




ANGRYVIPER (AV) IDE USER GUIDE

SEPTEMBER 26, 2018

IDE USER GUIDE
ANGRYVIPER TEAM

BIA-Boeing
Annapolis Junction, MD



Revision History

TABLE 1 REVISION HISTORY

Revision	Description of Change	Date
1	Created	2/2018
2	Revision for v.1-4	9/2018

Contents

Revision History	1
Table of Figures	4
1.0 References	6
1.1 Assumptions	6
2.0 Introduction	6
2.1 New Features in the 1.4 Release	7
3.0 Overview	8
3.1 AV Perspective Views	8
3.1.1 Open the AV Perspective	9
3.1.2 Adding the Perspective to the Tool Bar	10
4.0 AV IDE Overview and Features	11
4.1 Overview	11
4.2 OpenCPI Projects View Features	11
4.2.1 Building from the OpenCPI Projects View	12
4.3 Operations View Features	12
4.3.1 Assets Mode	13
4.3.2 Unit Test Panel	14
4.3.3 Unit Test Features	15
4.4 Build Status View Features	16
4.5 Eclipse Project Explorer View	18
4.6 Eclipse Console View	19
4.6.1 Notice Console	19
4.6.2 Execution Consoles	20
4.7 AV OpenCPI Asset Wizard	20
4.8 OpenCPI Asset XML Editors	21
4.8.1 Form Based Graphical Editors	21
4.8.2 Graphical Drag-and-Drop Editors	24
4.8.3 Asset XML Examples	24
4.8.4 Additional Notes About the AV Perspective	25
4.9 The Execution Configuration	25
4.9.1 XML Editors -Modifying Existing XML Files	25
4.9.3 Warning about a Non-OpenCPI Project	26

4.9.4 The Perspective is Not in Sync with the Workspace or the File System	26
5.0 OpenCPI Development Workflow Using the IDE	26
5.1 Introduction.....	26
5.2 OpenCPI the Core and Assets Projects.....	27
5.2 1 Importing OpenCPI and Asset Projects	28
5.2.1 Build the Projects for the desired RCC and HDL Platforms.....	29
5.3 Creating New Projects and Libraries.....	31
5.3.1 More on OpenCPI Projects.....	32
5.3.2 Project Registration	33
5.3.3 Errors in Creating New Project	33
5.3.4 OpenCPI Libraries	33
5.4 Creating Components, Protocols and Workers	33
5.4.1 Component Example from the Getting Started Guide	34
5.4.2 Worker Example from the Getting Started Guide	35
5.5 Creating Applications and Assemblies.....	37
5.5.1 Adding Components to an Application	38
5.5.2 Connecting Application Components	41
5.5.3 Assembly Editor.....	44
5.6 Creating Component Unit Tests.....	45
Appendix A.....	47
A.1 Eclipse Basics.....	47
A.2 Eclipse Basic Concepts.....	50
Appendix B.....	51
B.1 Addition Plugins for the IDE	51
B.2 TM Terminal 4.0.....	51
B.3 ANSI Escape in Console Plugin	51

Table of Figures

Figure 1 AV IDE showing the C/C++ Perspective	9
Figure 2 Open the Perspective and Add it to the Toolbar	10
Figure 3 List of Features	11
Figure 4 OpenCPI Projects View	12
Figure 5 Build Configuration Setup in the Operations Panel	13
Figure 6 Unit Tests Panel	14
Figure 7 List Panel Showing Two Different Remotes	15
Figure 8 Build Status View Features	16
Figure 9 Expanded View of Successful Build Execution	17
Figure 10 Status Bar	17
Figure 11 Eclipse Project Explorer View	18
Figure 12 Eclipse Console View	19
Figure 13 Notice Console	19
Figure 14 Execution Console	20
Figure 15 OpenCPI Asset Creation Wizard	20
Figure 16 Modify Attributes of HDL Worker	22
Figure 17 Add a Property to Worker	22
Figure 18 HDL Property Element Form	23
Figure 19 Diagram interface to generate XML files	24
Figure 20 PROCEDURES TO USE EXISTING PROJECTS	27
Figure 21 Importing OpenCPI and Asset Projects	28
Figure 22 Build Entry	29
Figure 23 Project is built from the Operations Panel	30
Figure 24 Core project built from the operations Panel	30
Figure 25 Create a new project	31
Figure 26 Asset Wizard	32
Figure 27 Terminal Window View of a project registry	32
Figure 28 Create a New OpenCPI Asset	34
Figure 29 Port Form for the ramp In	35
Figure 30 Worker Example	35
Figure 31 Addition of In and Out Interfaces	36
Figure 32 Asset Wizard	37

Figure 33 Application Editor	38
Figure 34 properties view	38
Figure 35 Adding the ramp component	39
Figure 36 Third Method to add a component instance.....	40
Figure 37 Select File Write as the new companion.....	40
Figure 38 Design view of the Application.....	41
Figure 39 Example of a Simple Connection.....	42
Figure 40 Pop Up Panel	43
Figure 41 Advanced Connection Views	43
Figure 42 Square HDL Worker	44
Figure 43 All inputs are Preset for the Ramp Spec.....	45
Figure 44 Unit Test Editor Panel	46
Figure 45 Display of Attribute Choice	46
Figure 46 Eclipse Launcher.....	47
Figure 47 Eclipse Welcome Screen.....	48
Figure 48 Eclipse Workspace.....	49

Hint! Ctrl + Click on the desired page number for quick navigation around the document.

1.0 References

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

1.1 Assumptions

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

TABLE 2 REFERENCE DOCUMENTS

Title	Published By	Link
Installation Guide	AV Team	RPM Installation Guide
Acronyms & Definitions	AV Team	Acronyms and Definitions
OpenCPI Website	OpenCPI	www.opencpi.org
Getting Started Guide	AV Team	opencpi.github.io/Getting_Started
OpenCPI Component Development Guide	AV Team	opencpi.github.io/OpenCPI_Component_Development
OpenCPI Application Development Guide	AV Team	opencpi.github.io/OpenCPI_Application_Development

2.0 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 existing Eclipse installation. Both methods are described in the RPM Installation Guide. Section 6 addresses the RPM Install and Appendix D addresses the plugin install. http://opencpi.github.io/RPM_Installation_Guide.pdf.

If the RPM method is used, the IDE is installed in **opt/opencpi/gui** and the command **ocpigui** is provided to start the IDE. If IDE is constructed using the plugin method, simply starting your Eclipse provides the IDE.

2.1 New Features in the 1.4 Release

In the 1.4 release, the IDE has expanded features in the AV Perspective that allows more capability to the OpenCPI Projects View that reduces the need to use command line and the Eclipse Project Explorer. New features to support the OpenCPI workflow include:

1. Asset Creation and Deletion
2. OpenCPI Project Registration
3. The Project View displays all supported assets and opens asset XML Editors
4. A Context menu that provides useful options based on the Selected asset
5. The AV Assets Wizard is now integrated into the AV Perspective, so it does not need to be refreshed when assets are created.
6. The IDE now fully supports component unit test creation and execution. Unit test creation is added to the Assets Wizard and a unit test Editor now supports test XML. Finally, a full feature unit test control panel is integrated into the Operations View to give test build and run capability.

3.0 Overview

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

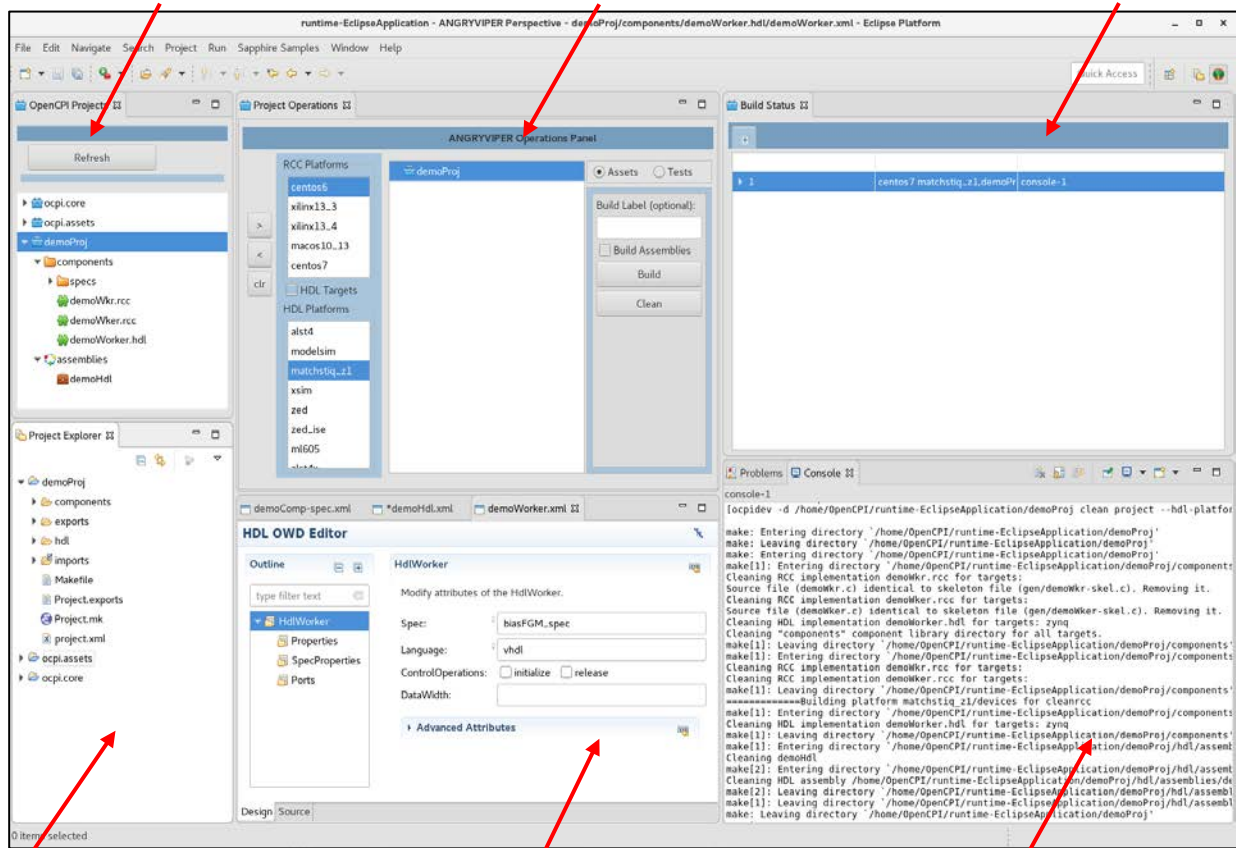
3.1 AV Perspective Views

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

OpenCPI Project View

AV Operation View

Build Status View



Eclipse Project Explorer View

Eclipse Text Editor Panel

Console View

THE ANGRYVIPER PERSPECTIVE

The full descriptions of the views are provided in Section 4. Below are brief summaries of each:

- **OpenCPI Project 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. This required a new control Panel and the user toggles between building and testing controls.
- **Build Status View** – provides a graphical view of build/run execution configurations and execution status. The user can also re-run builds and tests using this view. The resulting color after build process in this view convey meaning as follows:
 - White shows an active execution
 - Green indicates successful completion
 - Red shows a failure
 - Yellow shows an execution stopped by the user
- **Project Explorer View** – gives a view into a projects file system. This view primarily supports code development. It is included in the perspective for reasons provided in Section 4.
- **Eclipse Editor Panel** – Asset and text editors open in this Panel.
- **Eclipse Console View** – shows the various ocpi.dev command executions based on user requests as well as the result of the requests.

3.1.1 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 tool bar for regular use.

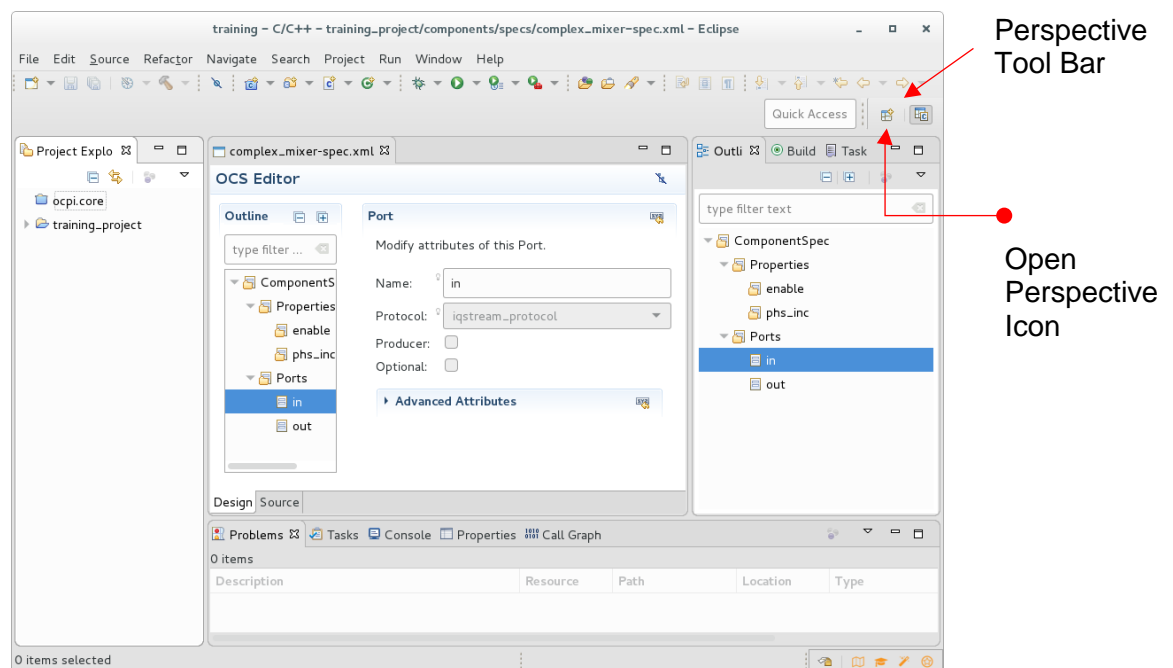


FIGURE 1 AV IDE SHOWING THE C/C++ PERSPECTIVE

3.1.2 Adding the Perspective to the Tool Bar

To open the Perspective and add it to the perspectives tool bar:

- Click the open perspective icon and Select it from the perspective list.
- Eclipse will switch to the perspective and add the AV icon to the perspectives tool bar.
- Click the perspectives tool bar Selections to change between perspectives.

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 add views to it.

It will then save the rearranged perspective and will display it from this way forward. (if the workspace is preserved).

The AV Perspective default layout was selected because it provides a core complement of tools to accomplish OpenCPI operations.

To see a listing of Eclipse perspectives:

- Navigate to Window → Perspective
- Open Perspective

A listing of views can be seen by navigating to Window → Show View.

