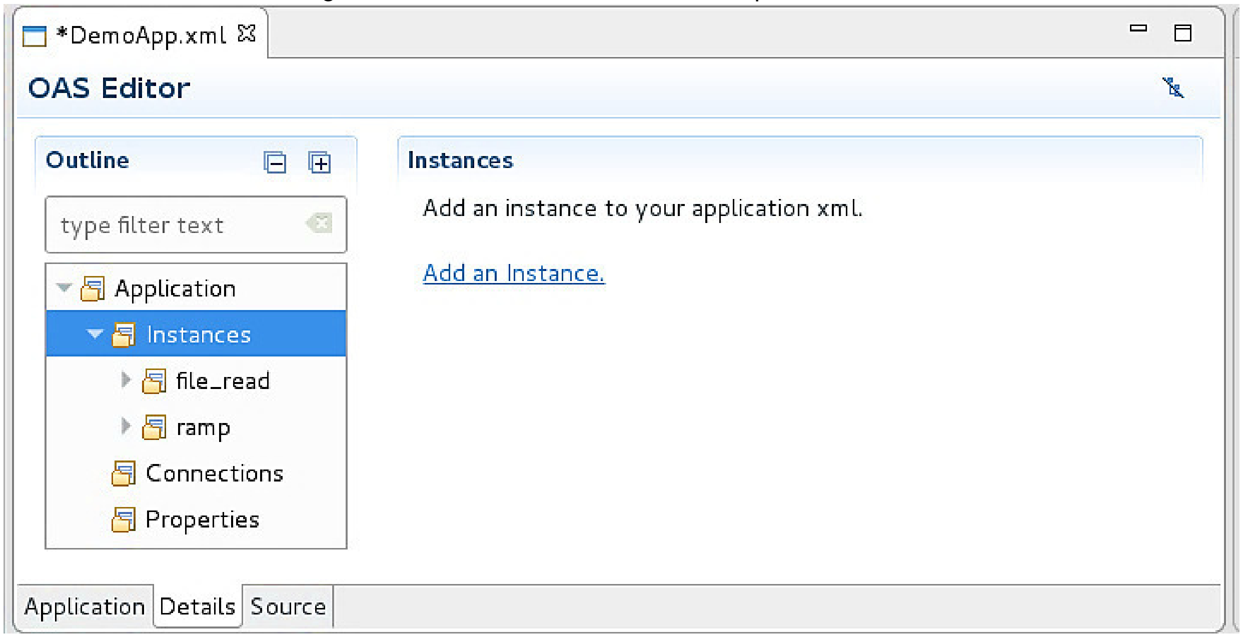
Figure 36: Adding the Ramp Component



The third method to construct an application is to use the Details Tab. This tab presents a form-based editor similar to the component and worker editors described above. The Application Details panel is shown in the figure below:

- Select the Instances element
- Click Add an Instance

Figure 37: Third Method to Add a Component Instance



Add an instance to your application xml.

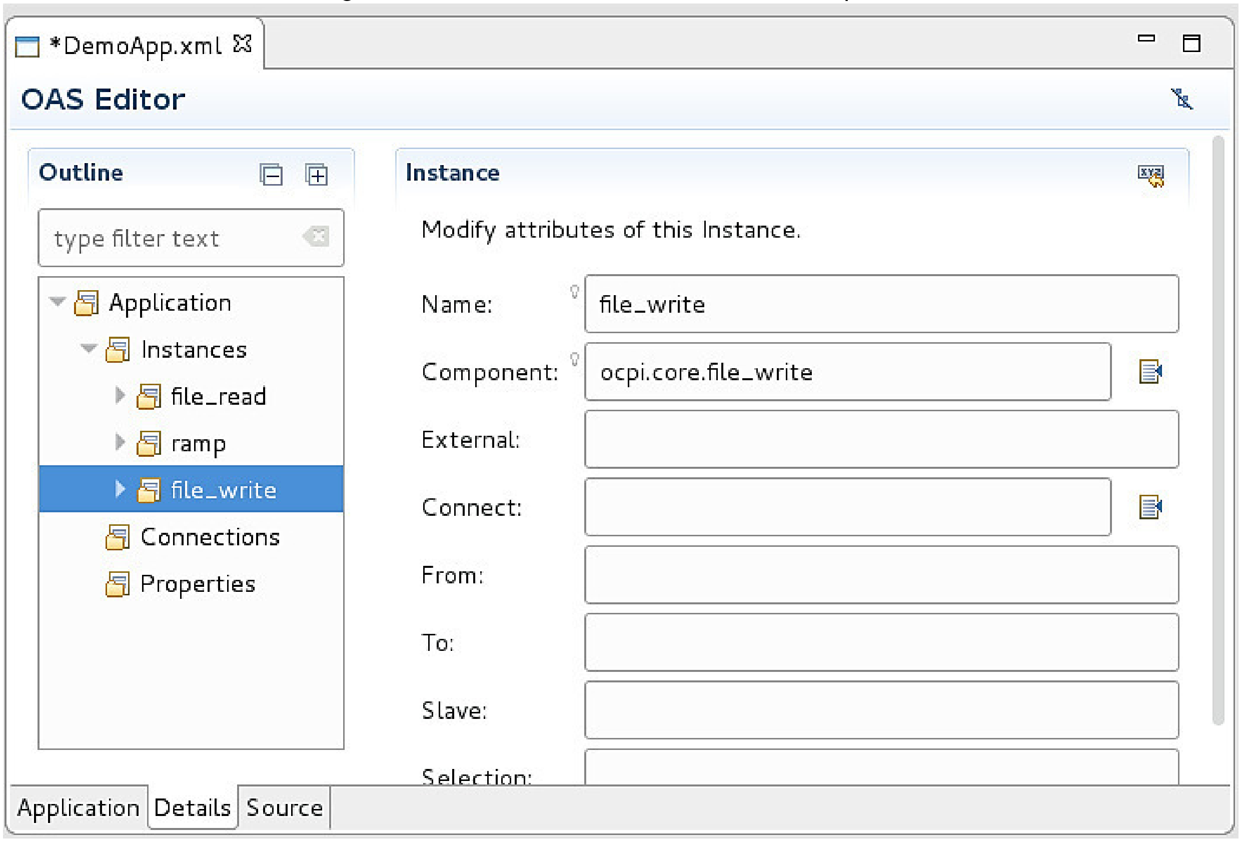
A form opens to populate the new instance as shown in the below image. Note that component choices originate from the current project and its project dependencies.

