Cadena 2.0: nesC Manual

A reference manual for using Cadena to develop nesC/TinyOS applications

Todd Wallentine
# Table of Contents

1. Overview ................................................................................................................................. 1  
   Manual Overview ..................................................................................................................... 1  
   Cadena .................................................................................................................................. 1  
   TinyOS and nesC Overview ........................................................................................................ 2  
2. Preparing for the Tutorial ......................................................................................................... 3  
   TinyOS Plugin Installation .......................................................................................................... 3  
3. Create ..................................................................................................................................... 4  
   Overview .................................................................................................................................. 4  
   New TinyOS Project .................................................................................................................... 4  
      New Project Wizard .................................................................................................................. 5  
   New TinyOS Module ................................................................................................................... 5  
   New TinyOS Interface Type ......................................................................................................... 7  
   New TinyOS Component Type ..................................................................................................... 8  
   New TinyOS Scenario ................................................................................................................. 12  
   New TinyOS Component Instances ........................................................................................... 13  
   New TinyOS Scenario Instances ............................................................................................... 15  
   New TinyOS Connection ............................................................................................................ 16  
   New TinyOS Module from a nesC Interface .............................................................................. 16  
   New TinyOS Module from a nesC Module ................................................................................. 19  
   New TinyOS Scenario from a nesC configuration ..................................................................... 20  
   New TinyOS Module from nesC interfaces and modules .......................................................... 22  
4. Import ..................................................................................................................................... 24  
   Overview .................................................................................................................................. 24  
   Import Interface Type into an Existing Module ........................................................................ 24  
   Import Component Type into an Existing Module ......................................................................... 25  
5. Generate ................................................................................................................................ 27  
   Overview .................................................................................................................................. 27  
   Configure Code Generation ........................................................................................................ 27  
      Turn Auto Generation On/Off ............................................................................................... 27  
      Delete Old Model Artifacts .................................................................................................... 28  
      Source Code Destination ........................................................................................................ 28  
   Generate nesC Configuration .................................................................................................... 28  
      Generate nesC Configuration using Editors Context Menu ................................................. 28  
      Generate nesC Configuration using Navigator Context Menu ............................................ 28  
      Generate nesC Configuration using Toolbar Button ............................................................ 29  
      Generate nesC Configuration using Menubar ...................................................................... 29  
   Generate nesC Module ............................................................................................................. 29  
      Generate nesC Module using Editors Context Menu ........................................................... 29  
      Generate nesC Modules using Editors Context Menu ......................................................... 30  
      Generate nesC Modules using Navigator Context Menu ..................................................... 30  
      Generate nesC Modules using Editors Toolbar Button ........................................................ 30  
      Generate nesC Modules using Menubar ............................................................................... 30  
6. Share ..................................................................................................................................... 31  
   Overview .................................................................................................................................. 31  
   Share using archive: Exporting .................................................................................................. 31  
   Share using archive: Importing ..................................................................................................... 32  
   Share using CVS .......................................................................................................................... 33  
7. Explore ................................................................................................................................... 37  
   Overview .................................................................................................................................. 37  
8. Maintain ................................................................................................................................. 38  
   Overview .................................................................................................................................. 38
List of Figures