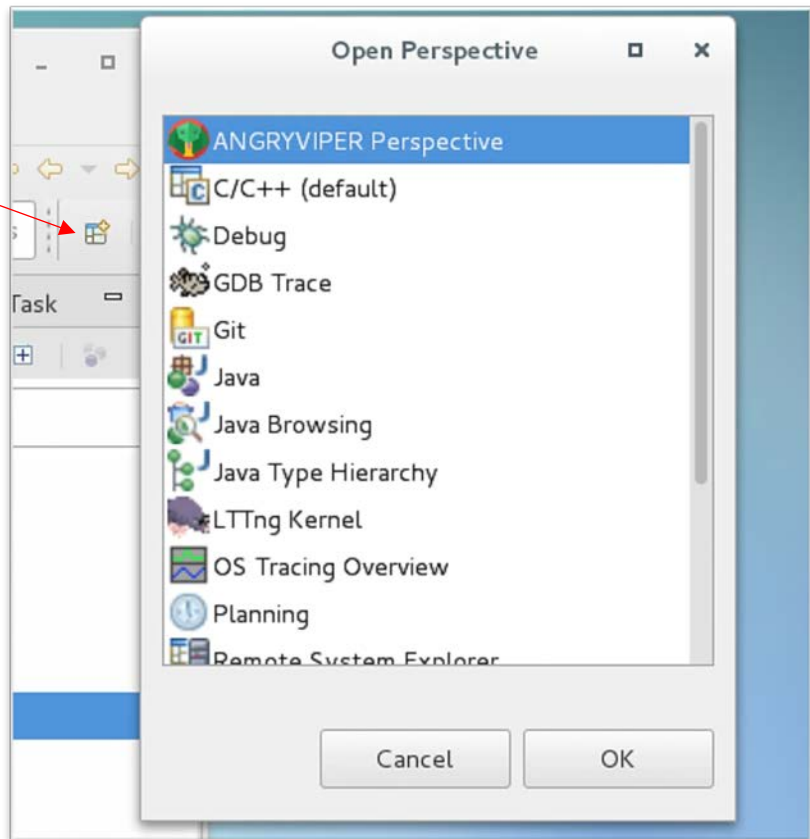


FIGURE 2 OPEN THE PERSPECTIVE AND ADD IT TO THE TOOLBAR

4.0 AV IDE Overview and Features

4.1 Overview

The AV Perspective default layout was Selected because it provides 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.

4.2 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. By double-clicking on an individual asset will perform open 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
- **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

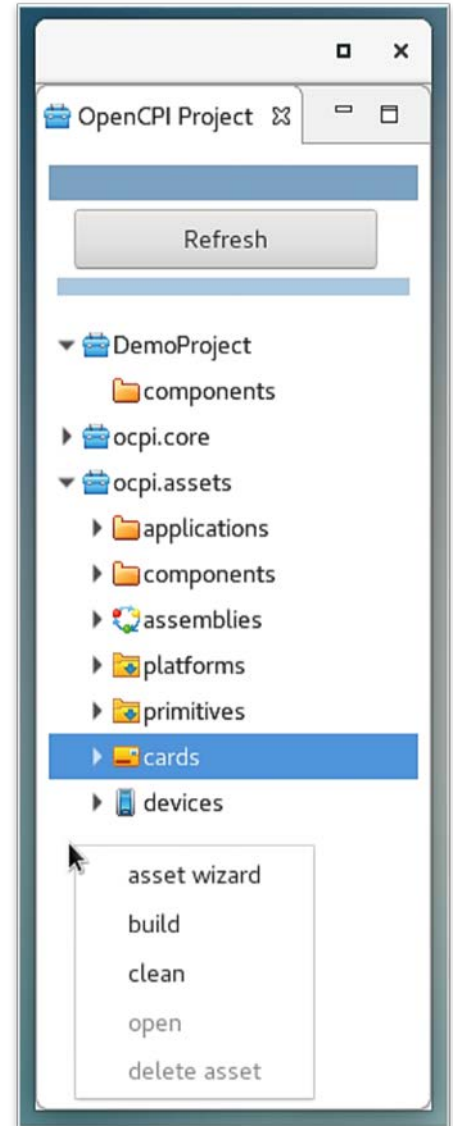


FIGURE 3 LIST OF FEATURES

4.2.1 Building from the OpenCPI Projects View

Select one or more assets in the view; right-click then 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. Figure 4 below displays a launch from the OpenCPI Projects View and the status bar for the build.

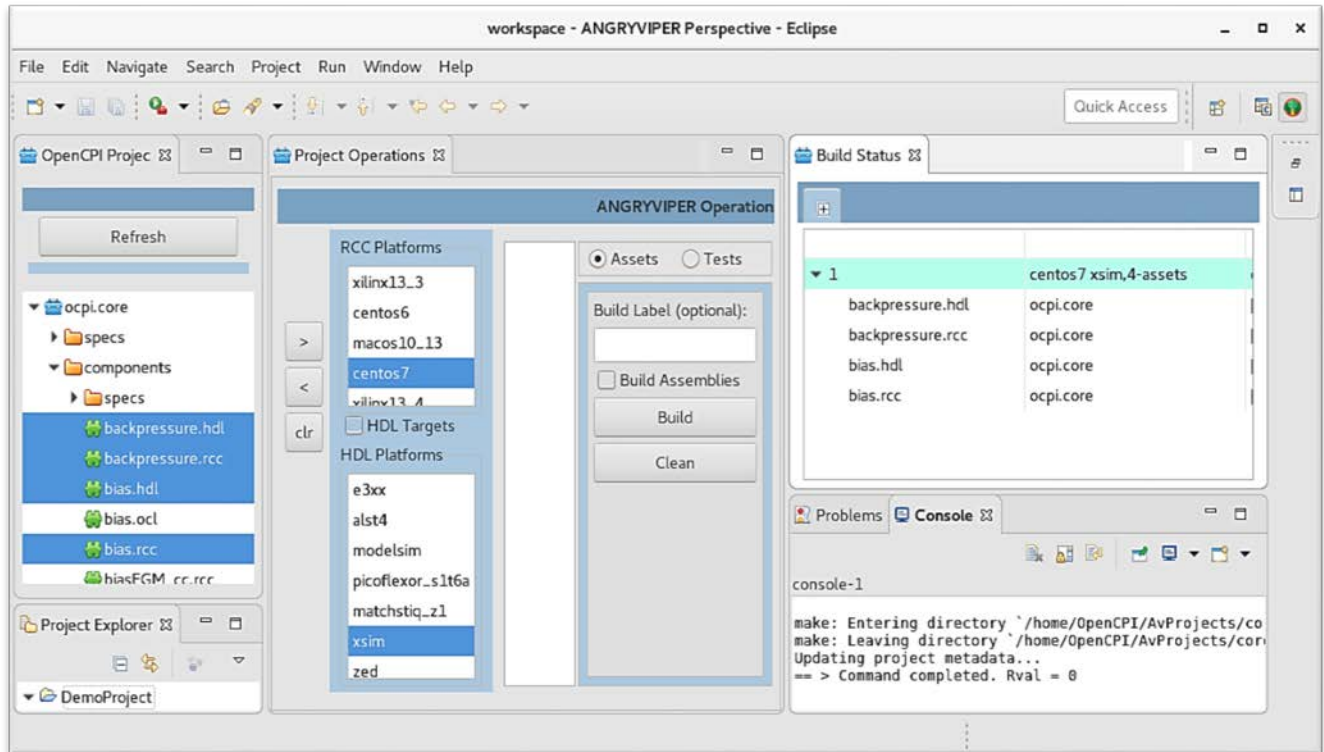


FIGURE 4 OPENCPI PROJECTS VIEW

4.3 Operations View Features

The AV Operations View is used to build OpenCPI assets and to build and run application component unit tests. This Panel has two modes of operation. The assets mode give asset build and clean operations and the 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
- **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 while working with an email app) and press the add button. One or more platforms may be selected for a build. The operations panel opens in assets mode by default. The radio buttons on the top of the Build Controls section toggle controls for building assets and running unit tests.

4.3.1 Assets Mode

A build or clean execution is initiated by the Build/Clean buttons. Once initiated, an execution configuration is established. The user may rerun the configuration as either, a build or clean from this Panel or the Status Monitor View.

Figure 6i 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 ocpi.dev 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, and individual assets).

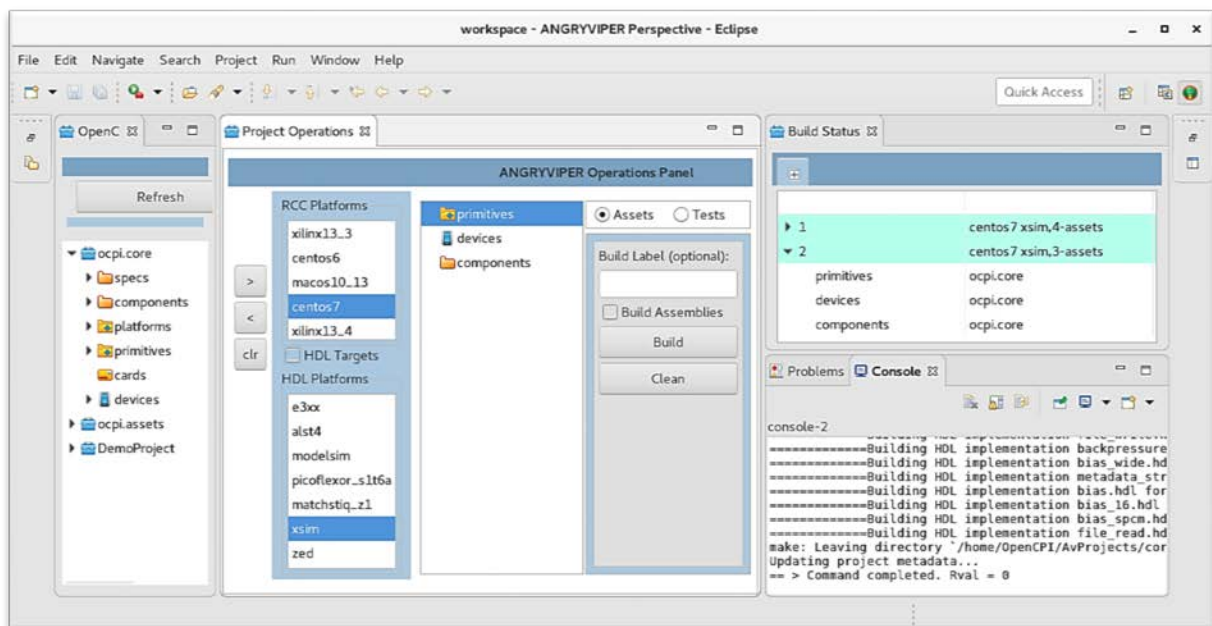


FIGURE 5 BUILD CONFIGURATION SETUP IN THE OPERATIONS PANEL

4.3.2 Unit Test Panel

The Unit Tests Panel (shown below in Figure 7) 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:

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

The Tests Panel displays support for these phases, (currently, there is no discrete support for the Build Phase) and provides three buttons that combine unit test phases that are performed during development and debugging of a unit test.

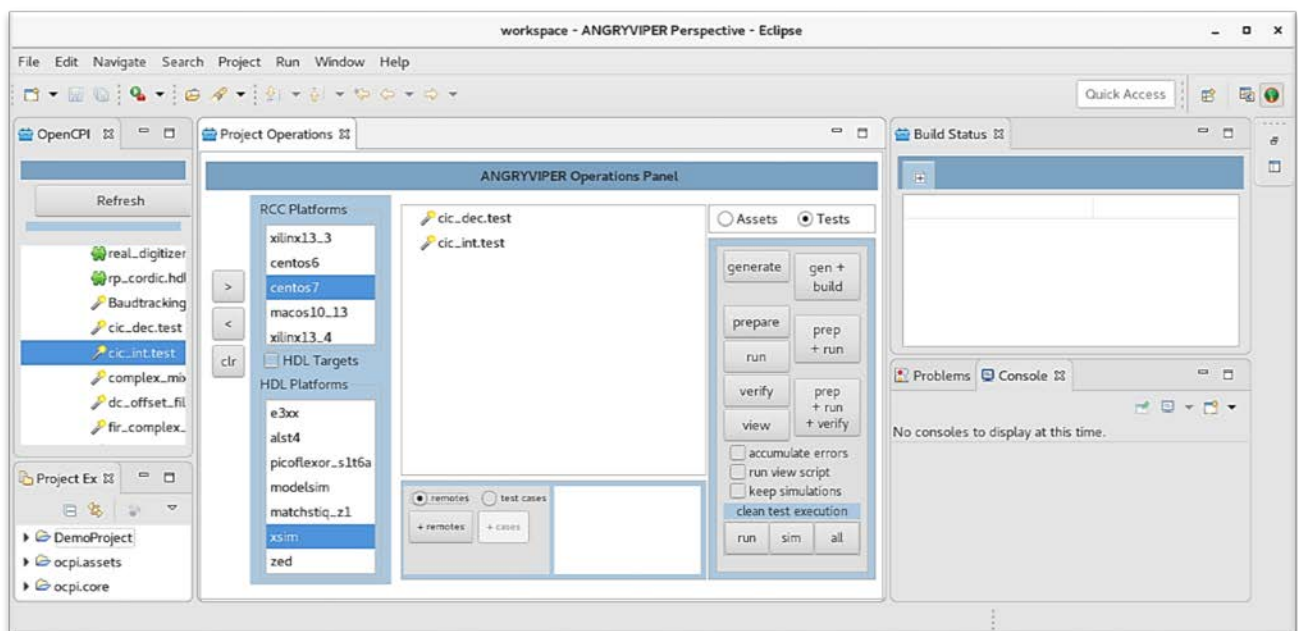


FIGURE 6 UNIT TESTS PANEL

4.3.3 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.

Warning: “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.

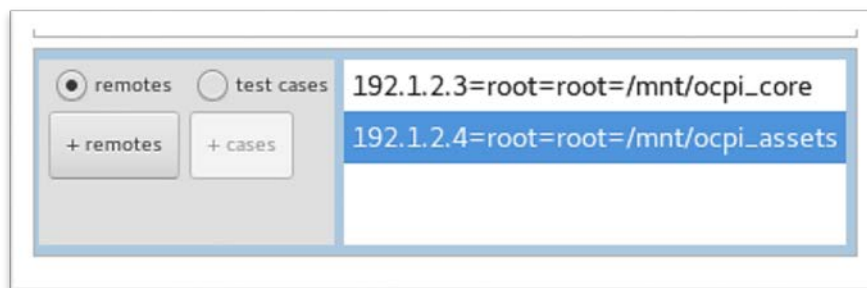


FIGURE 7 LIST PANEL SHOWING TWO DIFFERENT REMOTES

4.4 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
- **Yellow** - Indicates an execution stopped by the user

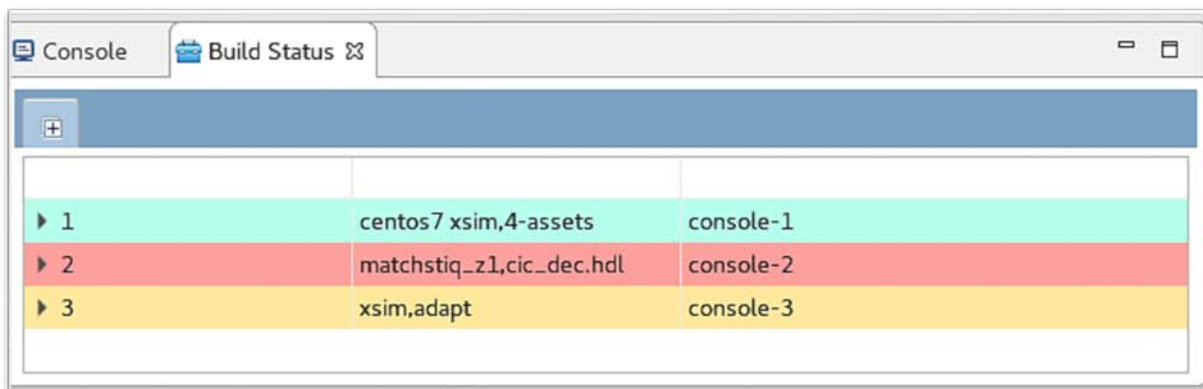


FIGURE 8 BUILD STATUS VIEW FEATURES

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; as rows are added as the execution proceeds.

Figure 9 demonstrates the status bar for a build that has completed successfully and the expanded view.