Click the List icon next to the Component text box input to the components list. The file write component was selected. The below figure shows the current design view of the application.

Connecting Application Components

The Application Editor provides three different methods to create connections between components. If simple connections are sufficient (single in/out components), the simpliest method is to use the application diagram (Application tab) and use the Simple Connection in the palette:

Figure 38: Select File Write as the New Companion


*DemoApp.xml

OAS Editor

Outline

type filter text

- Application
  - Instances
    - file_read
    - ramp
    - file_write
  - Connections
  - Properties

Instance

Modify attributes of this Instance.

Name: file_write

Component: ocpi.core.file_write

External:

Connect:

From:

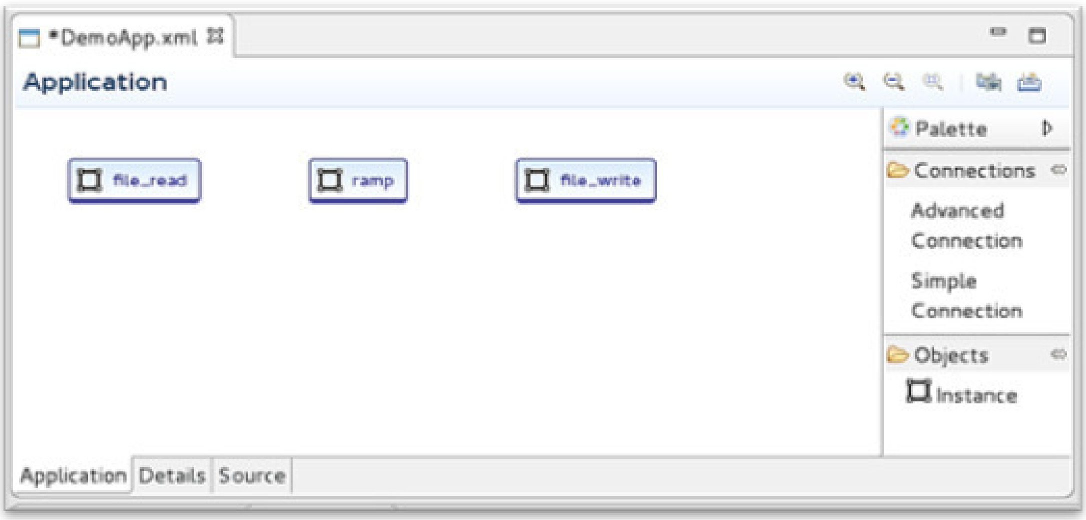
To:

Slave:

Selection:

Application | Details | Source

Figure 39: Design View of the Application



- Select simple connection in the palette
- Move the cursor to the source component
- Click while on it and then move the cursor to the receiving component, click it to complete a connection

Complex connection can be added using the palette and is the easiest way as well.

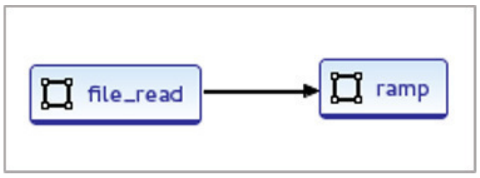
The second method is to use the XML node approach accessed using the Details tab:

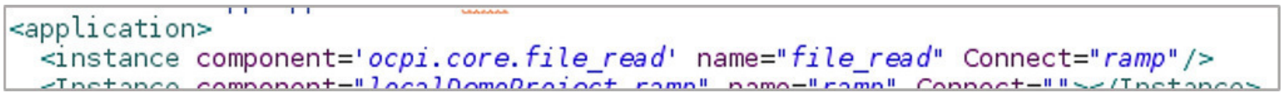
- Simple connections are added by selecting the instance then filling in the ”Connect” input (this indicates the component receiving the output of this instance)
- Complex connection are not as graceful to accomplish using this method, Click the Connections node in the outline

- Click the Add a Connection link
- Click add Port, Provide the Port name and instance, it connects as shown in the Advanced Connection Figure below.

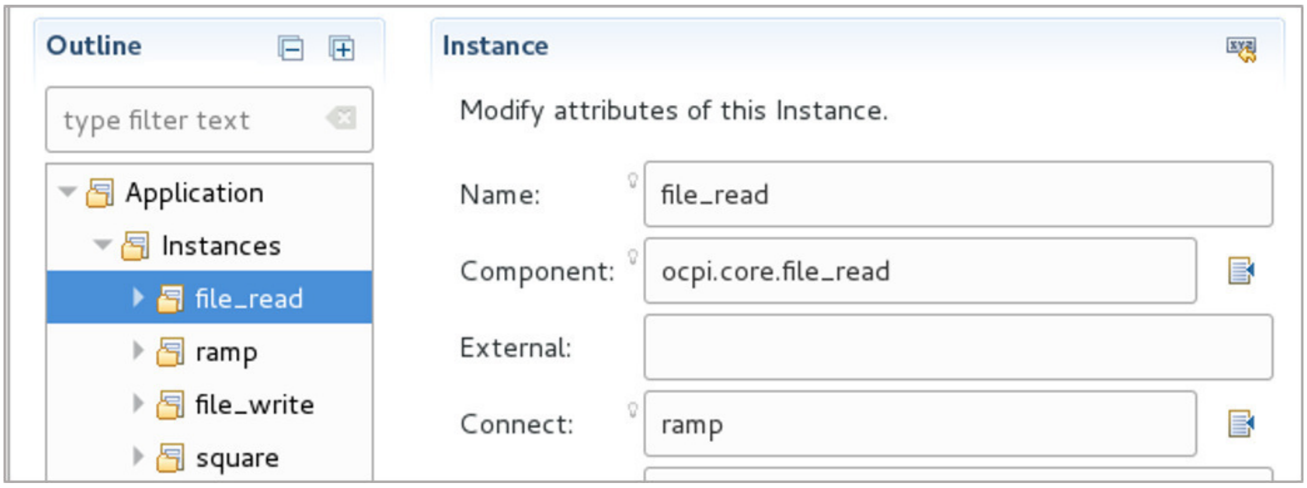
The third method is to edit the XML source directly. Once the template of a complex connection is put in place, this is likely the easiest method to add complex connections. It is also a good method used to make corrections to the diagram, particularly if instances are removed from the diagram or the component of an instance is changed. The following figures demonstrate a simple connection:
Advanced connections are easily created using the Application Tab. To make an advanced connection while in

Figure 40: Connection Example





```
<application>
  <instance component='ocpi.core.file_read' name="file_read" Connect="ramp"/>
  <Instance component="localDemoProject.ramp" name="ramp" Connect=""></Instance>
```



Modify attributes of this Instance.

Name: file_read
Component: ocpi.core.file_read
External:
Connect: ramp

the Application Tab:

- Select Advanced Connection in the palette
- Move the cursor to the source component
- Click it, move the cursor to the destination component and click it again

A pop-up form panel appears with inputs to complete the connection as shown below. Advanced connections can be named (optional). The Ports section opens with default Port names **out and in**. Click on the In-Port name and update it to the correct Port (in1) as shown above.

The following series of figures show the advanced connection in the 3 editor views:
See the Application Development Guide for detailed information about the Application and Assembly XML files. Note that the Application and Assembly Editors provide a basic capability to create the respective XML files.

Assembly Editor

The HDL Assembly Editor operates similarly to the Application Editor. The difference is that assemblies are built with application workers while applications are built with components. The drag-and-drop feature works with OpenCPI Worker Description (OWD) XML files just as the application Editor operates with OCS XML files.
Note: The Eclipse Project Explorer must be used for the current drag and drop feature; it is currently not supported in the OpenCPI Projects View.
Below shows the square.hdl worker added to an assembly via drag-and-drop. Also note that worker choices originate from registered OpenCPI projects.