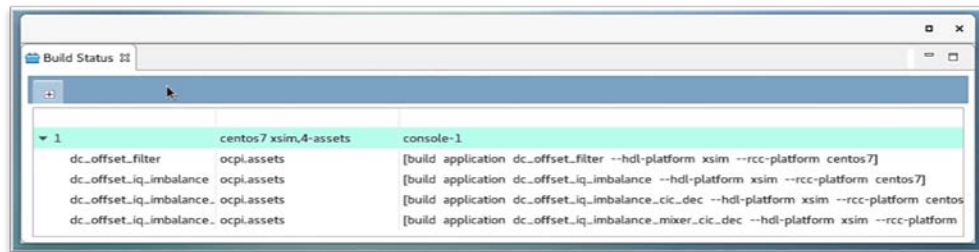


FIGURE 9 EXPANDED VIEW OF SUCCESSFUL BUILD EXECUTION

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.

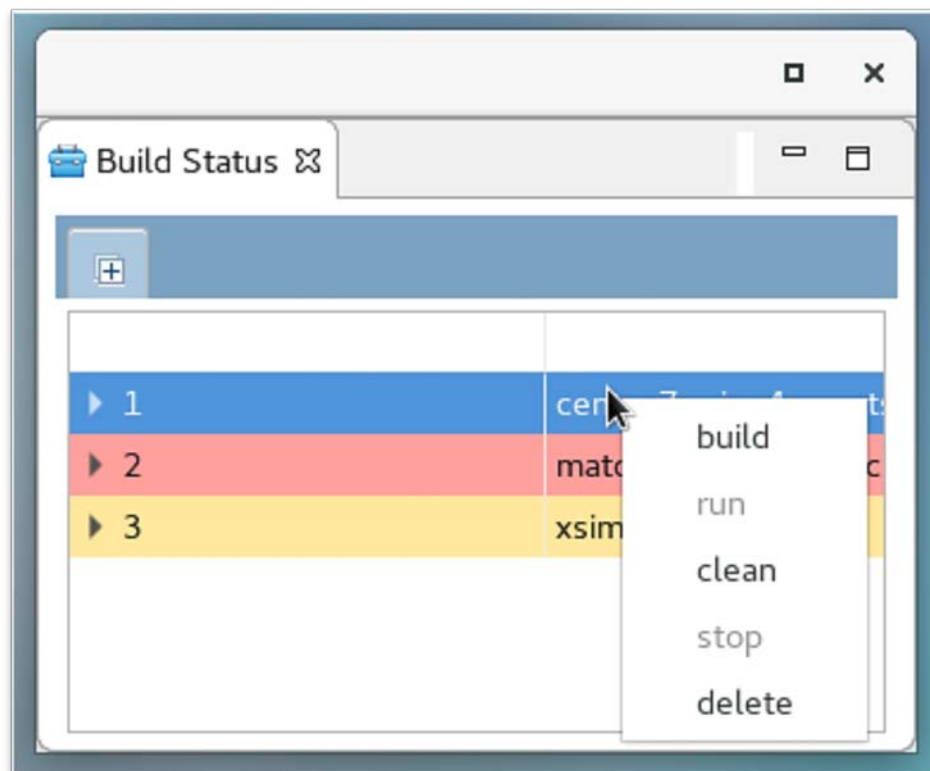


FIGURE 10 STATUS BAR

4.5 Eclipse Project Explorer View

The Eclipse Project explorer is provided in the AV Perspective because it provides a **file system view** of the projects.

This view has a right-click context menu that provides workspace features such as project import, workspace/project refresh, and access to the AV 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.

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 as it executes the underlying ocpidev delete command that completely removes the asset from the framework infrastructure.

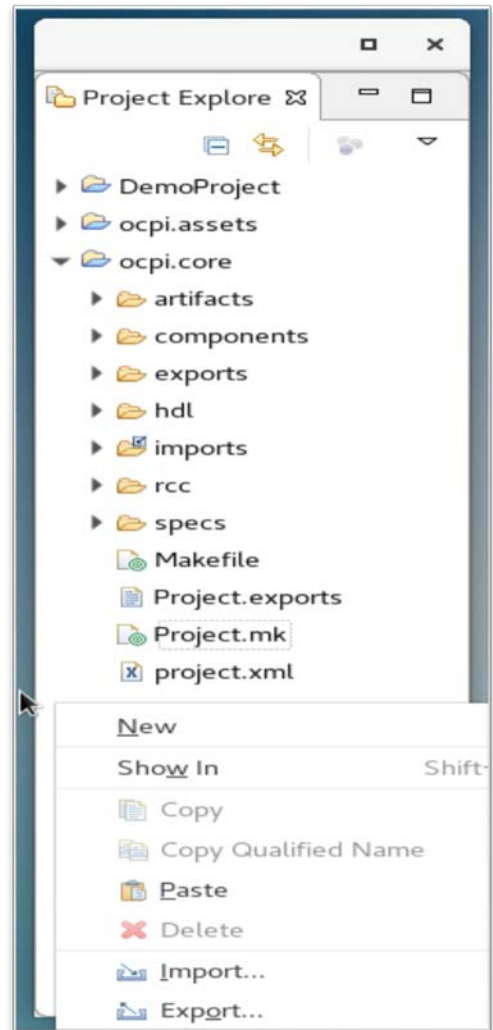


FIGURE 11 ECLIPSE PROJECT EXPLORER VIEW

4.6 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 twenty-five execution consoles. There are some useful features in the view toolbar: **Clear**, **Scroll Lock**, **Display Selected Console**.

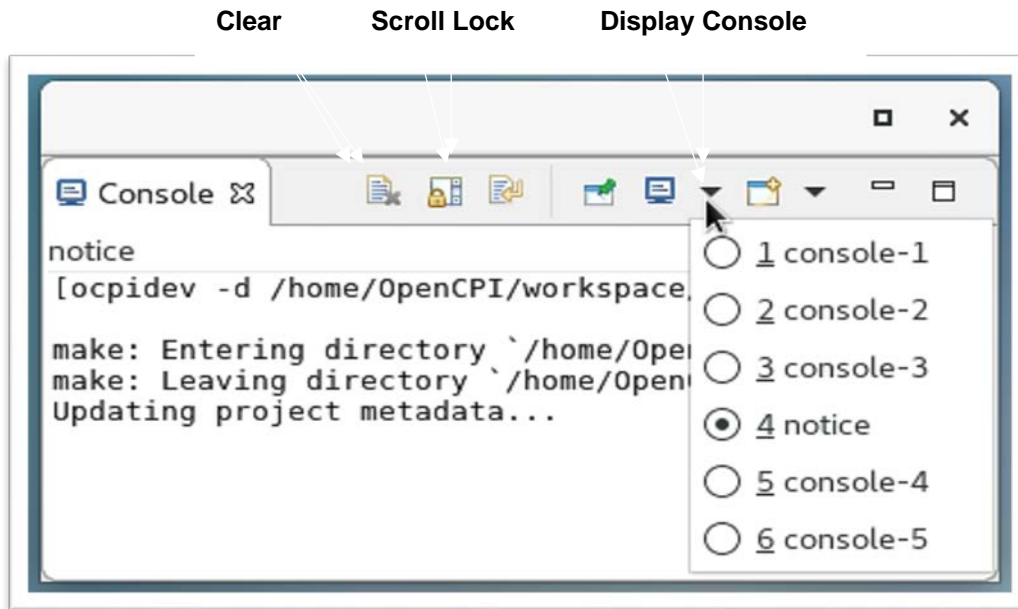


FIGURE 12 ECLIPSE CONSOLE VIEW

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

4.6.1 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.

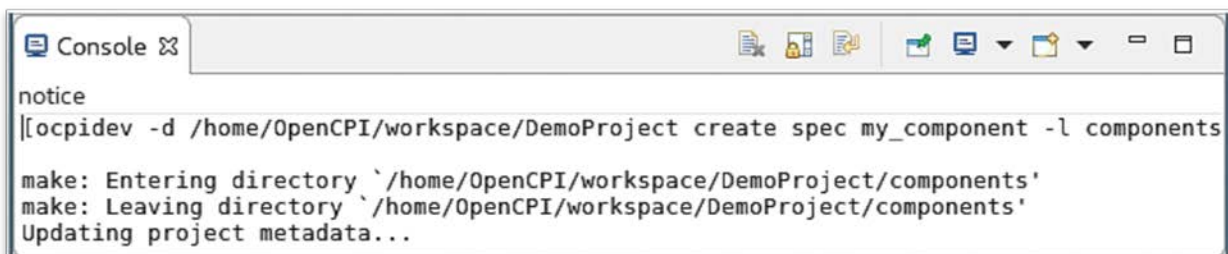
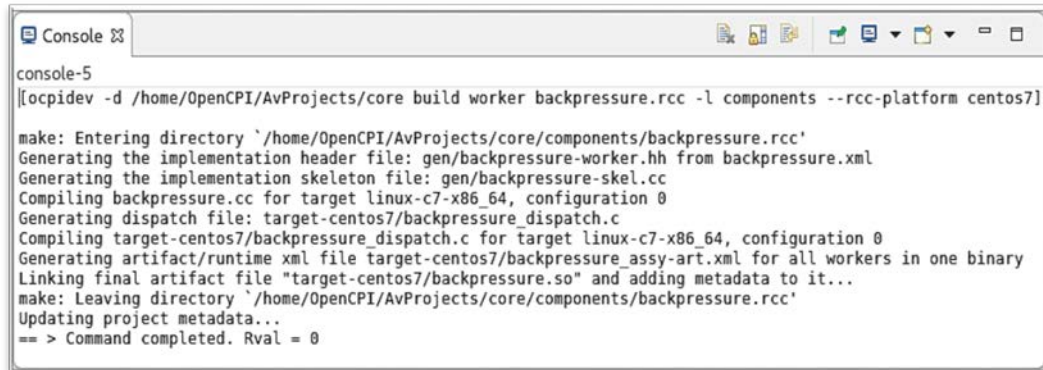


FIGURE 13 NOTICE CONSOLE

4.6.2 Execution Consoles

Execution consoles provide **build and run** commands and their output. The IDE supports up to twenty-five active execution configurations at a time and each configuration gets its own console. Figure 15 shows a console for a successful build.



```
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
```

FIGURE 14 EXECUTION CONSOLE

4.7 AV 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 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.

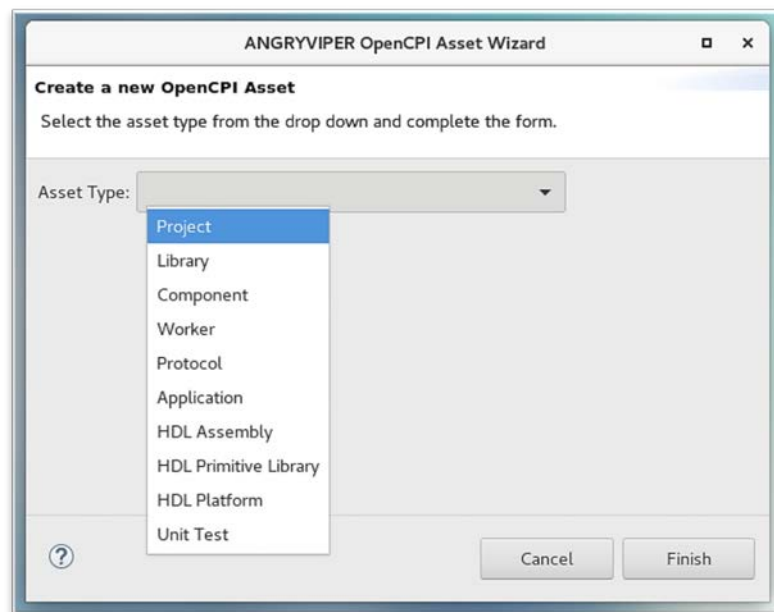


FIGURE 15 OPENCPI ASSET CREATION WIZARD

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.

4.8 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
- **RCC Application Worker** – OWD RCC Editor
- **HDL Application Worker** – OWD HDL Editor
- **HDL Assembly** – OHAD Editor
- **OpenCPI Application** – OAS Editor
- **HDL Platform**
- **Component Unit Tests**

There are two implementations of the XML Editors in the AV plugin. The component Editors are form based while the application and assembly Editors are diagram based and support the form-based feature.

4.8.1 Form Based Graphical Editors

The Graphical Editor has two panels:

1. Outline Panel
2. Form-Input Panel

The Outline Panel is used to navigate the XML elements for this worker. These are the top-level HdIWorker element 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.

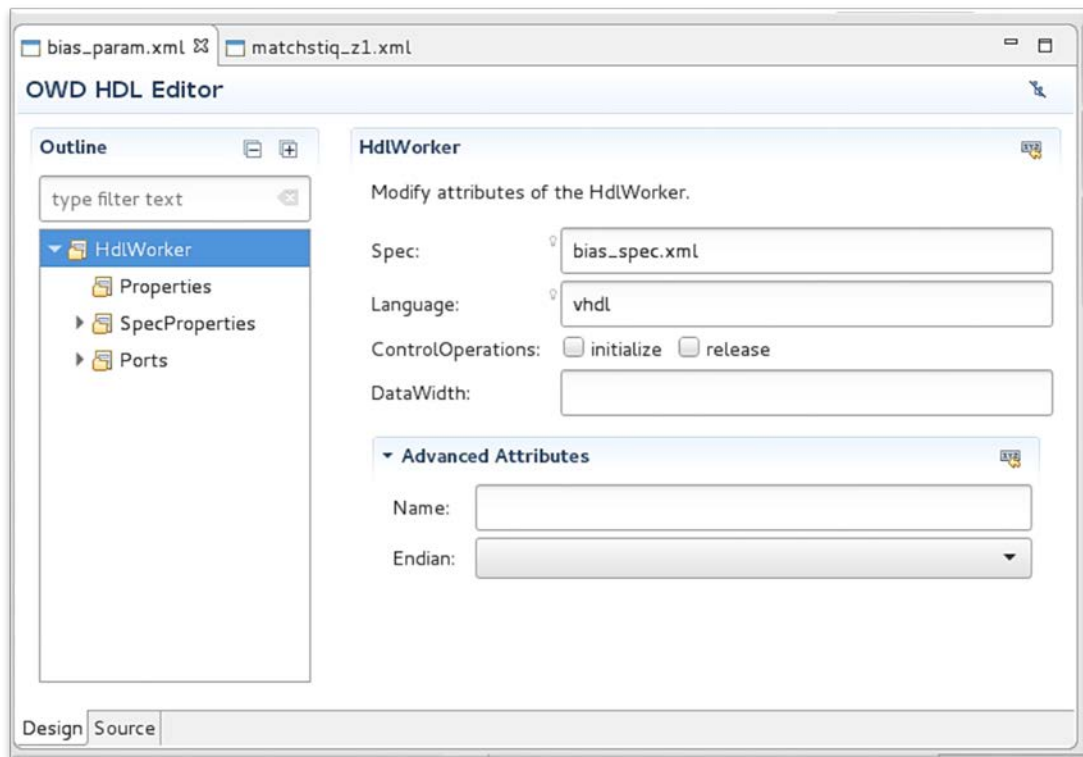


FIGURE 16 MODIFY ATTRIBUTES OF HDL WORKER

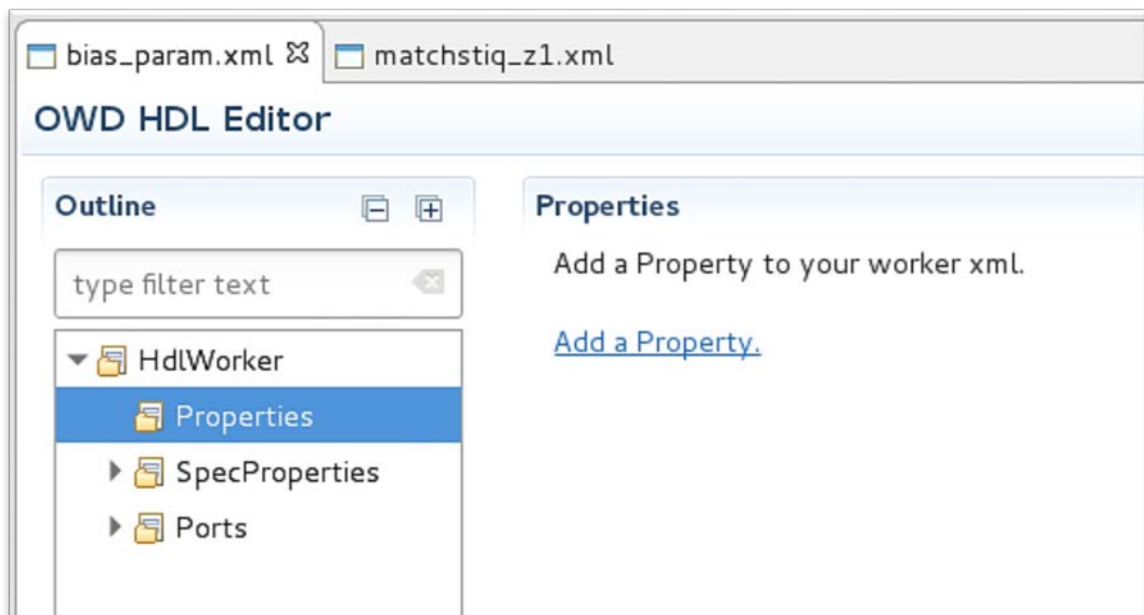


FIGURE 17 ADD A PROPERTY TO WORKER

In the adjacent example a property element is added to the worker. Select the Properties element in the Outline, right Click → Add Property. The form to populate the new element appears as shown in Figure 19 below.