Figure 41: Pop Up Panel

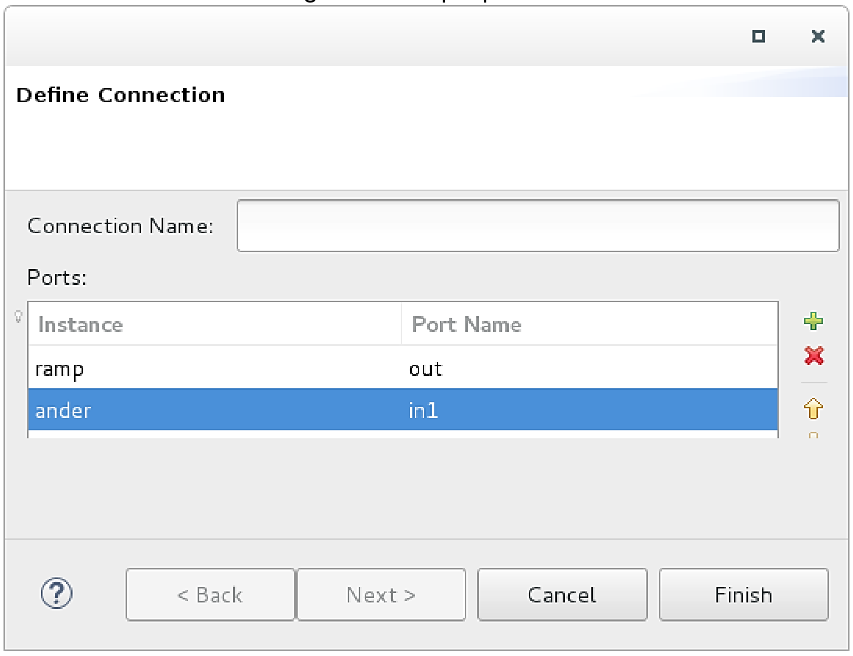
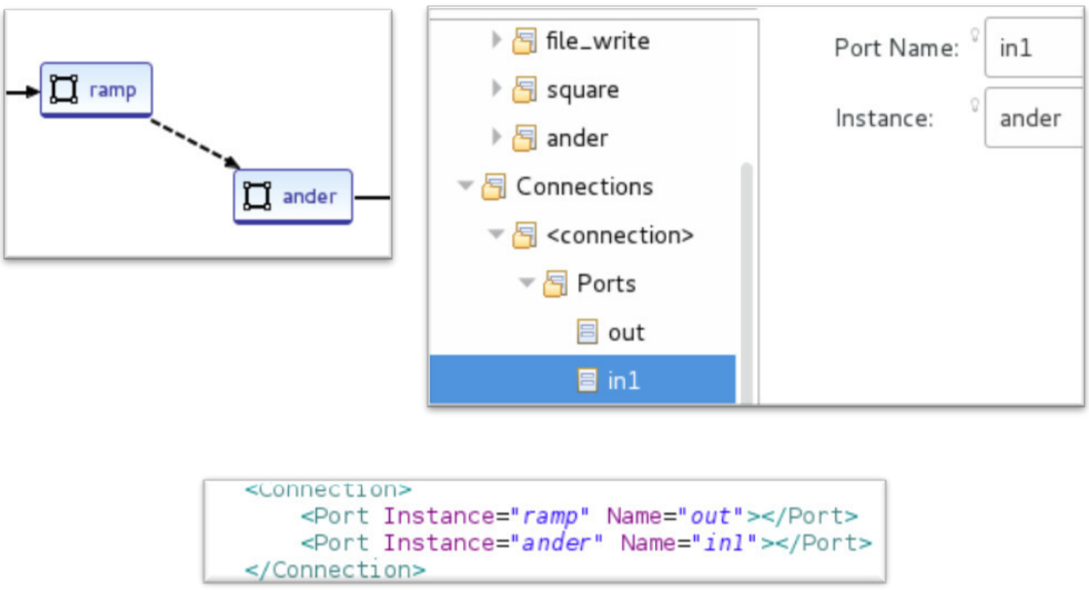
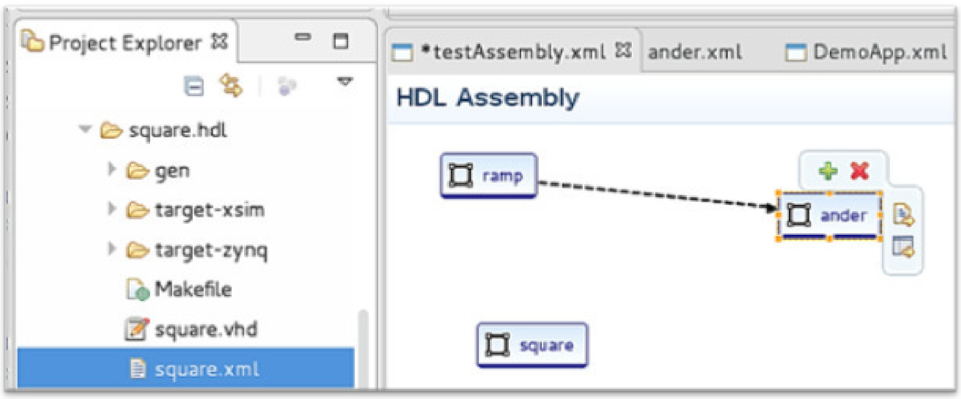


Figure 42: Advanced Connection Views



```
<Connection>
    <Port Instance="ramp" Name="out"></Port>
    <Port Instance="ander" Name="in1"></Port>
</Connection>
```

Figure 43: Square HDL Worker

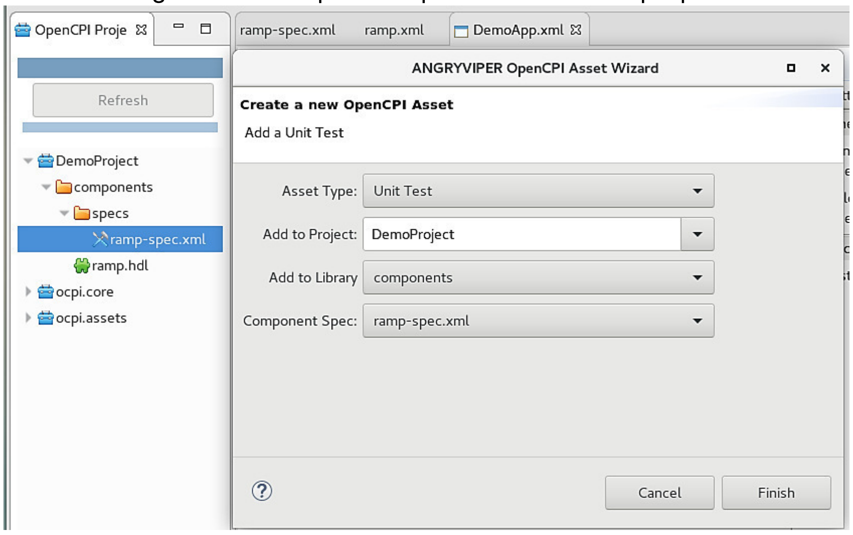


Creating Component Unit Tests

Creating a Unit Test using the Asset Wizard is demonstrated in this section.