The figure below displays the property form.

The screenshot shows the OWD HDL Editor interface. On the left is the 'Outline' pane with a tree view containing 'HdlWorker', 'Properties', 'newProperty' (selected), 'SpecProperties', and 'Ports'. The main area is titled 'Property' and contains the following fields:

- Name:** newProperty
- Type:** (dropdown menu)
- Default:** (text input)
- SequenceLength:** (text input)
- ArrayDimensions:** (text input)
- Parameter:** ☐

Below these fields are two sections with checkboxes:

- Accessibility:**
 - Readable: ☐
 - Volatile: ☐
 - Writable: ☐
 - Initial: ☐
 - Padding: ☐
- Advanced Attributes:**
 - ReadSync: ☐
 - WriteSync: ☐
 - ReadError: ☐
 - WriteError: ☐

FIGURE 18 HDL PROPERTY ELEMENT FORM

Select the source Tab to see how the new element is added to the XML file.

To remove an element: Select it, 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 detailed information about the XML files these Editors support.

Note: *these Editors provide a basic capability to create these files.*

4.8.2 Graphical Drag-and-Drop Editors

The AV IDE Application and Assembly Editors provide a diagram interface to generate the respective XML files. (This Editor is described in detail in Section 5.)

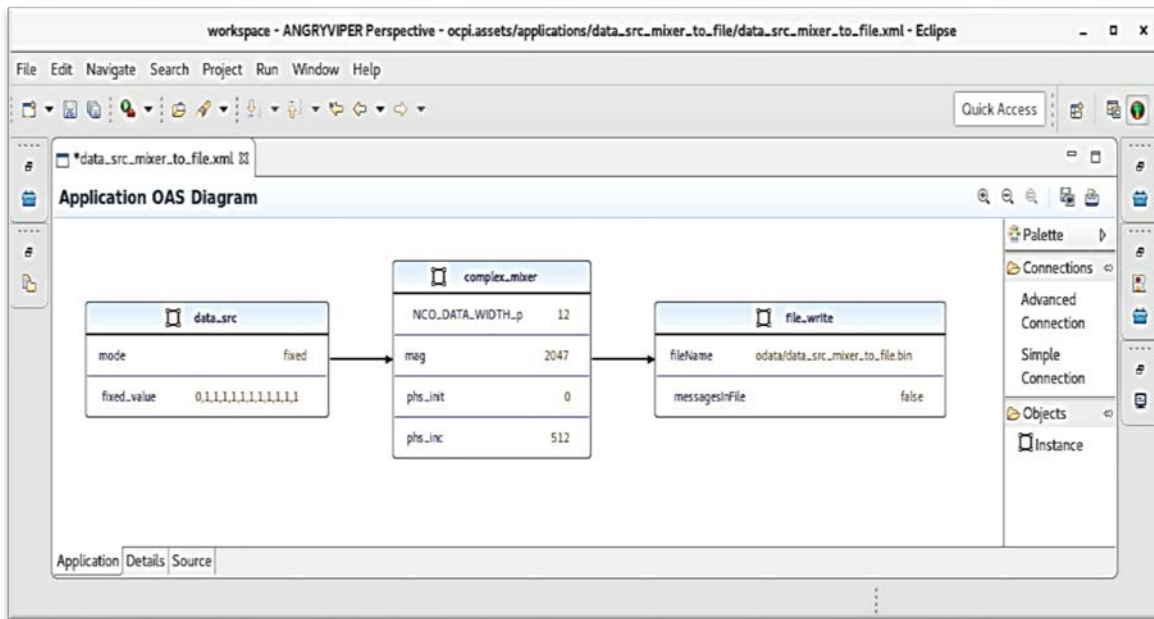


FIGURE 19 DIAGRAM INTERFACE TO GENERATE XML FILES

4.8.3 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. **OWD HDL 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

4.8.4 Additional Notes About the AV Perspective

The AV 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.

4.9 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 AV 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 twenty-five.

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 Selections for this reason. Once done execution numbers will continue to increase but consoles will be reused.

4.9.1 XML Editors -Modifying Existing XML Files

The following Editors will modify existing XML files when they are opened with the Editor. The version 1.4 IDE will indicate this is occurring and why; however, it will only do it once per Eclipse session. The Editors are:

- **OCS Editor**
 - The Component Editor corrects the primitive *type* attribute in the property element.
- **OAS Editor**
 - The Application Editor adds a name attribute to the instance element to better support presentation.
- **HDL Platform Editor**
 - This Editor adds an extension element to the signal element to support presentation of signal direction and name. The OpenCPI Framework ignore the extension element.

The modifications do not have to be saved.

4.9.3 Warning about a Non-OpenCPI Project

The AV 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.

4.9.4 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 (yet). This means changes made to a project outside of the IDE will 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.

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

5.0 OpenCPI Development Workflow Using the IDE

5.1 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. [//opencpi.github.io/Getting_Started](https://opencpi.github.io/Getting_Started). 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.

5.2 OpenCPI the Core and Assets Projects

A number of existing projects are provided in OpenCPI Release 1.4 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:

- Select an area in the file system to put these projects and create the folder if necessary.
- 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.
- Import the desired projects into Eclipse.

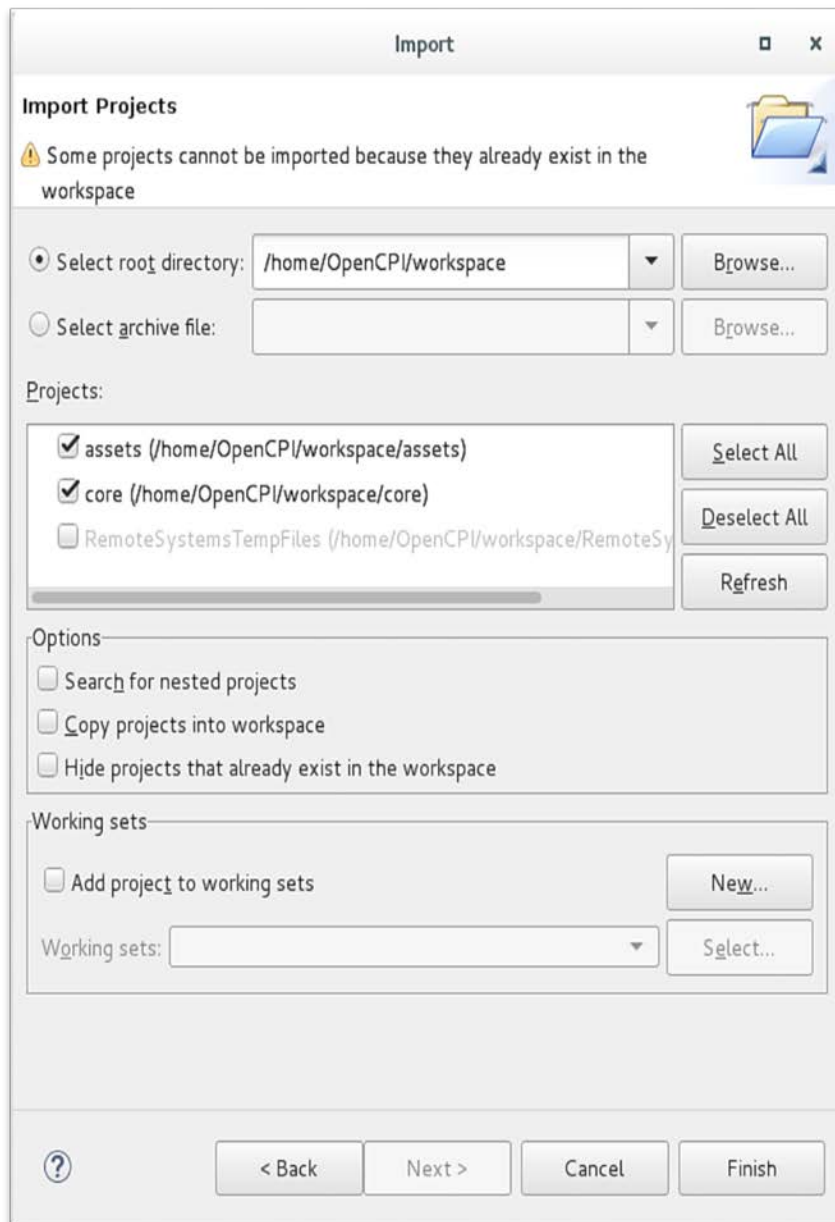


FIGURE 20 PROCEDURES TO USE EXISTING PROJECTS

5.2 1 Importing OpenCPI and Asset Projects

This demonstration depicts importing the core and assets projects.

- Place the cursor in the Eclipse Project Explorer Panel.
- Right Click→Import→General → Existing projects into workspace.
 - This opens the import wizard.
- Click Browse for the Select root directory input
- Navigate to and select the folder holding these projects
- Select the checkboxes on the desired projects to import as shown below
- Click Finish and the projects will be brought into the Eclipse workspace
 - They will appear in the Eclipse Project Explore View
- Click Refresh in the OpenCPI Projects View to update that view with the imported projects
- Import the desired projects into Eclipse

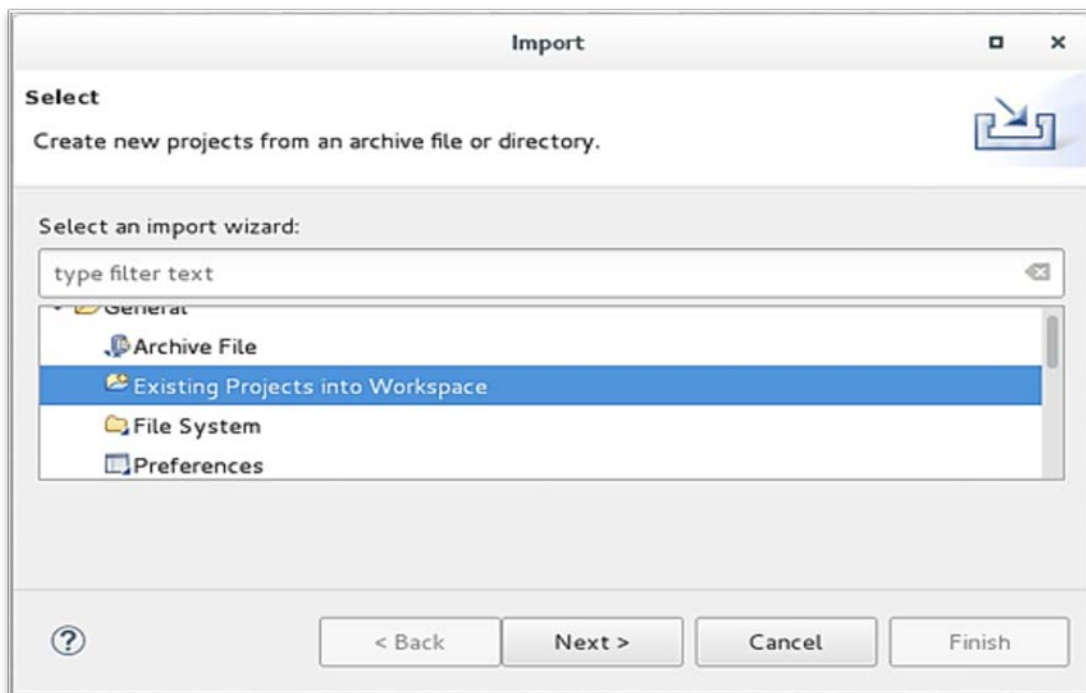


FIGURE 21 IMPORTING OPENCPI AND ASSET PROJECTS

5.2.1 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 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
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.

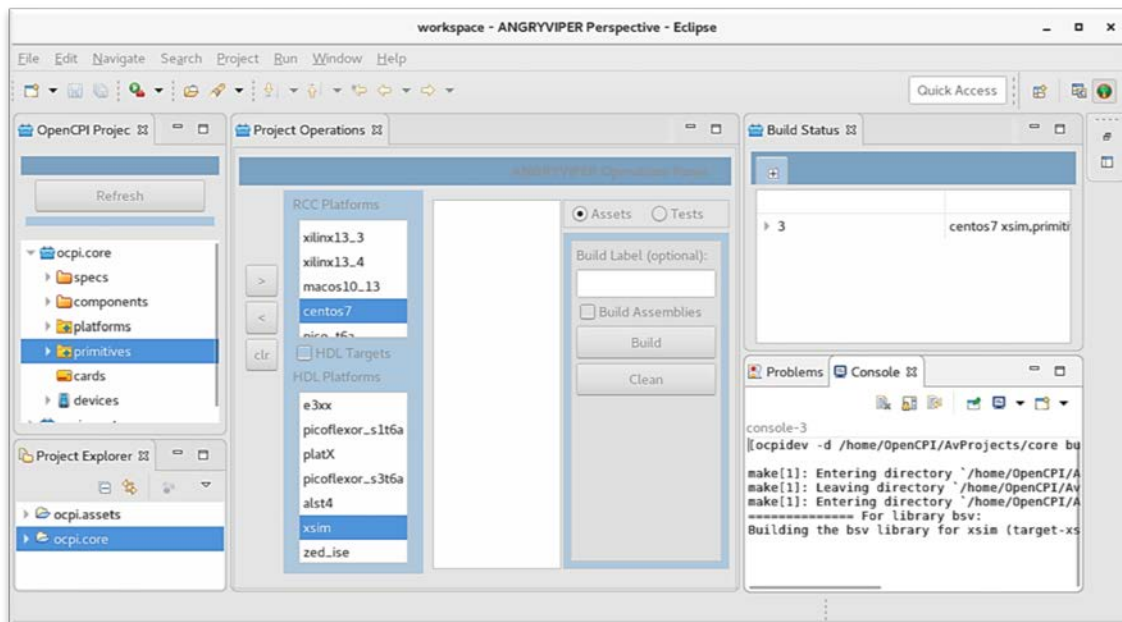


FIGURE 22 BUILD ENTRY

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.

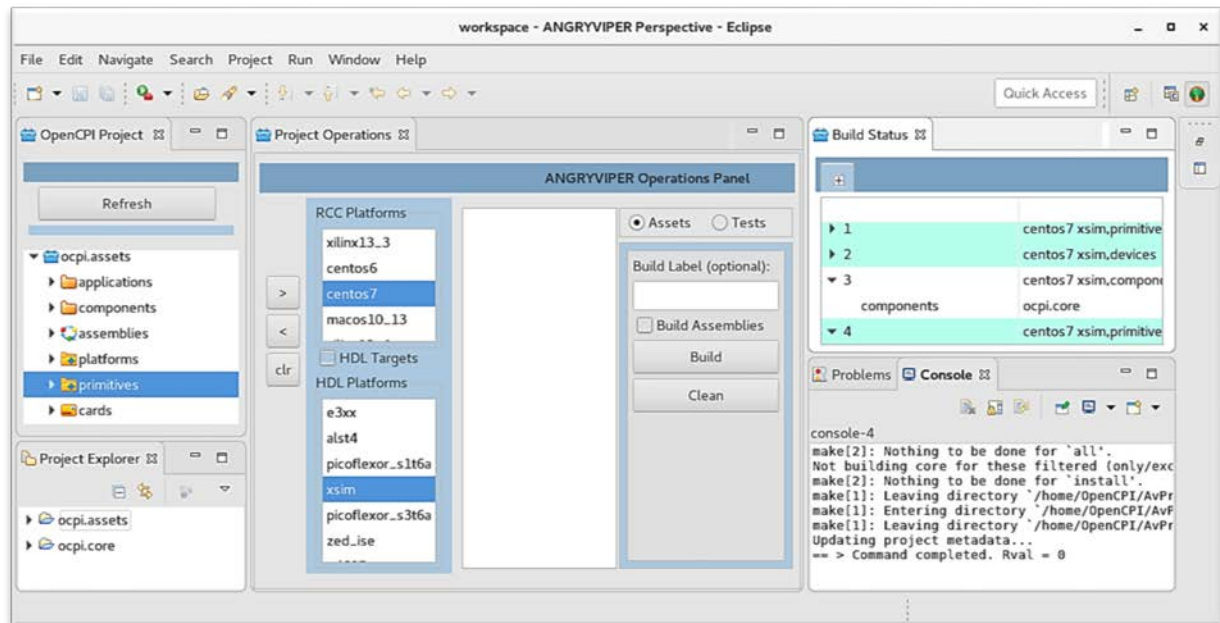


FIGURE 23 PROJECT IS BUILT FROM THE OPERATIONS PANEL

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

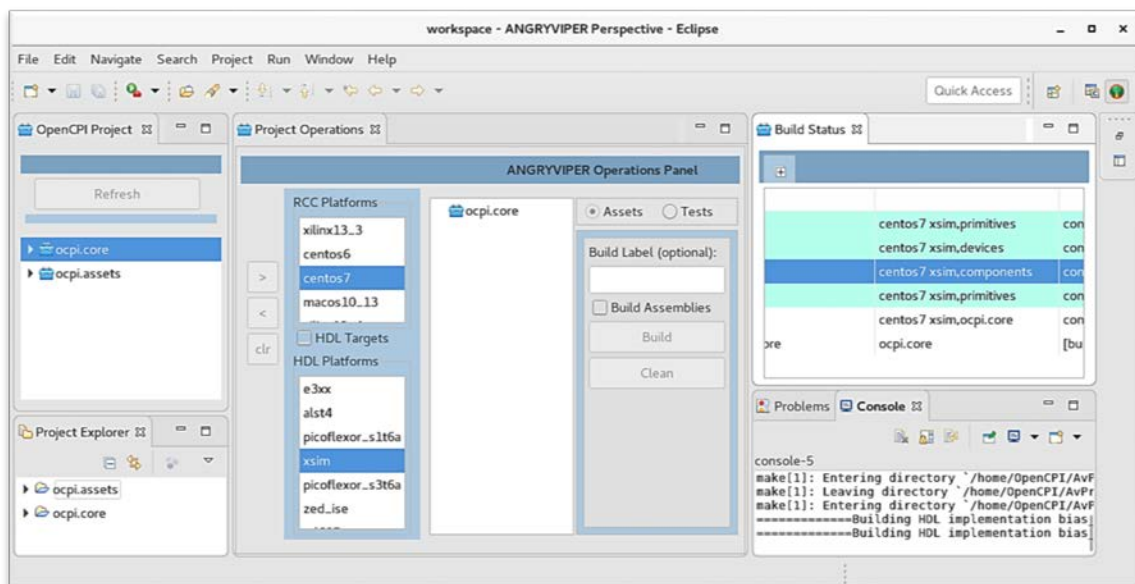


FIGURE 24 CORE PROJECT BUILT FROM THE OPERATIONS PANEL

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

5.3 Creating New Projects and Libraries

To create a new project,

- Place the cursor in the OpenCPI Projects Panel
- 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.

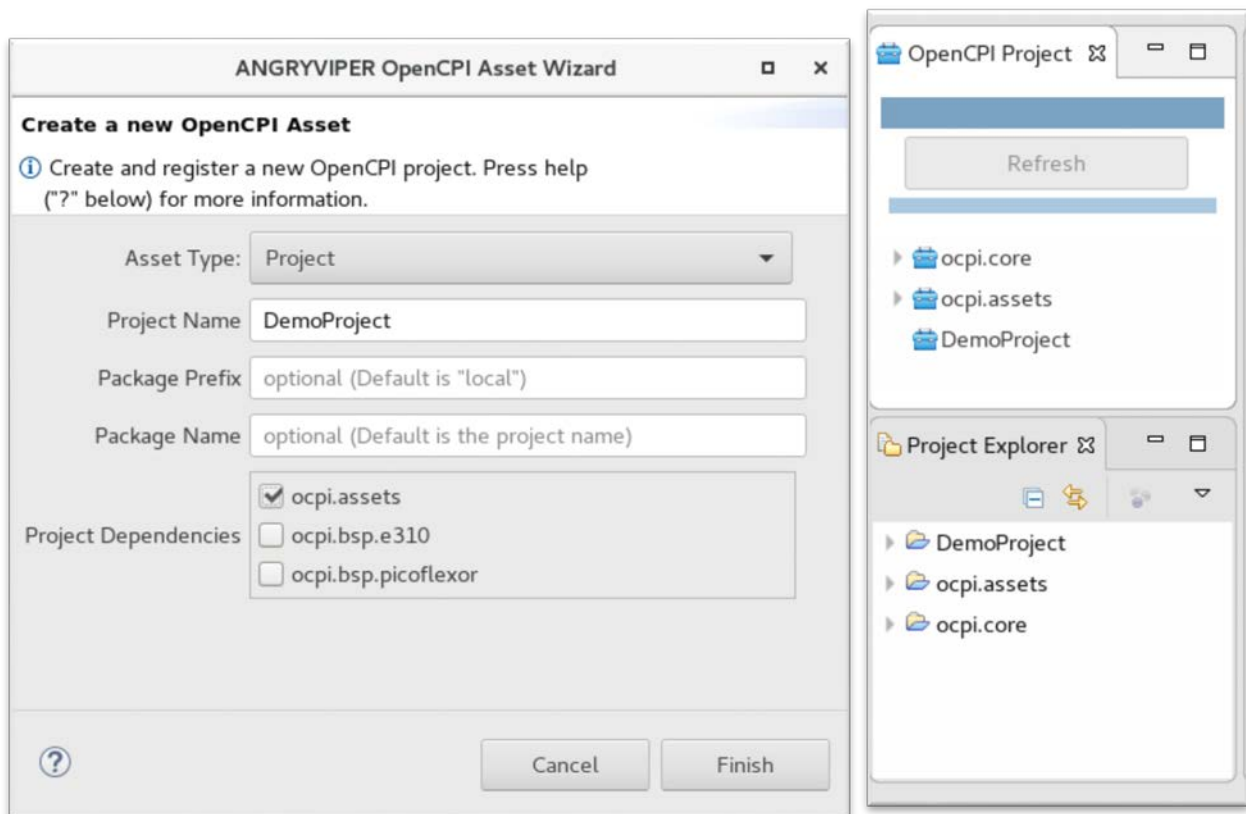


FIGURE 25 CREATE A NEW PROJECT

Next, a components library is added to DemoProject

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

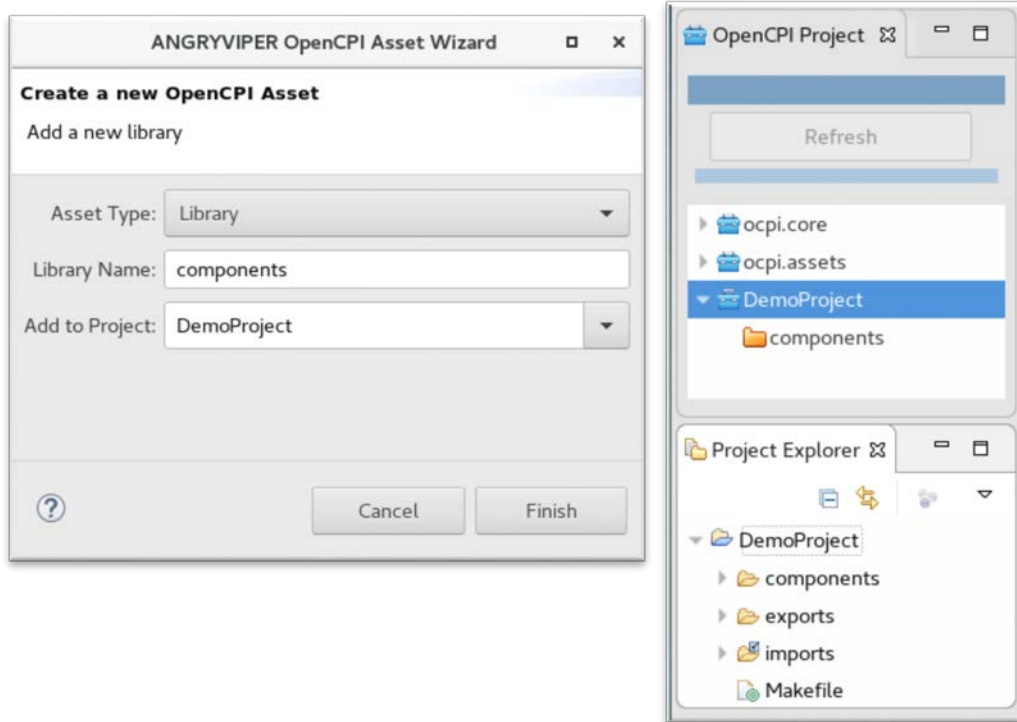


FIGURE 26 ASSET WIZARD

5.3.1 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 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

FIGURE 27 TERMINAL WINDOW VIEW OF A PROJECT REGISTRY

5.3.2 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.

5.3.3 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.

5.3.4 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

5.4 Creating Components, Protocols and Workers

Some fundamentals regarding creating components, protocols, and workers:

- Protocols specify expected data for component Ports
- Protocols and component can exist 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.

5.4.1 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 Right-Click, Select component from the context menu
- Fill in the name as shown below

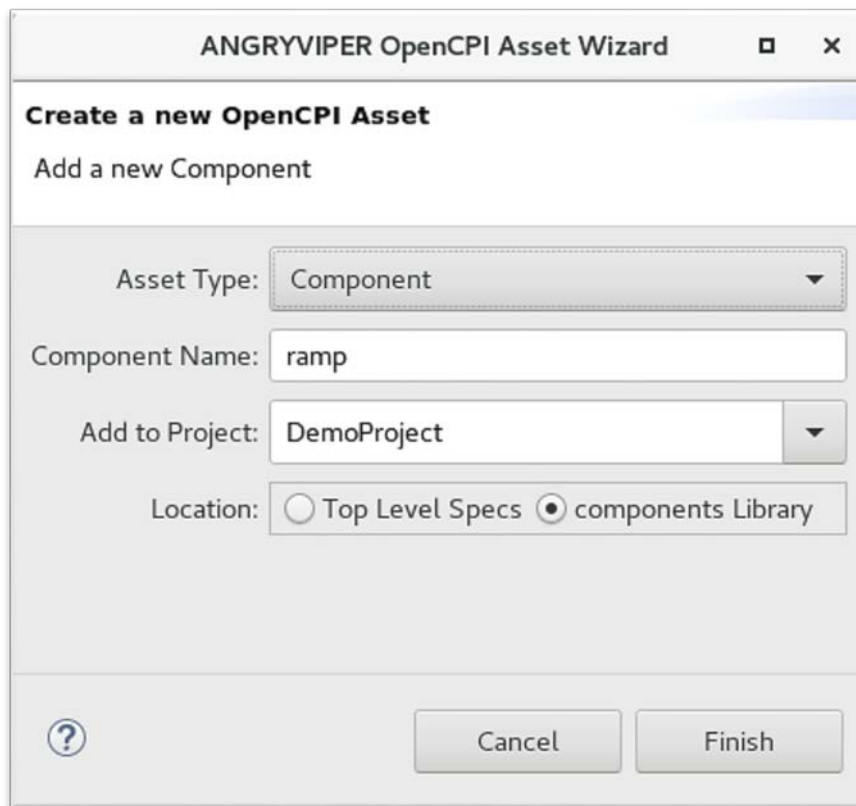


FIGURE 28 CREATE A NEW OPENCPI ASSET

- Once the OCS Editor opens, Select Ports in the outline
- Click the Add a Port link and the Port form appears
- Name the Port “**in**”
- Select restream protocol from the drop-down
- As above, add the “**out**” Port

- The figure below shows the Port form for the ramp “in”

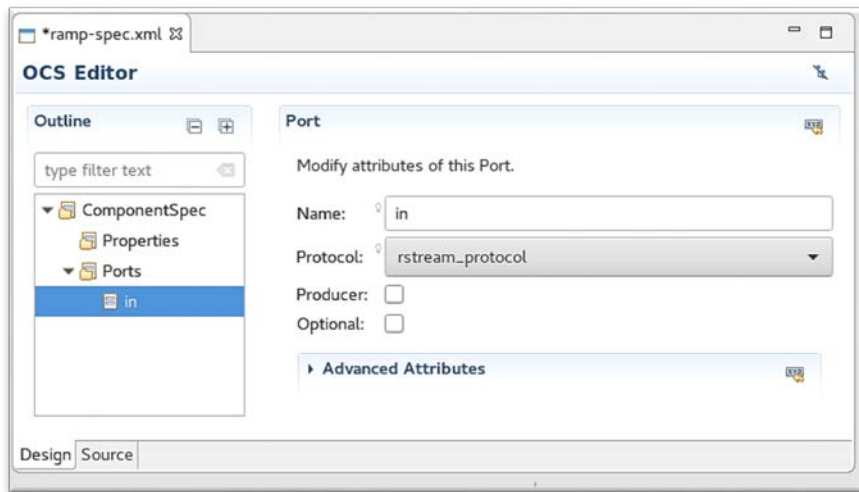


FIGURE 29 PORT FORM FOR THE RAMP IN

5.4.2 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 tool bar at the top)