- Using the OpenCPI Projects View
- Navigate to DemoProject
- Components
- Specs
- Select Ramp-Spec.XML
- Right-click and select **New Unit Test** from the context menu.
- The Asset Wizard opens with all inputs preset for the ramp-spec selection as shown below and then Click Finish.

Figure 44: All Inputs are preset for the Ramp Spec



ANGRYVIPER OpenCPI Asset Wizard

Create a new OpenCPI Asset

Add a Unit Test

Asset Type: Unit Test

Add to Project: DemoProject

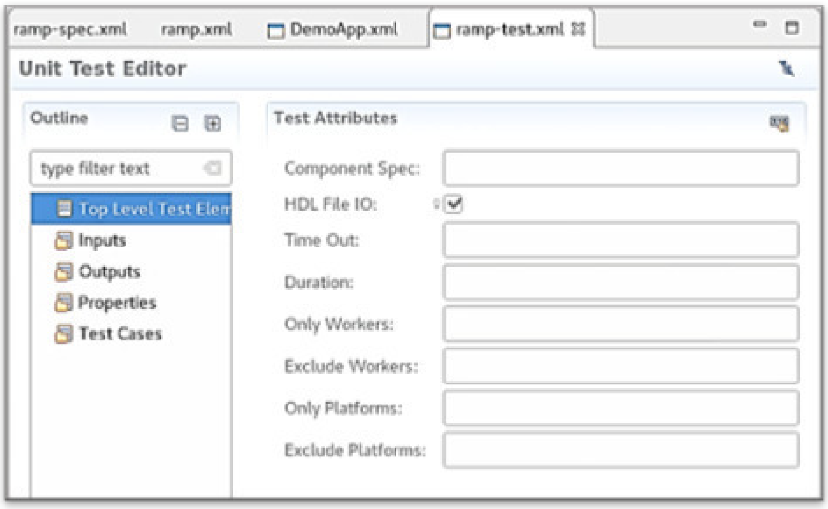
Add to Library: components

Component Spec: ramp-spec.xml

When the action completes, the Unit Test Editor opens in the Editor panel as shown below.

Unit testing requires an in-depth knowledge on how to implement them so this will not be addressed here. Refer to the Component Development Guide to learn about them. The Editor supports most aspects of developing unit test XML.

Figure 45: Unit Test Editor Panel



Unit Test Editor

Test Attributes

Component Spec:

HDL File IO: ✓

Time Out:

Duration:

Only Workers:

Exclude Workers:

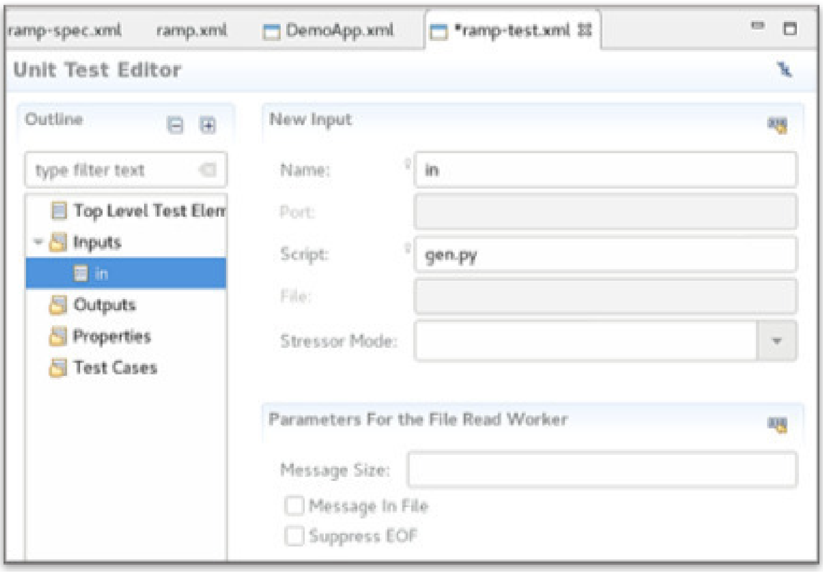
Only Platforms:

Exclude Platforms:

Note: There are a number of cases where one attribute out of a set may go in the test XML element. This is very prevalent for the input, output, and test case elements. In this implementation of the Editor, once an attribute out of the set is added, the inputs for other choices are disabled.

The figure below demonstrates this behavior. An input element can either have a name or a Port and the source data from them can either be a script or a file, so once filled in the other inputs are disabled. Simply clear the input to start over (use backspace or select delete).

Figure 46: Display of Attribute Choice



Name: in

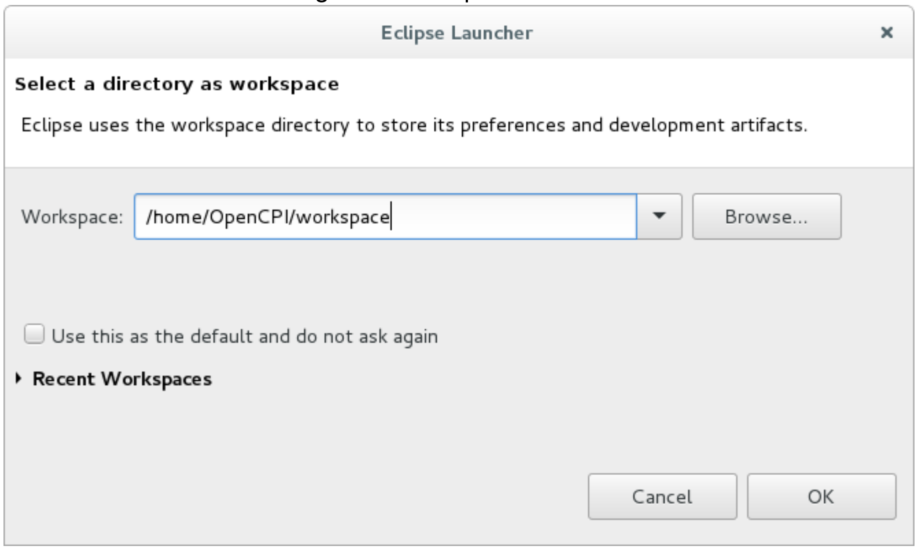
Script: gen.py

Appendices

A. Eclipse Basics

A central abstraction for Eclipse is **the workspace**. This is a directory Eclipse uses to maintain state and projects. When Eclipse opens, it prompts the user to Select a workspace with the following dialog preset for a default directory name in the user's home directory. If the workspace directory does not exist yet, it will be created. This directory path name can be changed. The Browse Button allows navigation to another location to either create the workspace or select an existing workspace directory.

Figure 47: Eclipse Launcher


Eclipse Launcher

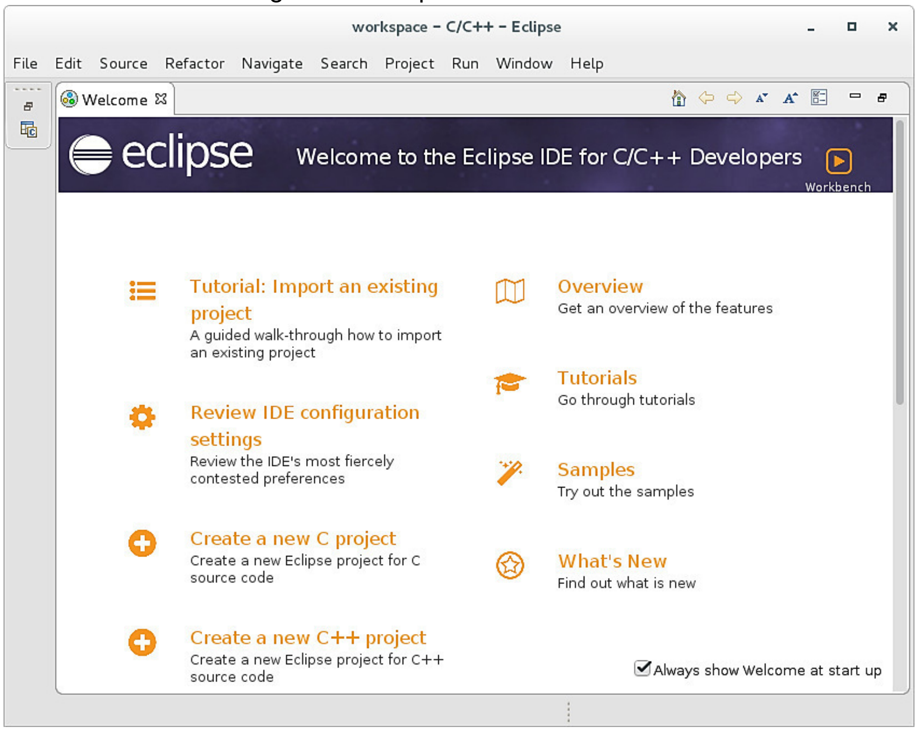
Select a directory as workspace

Eclipse uses the workspace directory to store its preferences and development artifacts.

Workspace: /home/OpenCPI/workspace

Since this is a new workspace, when Eclipse completes start-up, it presents the Welcome screen as shown in the figure below. If you are new to Eclipse or an IDE of this type, this is a good screen to explore. Overview gives basic concepts and definitions such as Toolbars, Perspectives, and Views. The other selections provide how-to

Figure 48: Eclipse Welcome Screen


workspace - C/C++ - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help
Welcome
eclipse
Workbench

Welcome to the Eclipse IDE for C/C++ Developers

Tutorial: Import an existing project
A guided walk-through how to import an existing project

Review IDE configuration settings
Review the IDE's most fiercely contested preferences

Create a new C project
Create a new Eclipse project for C source code

Create a new C++ project
Create a new Eclipse project for C++ source code

Overview
Get an overview of the features

Tutorials
Go through tutorials

Samples
Try out the samples

What's New
Find out what is new

✓ Always show Welcome at start up

instruction. To continue to see this screen when Eclipse opens, keep the "Always Show Welcome on Startup" check box checked and go through these items.