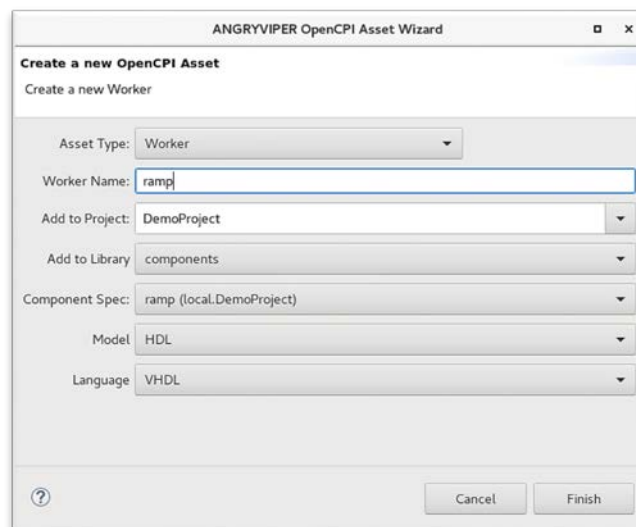


FIGURE 30 WORKER EXAMPLE

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

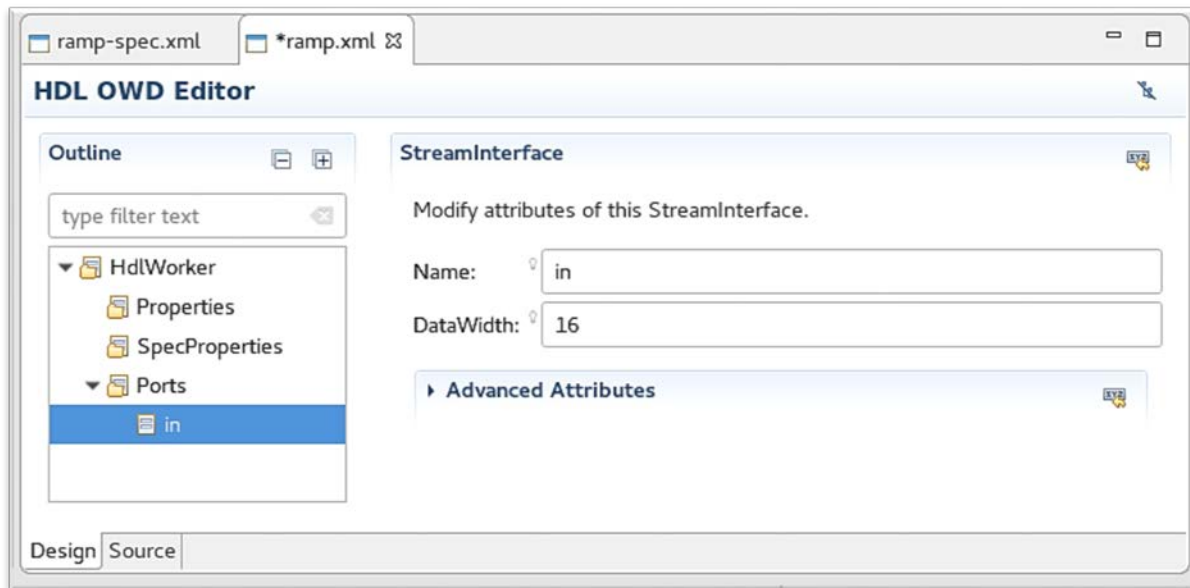


FIGURE 31 ADDITION OF IN AND OUT INTERFACES

5.5 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:

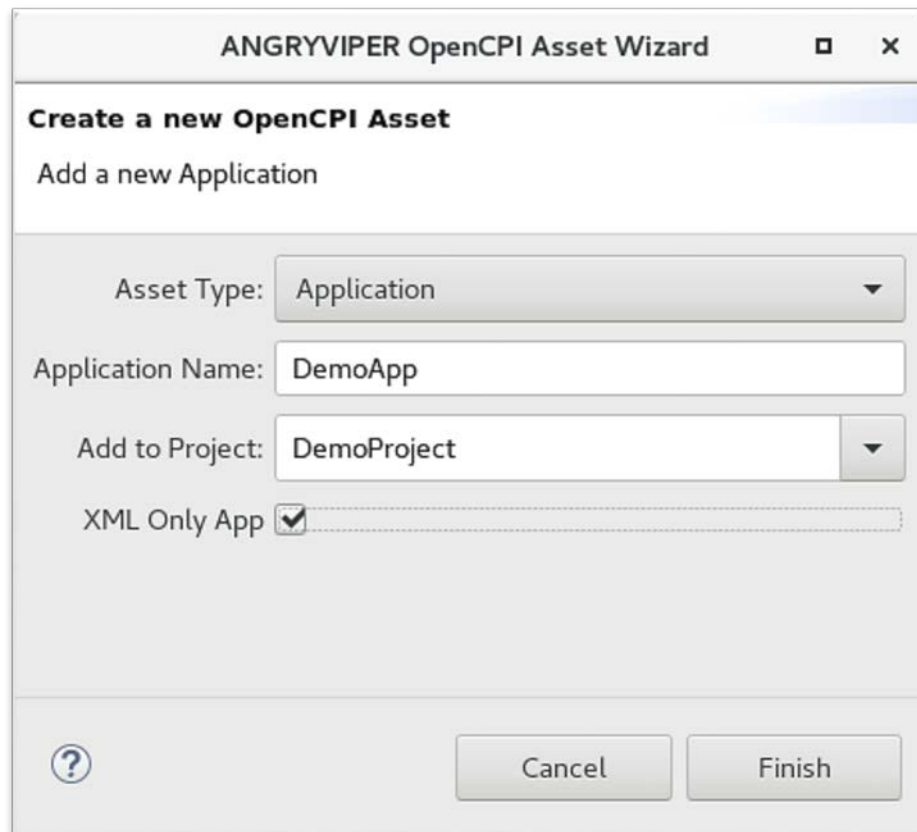


FIGURE 32 ASSET WIZARD

Click Finish and the Application Editor opens:

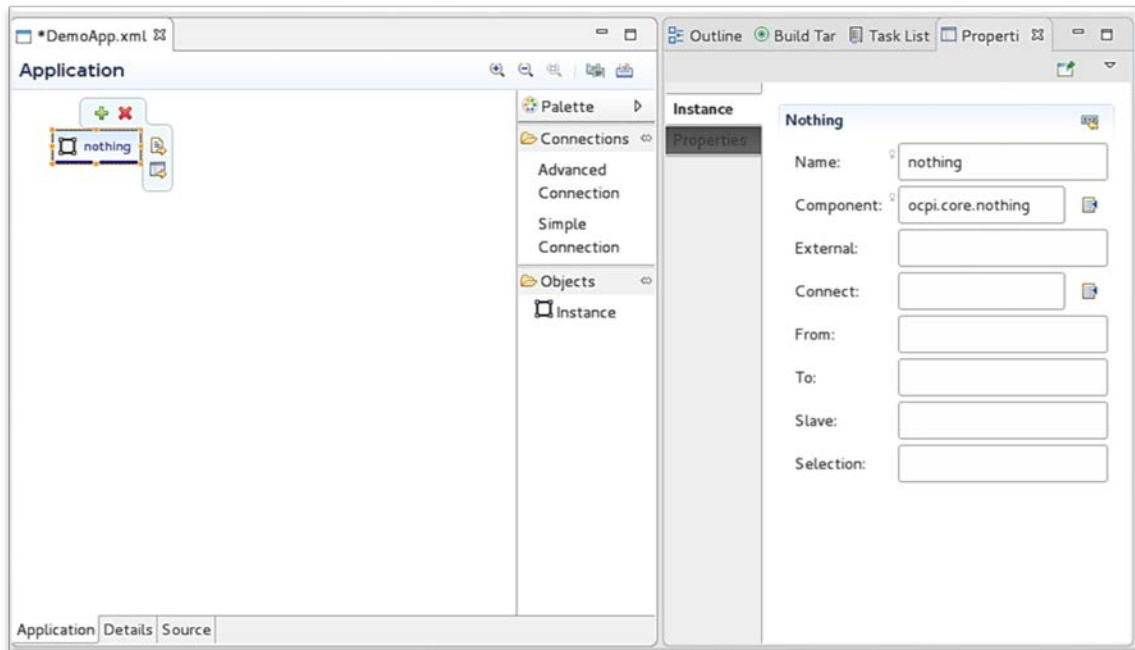


FIGURE 33 APPLICATION EDITOR

5.5.1 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.

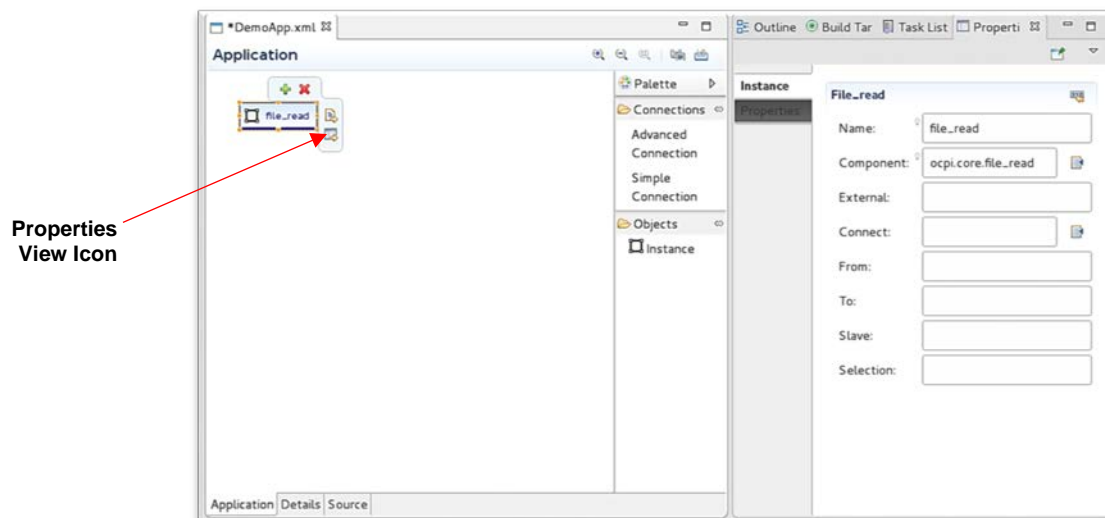


FIGURE 34 PROPERTIES VIEW

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 text box. 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.

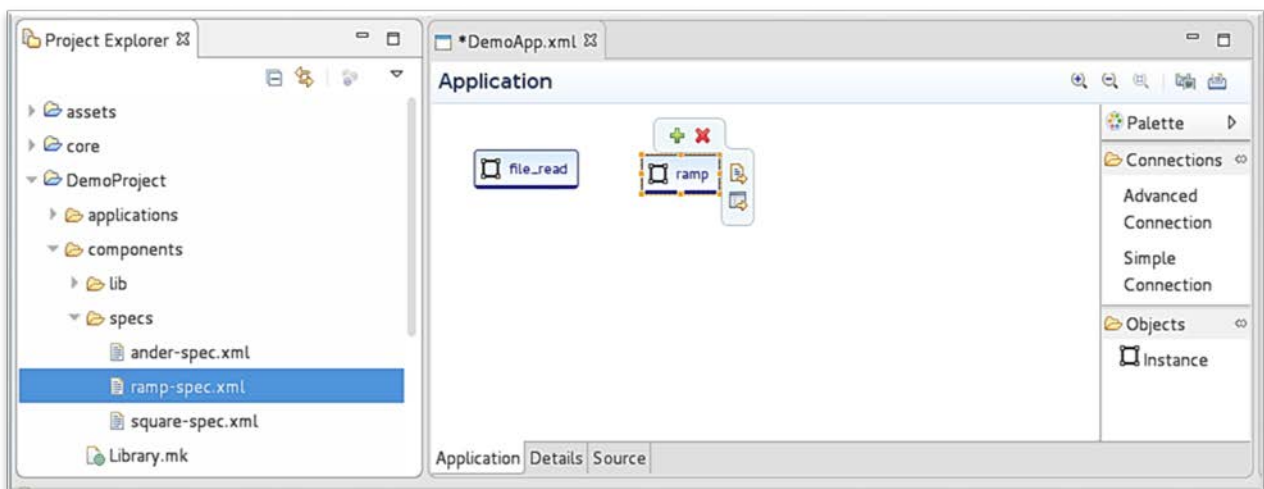


FIGURE 35 ADDING THE RAMP COMPONENT

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

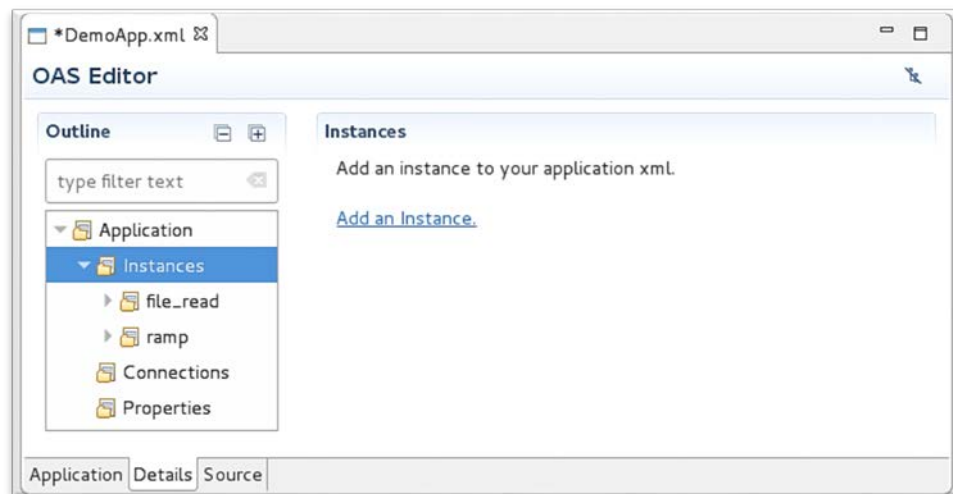


FIGURE 36 THIRD METHOD TO ADD A COMPONENT INSTANCE

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.

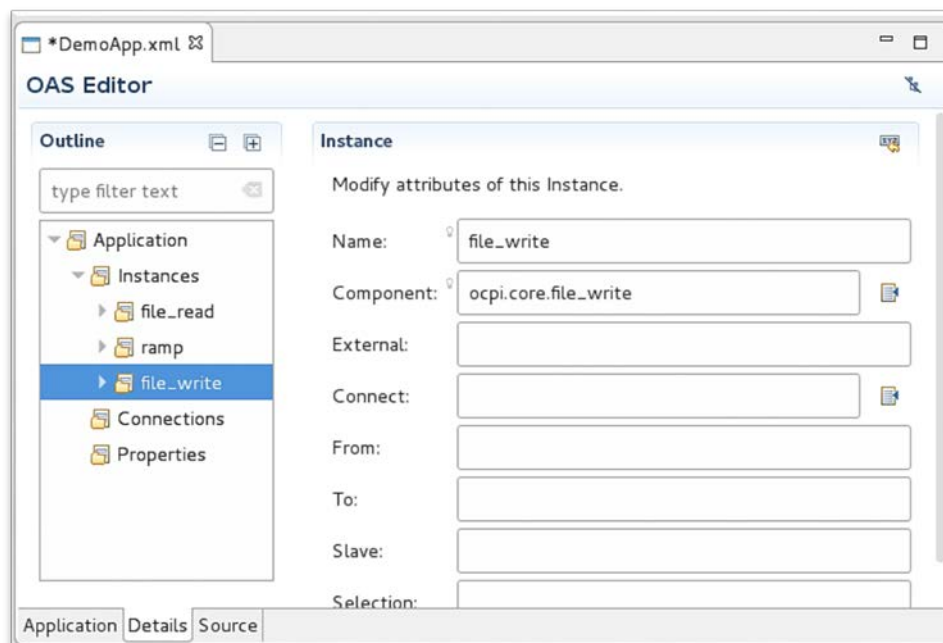


FIGURE 37 SELECT FILE WRITE AS THE NEW COMPANION

Click the List icon next to the Component text box input to the components list. The **file write** component was selected. Figure 40 shows the current design view of the application.