For now, move on to the workbench:

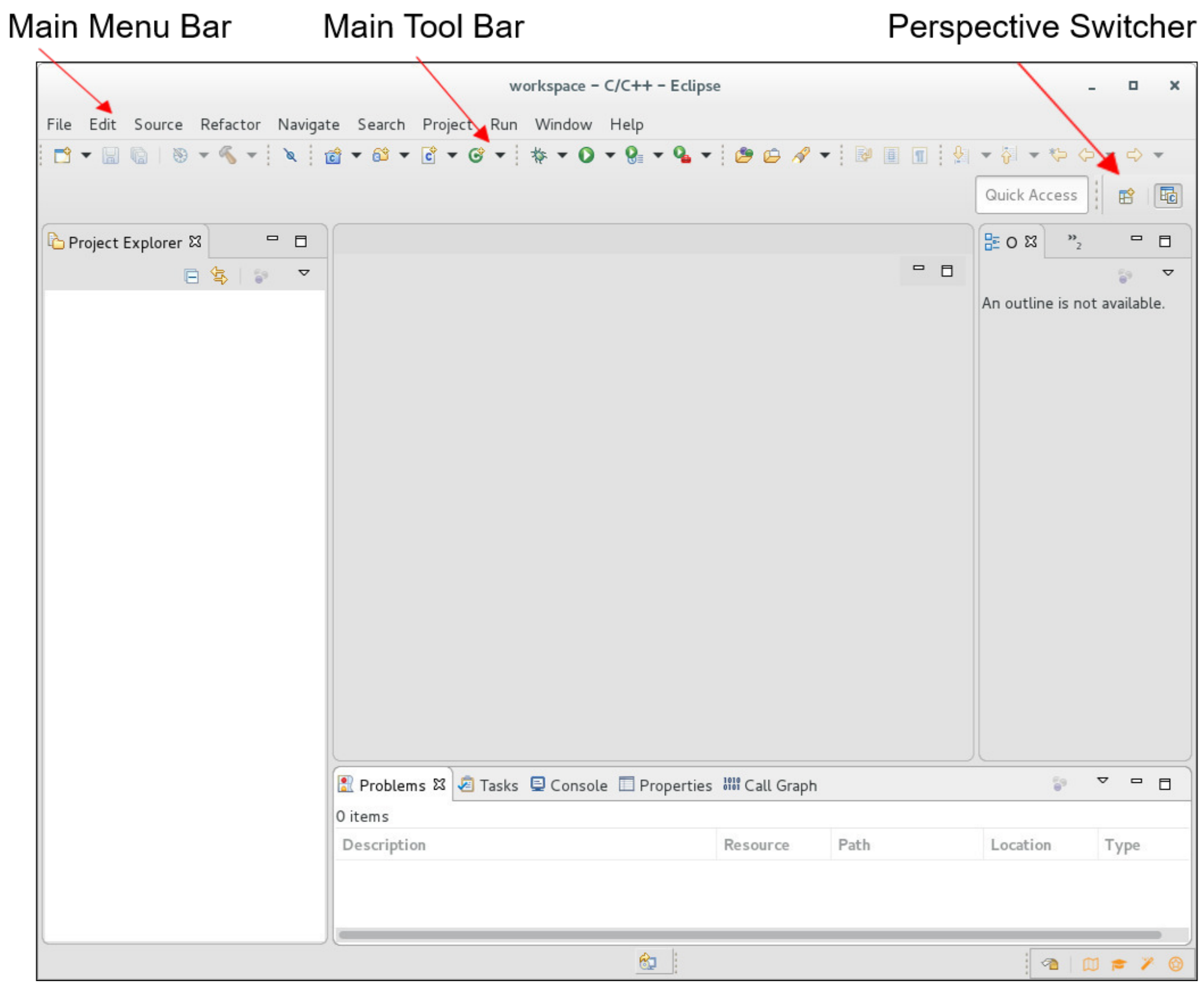
- Click the workbench arrow button in the upper right corner
- When the workbench first opens, the IDE defaults to the C/C++ perspective1 as show in figure below

Note: Section 3 above explains how to open the Perspective and how to use the IDE for OpenCPI development is explained in Section 5.

Note: Even if Eclipse projects are placed in the workspace directory, Eclipse will not bring them into the workspace. Projects must be imported or locally created to be appear in the Project Explorer View.

OpenCPI projects will not appear in the OpenCPI Projects View unless they are open projects.

Figure 49: Eclipse Workspace


Main Menu Bar
Main Tool Bar
Perspective Switcher
workspace - C/C++ - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Project Explorer
An outline is not available.
Problems Tasks Console Properties Call Graph
0 items
Description Resource Path Location Type

B. Eclipse Basic Concepts

Eclipse uses **Perspectives** to provide various sets of tools that together focus on a given perspective of development. A perspective consists of a workbench layout of a complimentary set of Eclipse views. A **view** is a workbench panel that provides a specific capability. For example, the Eclipse Project Explorer View provides a graphical view of the file system and the ability to navigate through and act upon it. The Window Tab in the Main Menu Bar at the top of the window provides navigation to the available perspectives and views, as well as, workbench window controls and user preferences.

A final note: the user can rearrange a perspective layout, add, or remove views, and modify panel sizes. These setting are saved and become the default layout for that perspective. Projects must be imported to appear in the workspace. The Eclipse Project Explorer provides these controls via a context menu that is opened by right-click when the cursor is in the view. This menu provides access to create new things in the workspace or project, controls to open or closed a project and to remove files, folders, and projects from the workspace.

Caution! Using Project Explorer to delete OpenCPI assets can cause problems; deleting/renaming source files is okay. It is okay to remove a project from the workspace (delete is used), however, do not remove OpenCPI project contents from the file system unless it has been unregistered.