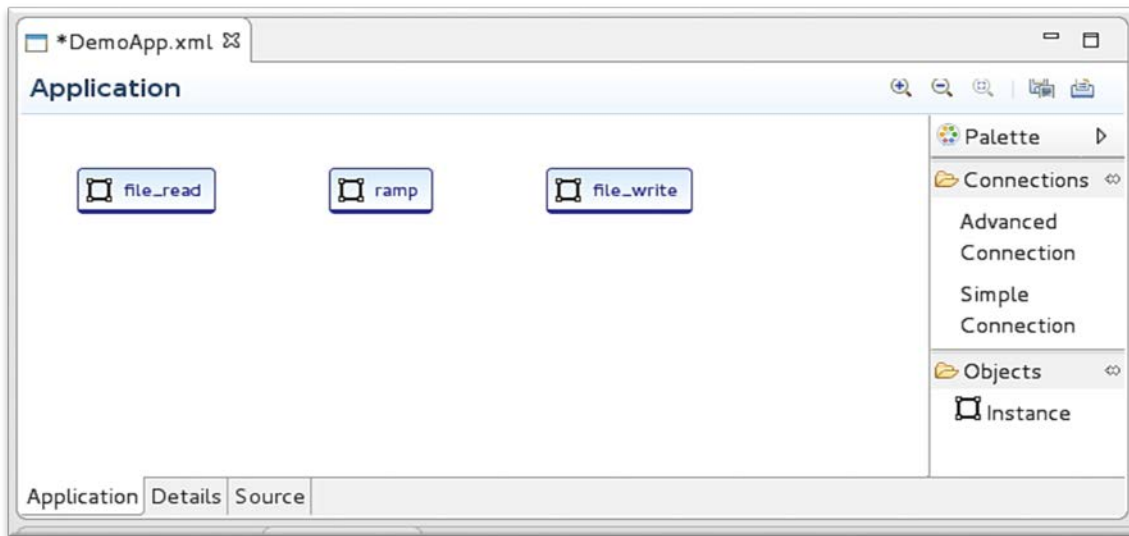


FIGURE 38 DESIGN VIEW OF THE APPLICATION

5.5.2 Connecting Application Components

There are two ways to specify a connection between components in an application or assembly. Components with single input and output Ports may be connected by a simple connection. Components with multiple Ports on a given interface (in or out) must be connected by an Advanced Connection (the XML for these must specify the name of the Port that is being connected).

The tools available in the Application Editor are:

- Select simple or advanced connection from the palette (behaves like a button)
- Move the cursor to the source component
- Click while on it, and then move the cursor to the receiving component
- Click again while on it
- Using the Details Tab, add a simple connection by Selecting the instance then fill in the "Connect" input
 - Complex connects must be added by Clicking Connections in the outline
- Click the Add a Connection link
- Open the connection element and Click Ports
- Click add Port
 - Provide the Port name and instance it connects as shown in Figure 42 below
- Edit the XML source directly

The following Figures demonstrate a simple connection:

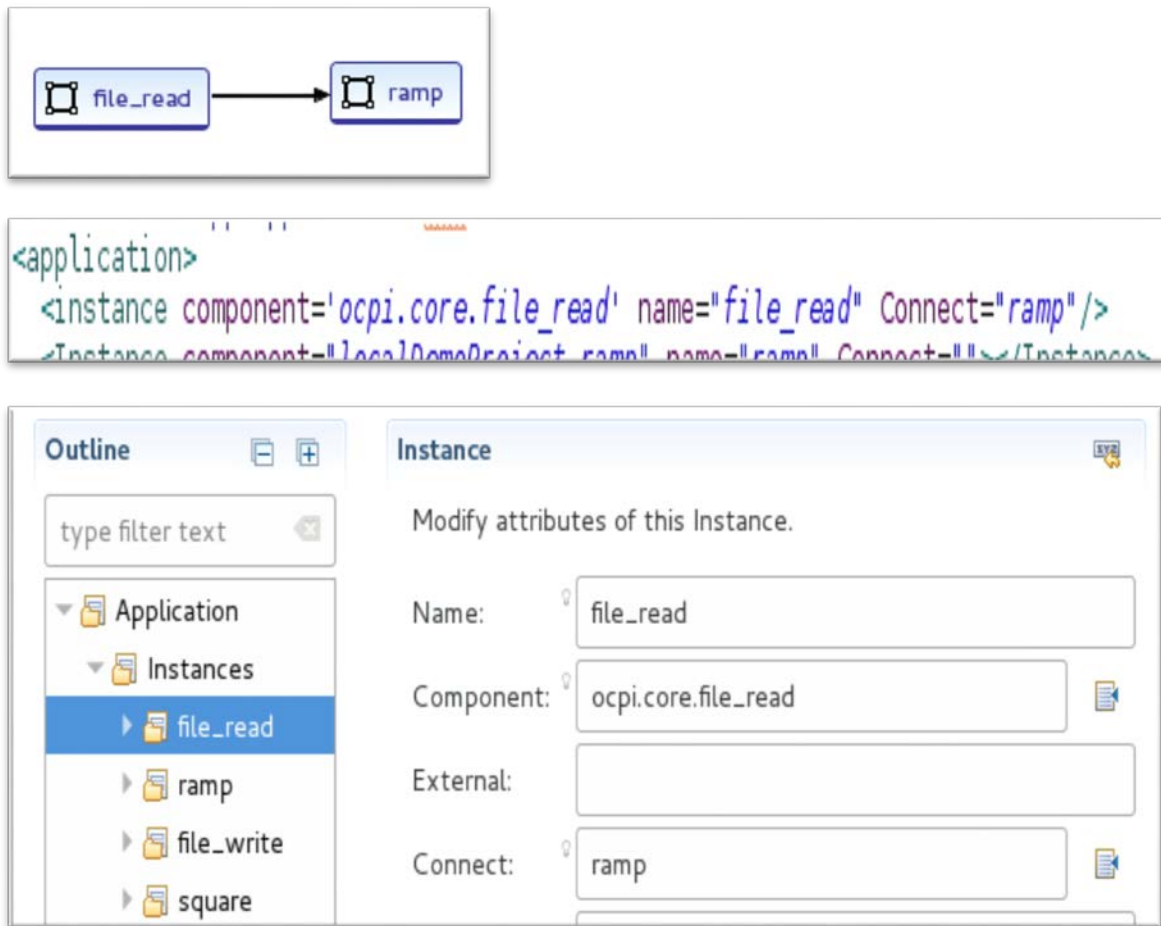


FIGURE 39 EXAMPLE OF A SIMPLE CONNECTION

Advanced connections are easily created using the Application Tab. To make an advanced connection while in 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

A pop-up form panel appears with inputs to complete the connect as shown below in Figure 42.

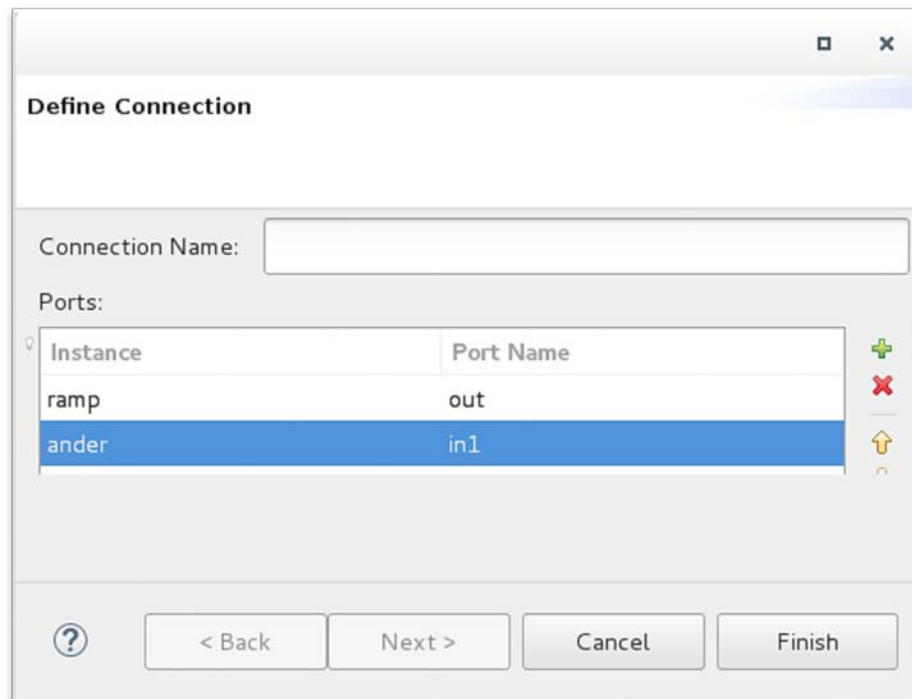


FIGURE 40 POP UP PANEL

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 three Editor views:

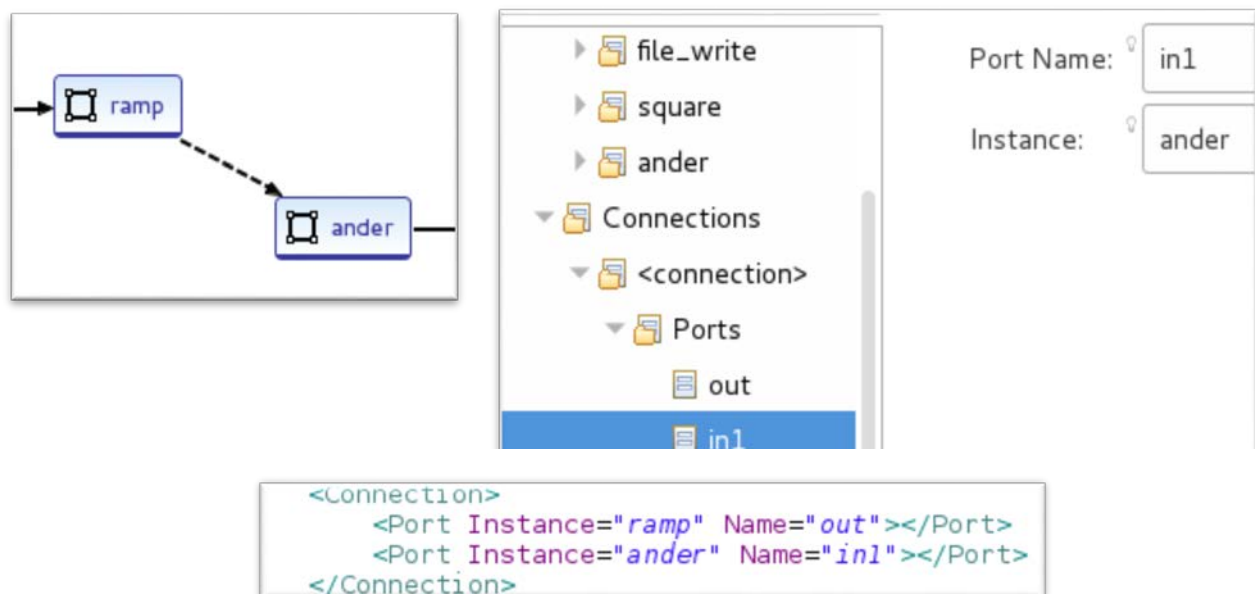


FIGURE 41 ADVANCED CONNECTION VIEWS

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.

5.5.3 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.***

Figure 44 below shows the square.hdl worker added to an assembly via drag-and-drop.

Also note, that worker choices originate from registered OpenCPI projects.

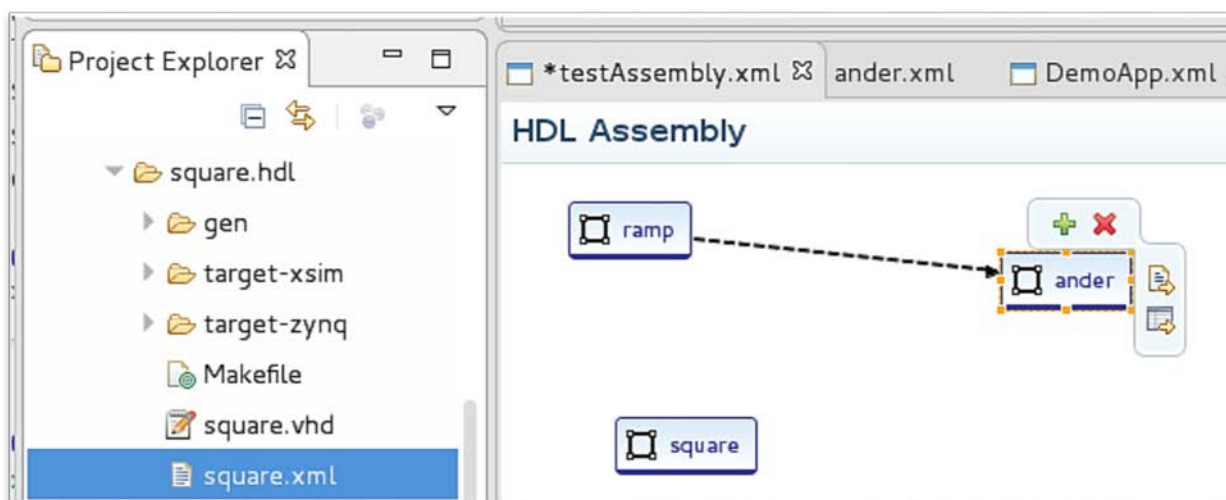


FIGURE 42 SQUARE HDL WORKER

5.6 Creating Component Unit Tests

Creating a unit test using the AV 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
- Click Finish

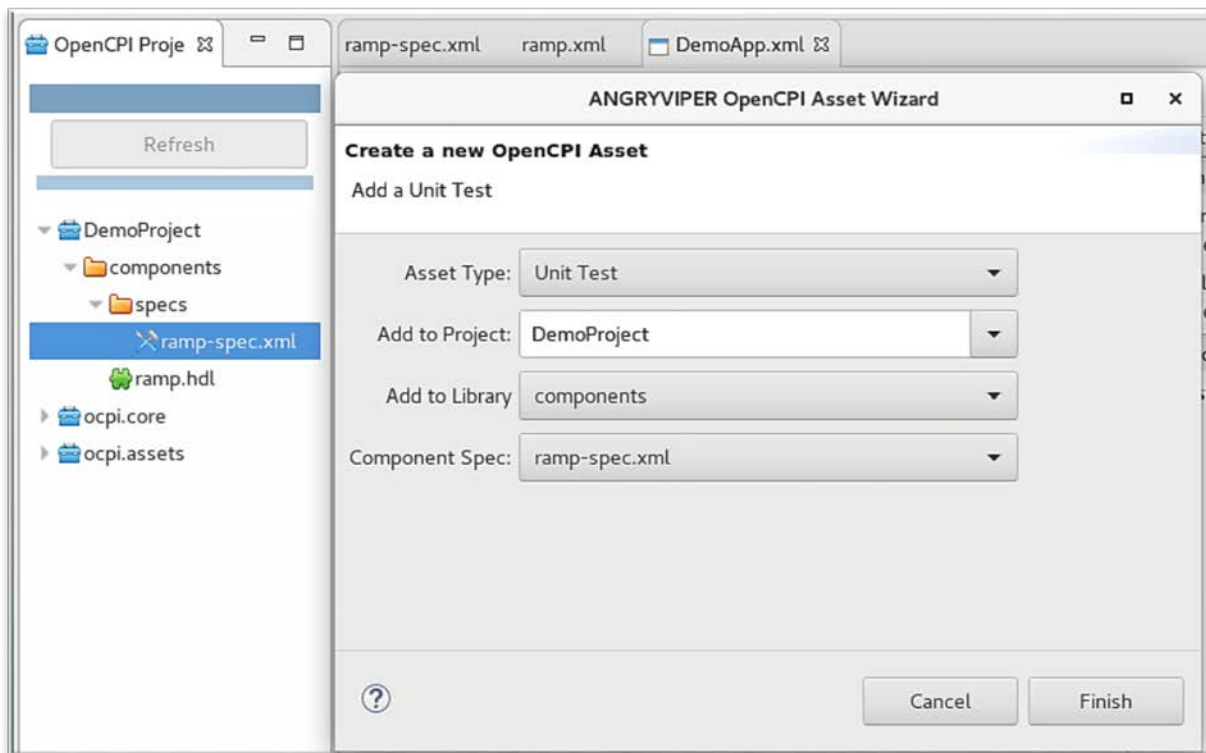


FIGURE 43 ALL INPUTS ARE PRESET FOR THE RAMP SPEC

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 OpenCPI Component Development Guide to learn about them.