The OpenCPI Projects View or the ocpidev command should be used to delete OpenCPI projects and assets. Editors and tool chains – Review the Workbench Overview panel and look over C/C++ Development documentation.

C. Additional Plugins for the IDE

If the development machine has access to the internet or a mirrored Eclipse Marketplace, consider the following software to enhance the AV IDE experience. The Marketplace link is found under the Help link in the top navigation bar. The Marketplace makes it very easy to search, browse and review software. Using the IDE navigate to Help-Eclipse Marketplace. Search for the packages below; it is a simple install button click to get the package.

TM Terminal 4.0

Search for Terminal in the Marketplace to see a number of available terminal packages. The TM Terminal 4 has been used successfully by the AV team. Click the install button to install the product. Typically, an agreement must be accepted to continue.

D. Build The IDE

Introduction

The OpenCPI/ANGRYVIPER IDE Plugin can be built from source with the following instructions. These commands assume they are performed by the “root” user, while normally discouraged, is usually performed within a Docker container running a Fedora OS.

Instructions

Follow the steps below, in the directory of your choice “`$DIR`”.

1. Install Sapphire Dependency

   cmd: `mkdir $DIR/temp && cd $DIR/temp`

   cmd: `wget --output-document=tempo.zip "http://www.eclipse.org/downloads/download.php?file=/sapphire/9.1/sapphire-repository-9.1.zip&r=1"`

   cmd: `unzip -q tempo.zip`

   cmd: `mkdir -p /usr/share/java/sapphire/`

   cmd: `mv plugins/*jar /usr/share/java/sapphire/`

   cmd: `rm -rf *`

2. Install Web Tools Platform (WTP) Dependency

   cmd: `mkdir $DIR/temp && cd $DIR/temp`

   cmd: `wget --output-document=tempo.zip "https://www.eclipse.org/downloads/download.php?file=/webtools/downloads/drops/R3.9.5/R-3.9.5-20180409100740/wtp-repo-R-3.9.5-20180409100740.zip&r=1"`

   cmd: `unzip -q tempo.zip`

   cmd: `mkdir -p /usr/share/java/wtp/`

   cmd: `mv plugins/*jar /usr/share/java/wtp/`

   cmd: `rm -rf *`

3. Install Eclipse Dependencies

   cmd: `dnf install eclipse-gef-sdk tycho tycho-extras`

4. Download Eclipse Neon

   cmd: `cd $DIR`

   cmd: `wget --output-document=eclipse.tar.gz "https://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/neon/3/eclipse-cpp-neon-3-linux-gtk-x86_64.tar.gz&r=1"`

   cmd: `tar xf eclipse-cpp-neon-3-linux-gtk-x86_64.tar.gz`

5. Use Eclipse Command-Line To Download Prerequisites

   cmd: `cd $DIR`

   cmd:
   ```
   ./eclipse/eclipse \
   -clean -purgeHistory -noSplash -destination ./temp \
   -application org.eclipse.equinox.p2.director \
   -repository \
    "http://download.eclipse.org/tools/gef/updates/legacy/releases/, \
    http://download.eclipse.org/releases/neon" \
   -installIUs "\
    org.eclipse.gef.sdk.feature.group, \
    org.eclipse.jdt.feature.group, \
    org.eclipse.sapphire.feature.group, \
    org.eclipse.sapphire.ui.feature.group, \
   ```

```
org.eclipse.sapphire.platform.feature.group, \
org.eclipse.sapphire.java.jdt.feature.group, \
org.eclipse.sapphire.java.feature.group, \
org.eclipse.sapphire.osgi.feature.group, \
org.eclipse.sapphire.sdk.feature.group, \
org.eclipse.sapphire.ui.swt.gef.feature.group, \
org.eclipse.sapphire.ui.swt.xml.editor.feature.group, \
org.eclipse.sapphire.modeling.xml.feature.group"
```

cmd: `mkdir -p $DIR/eclipse/av_prereqs/plugins`

cmd: `cp -r $DIR/temp/plugins/. $DIR/eclipse/av_prereqs/plugins`

cmd: `cd $DIR/eclipse/av_prereqs/plugins`

cmd: `ls -1 $DIR/eclipse/plugins/ | xargs -r rm -rf`

6. Clone OpenCPI IDE repository

   cmd: `cd $DIR`

   cmd: `git clone https://github.com/opencpi/angryviper_gui.git`

   cmd: <`git checkout` appropriate branch>

7. Build OpenCPI Plugin

   cmd: `cd angryviper_gui/av.proj.ide.tycho`

   cmd: `xmvn -o clean verify`

8. Combine Prerequisites + OpenCPI Plugin + Eclipse

   cmd: `mdkir -p $DIR/eclipse/dropins/angryviper/plugins`

   cmd: `mv $DIR/eclipse/av_prereqs/plugins/* $DIR/eclipse/dropins/angryviper/plugins/`

   cmd: `rm -rf $DIR/eclipse/av_prereqs`

   cmd: `mv $DIR/angryviper_gui/av.proj.ide.tycho/releng/av.proj.ide.update/target/repository/plugins/*.jar $DIR/eclipse/dropins/angryviper/plugins/`

9. Start Eclipse With IDE Plugin

   cmd: `$DIR/eclipse/eclipse &`