The Editor supports most aspects of developing unit test XML.

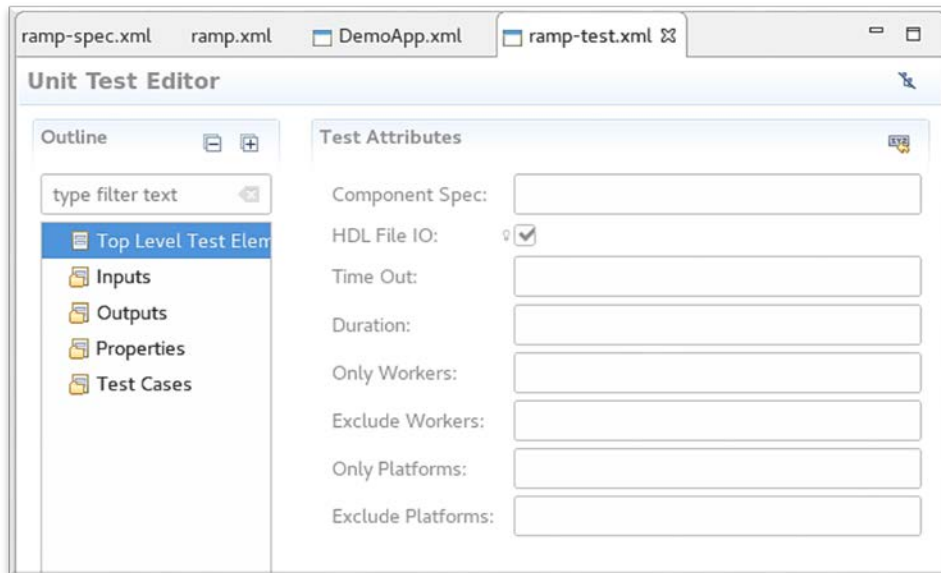


FIGURE 44 UNIT TEST EDITOR PANEL

Note: There are a number of cases where one attribute out of a set may go in their 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).

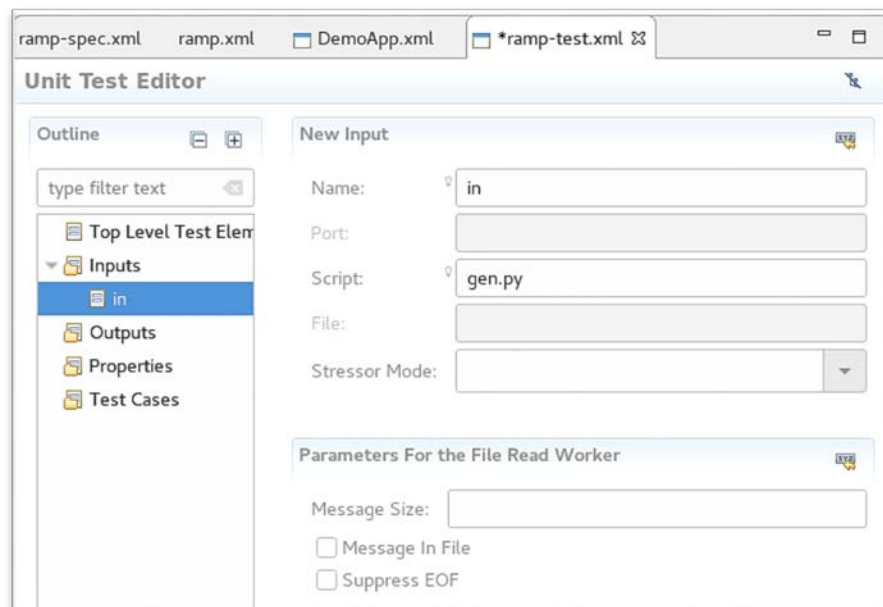


FIGURE 45 DISPLAY OF ATTRIBUTE CHOICE

Appendix A

A.1 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.

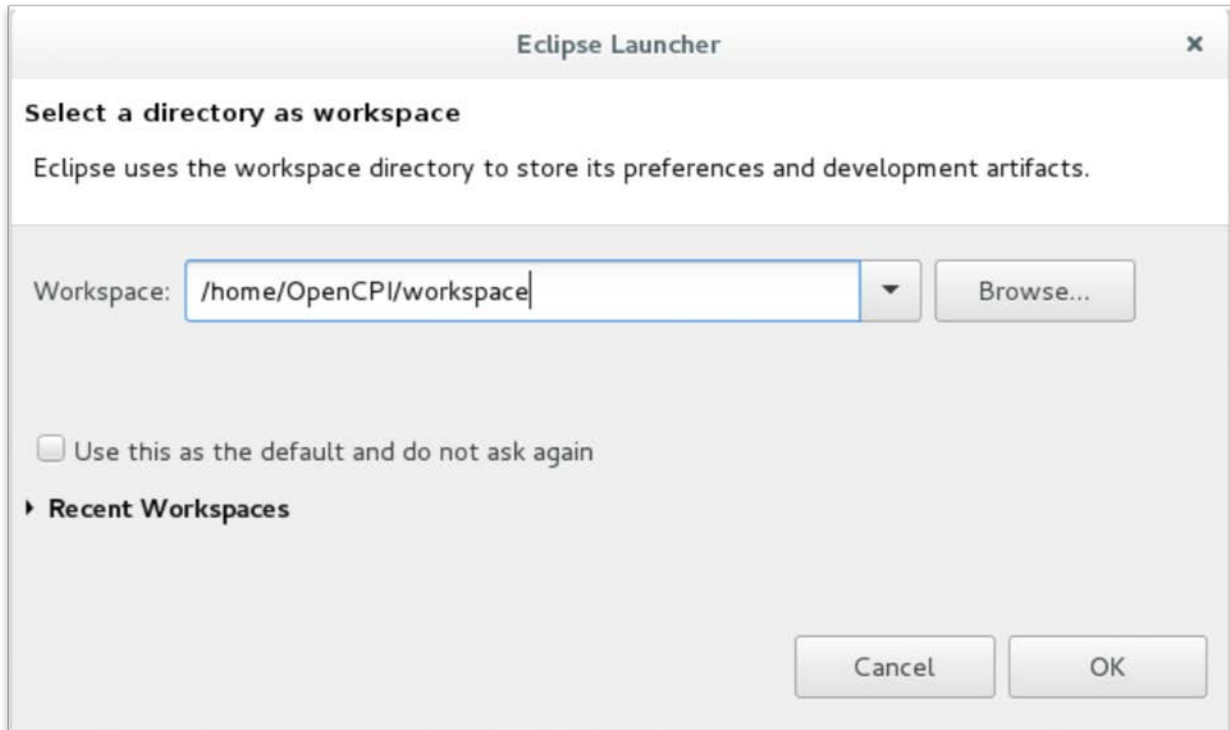


FIGURE 46 ECLIPSE LAUNCHER

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 Tool Bars, Perspectives, and Views. The other selections provide how-to 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.

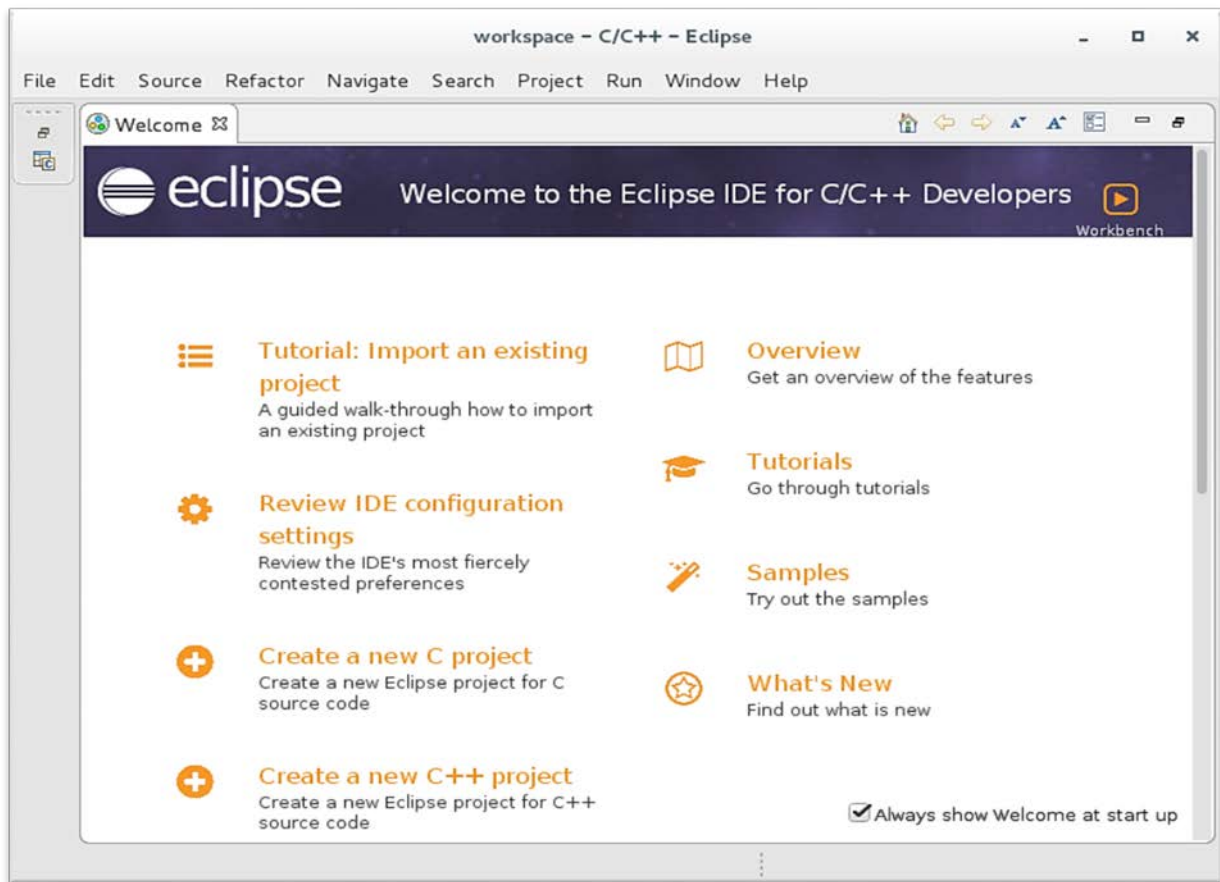


FIGURE 47 ECLIPSE WELCOME SCREEN

When the workbench first opens, the IDE defaults to the C/C++ perspective, as shown in Figure 50 below. Section 3 above explains how to open the AV perspective. How to use the IDE for OpenCPI development is explained in Section 5.

One final 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 appear in the Project Explorer View. OpenCPI projects will not appear in the OpenCPI Projects View unless they are open projects.

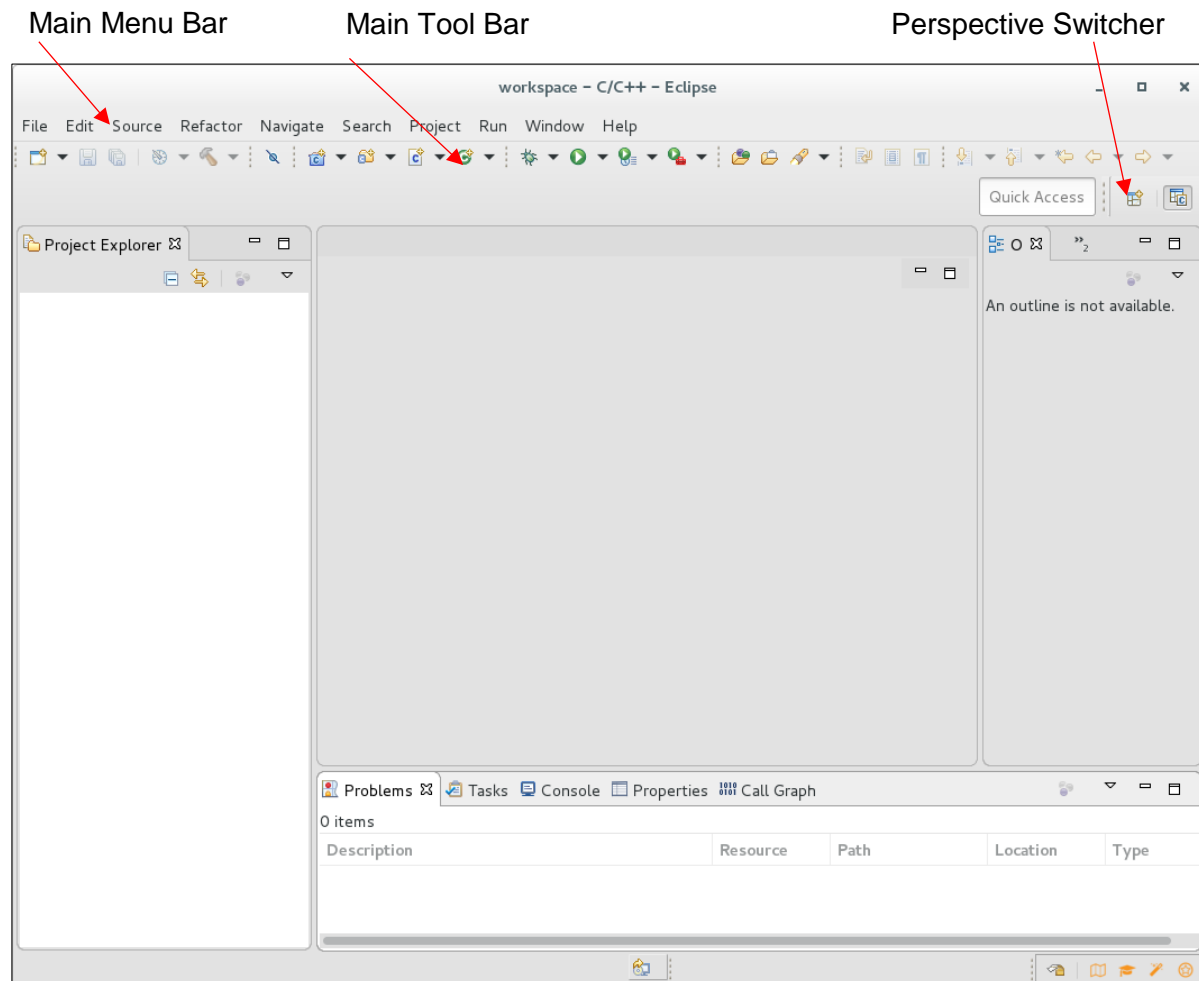


FIGURE 48 ECLIPSE WORKSPACE

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. By default, it will create directory is named workspace in the user's home directory. In this example a directory named “workspace” has already been created in /home/OpenCPI and the core and assets projects are registered there.

A.2 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. One final note: the user can rearrange a perspective layout, add, or remove views, and modify Panel sizes. These settings 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 close 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.

Appendix B

B.1 Addition 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.

B.2 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 excepted to continue.

B.3 ANSI Escape in Console Plugin

While not entirely effective this is the only product available to deal with the ANSI color escape sequences in the Eclipse Console View (seen when running tests). Search for ANSI Escape and install.