1.1. The Cadena meta-modeling language ................................................................. 2
3.1. Eclipse New Project Wizard ............................................................................. 5
3.2. New TinyOS Project Wizard ........................................................................... 5
3.3. New Module Step 1 ......................................................................................... 6
3.4. New Module Step 2 ......................................................................................... 6
3.5. New Module Step 3 ......................................................................................... 7
3.6. New Module Step 5 ......................................................................................... 7
3.7. New nesC Interface Step 1 .............................................................................. 8
3.8. New nesC Interface Step 2 .............................................................................. 8
3.9. New Component Type Step 1 ......................................................................... 9
3.10. New Component Type Step 2 ...................................................................... 9
3.11. New Component Type Step 3 ....................................................................... 10
3.12. New Component Type Step 4 ...................................................................... 10
3.13. New Component Type Step 5 ...................................................................... 11
3.14. Properties view before setting a parameter ................................................. 11
3.15. Properties view after setting a parameter .................................................... 12
3.16. New Scenario Wizard .................................................................................. 13
3.17. Importing Modules to Scenarios ................................................................. 13
3.18. Scenario Overview ....................................................................................... 14
3.19. Component Instance Wizard ..................................................................... 14
3.20. New Scenario Instance Wizard ................................................................ 15
3.21. New Scenario Instance Wizard ................................................................ 16
3.22. Import Interface Wizard: Select New Module Location ................................ 17
3.23. Import Interface Wizard: Select Interface File .......................................... 18
3.24. Import Interface Wizard: Confirmation Page ............................................... 19
3.25. Import nesC Module .................................................................................... 20
3.26. New Cadena Scenario .................................................................................. 21
3.27. Select Configuration File ............................................................................. 21
3.28. Scenario Imports ......................................................................................... 22
3.29. Missing Imports .......................................................................................... 23
4.1. nesC Interface File Import Dialog: Empty .................................................... 24
4.2. nesC Interface File Import Dialog: StdControl ........................................... 25
4.3. nesC Module File Import Dialog: Empty ...................................................... 25
4.4. nesC Module File Import Dialog: TimerM .................................................... 26
5.1. nesC Code Generation Project Properties for Surge .................................... 27
6.1. Export ............................................................................................................. 31
6.2. Export Wizard Page 2 .................................................................................... 32
6.3. Import ............................................................................................................ 32
6.4. Import Wizard Page1 ..................................................................................... 33
6.5. Import Wizard Page2 ..................................................................................... 33
6.6. Open Perspective .......................................................................................... 34
6.7. Select a Perspective ....................................................................................... 34
6.8. Add new location .......................................................................................... 35
6.9. New CVS location ......................................................................................... 35
6.10. Sharing your project .................................................................................... 36
Chapter 1. Overview

Manual Overview

The Cadena 2.0: nesC Manual was created as a complete reference manual for the Cadena/TinyOS development environment. It has a feature-centric focus meaning that it describes features and how they are used. This is slightly different than the Cadena 2.0: nesC Tutorial which provides a task-centric focus meaning that it describes tasks and how they can be accomplished using features available in the Cadena/TinyOS development environment.

This manual starts with this overview which includes a little background information as well as some pointers to more details about those topics. It then continues with directions to prepare you for the features documented in this manual. After that, there are several chapters that provide details about the features available in the Cadena/TinyOS development environment. Each chapter focuses on a loose collection of features. Each section in a chapter provides details on a specific feature available.

Cadena

Cadena is an Eclipse-based extensible integrated modeling and development framework for component-based systems. Cadena’s models are type-centric in that multi-level type systems are used to specify and enforce a variety of architectural constraints relevant to development of large-scale systems and software product lines.

Cadena provides the following capabilities to system architects, infrastructure developers, and system developers:

- Define modeling environments for widely-used component models: Cadena’s meta-modeling capabilities can be used to formally capture the definition of widely used component models such as the CORBA Component Model (CCM), Enterprise Java Beans (EJB), and nesC (a component model for sensor networks built on TinyOS). Meta-models can include attributes that represent settings and parameters for underlying middleware frameworks on which systems will be deployed.

- Define domain-specific component models: Cadena meta-modeling can also be applied to specify new component models, including domain-specific component models that are tailored to the characteristics of a particular domain or underlying middleware capabilities.

- Flexibly combine and extend multiple component models in a single system: Cadena meta-models (called styles) can be directly manipulated using style operations. This provides a variety of powerful and useful capabilities to system architects.

- Styles can be extended through inheritance. This enables reuse of meta-model definitions, and facilities refinement of platform definitions (multi-step platform-independent to platform-specific model refinement).

- Multiple styles can be combined within the same architecture model environment to support development of systems of systems that incorporate multiple component models.

- Define end-to-end model-driven development environments: Cadena’s base set of capabilities can be extended using plug-in mechanisms based on the Eclipse plug-in architecture. This enables infrastructure developers to build end-to-end model-driven development environments that include facilities for editing component implementations, model-level configuration of middleware capabilities, code generation, simulation, verification, and creating system builds. Plug-ins can also be developed.
to link other development tools including tools for requirements capture and down-stream class-level modeling tools such as Rational Rose or Modeler or iLogix Rhapsody.

Figure 1.1. The Cadena meta-modeling language

TinyOS and nesC Overview

TinyOS is "an open-source operating system designed for wireless embedded sensor networks. It features a component-based architecture which enables rapid innovation and implementation while minimizing code size as required by the severe memory constraints inherent in sensor networks" [TinyOS:URL].

nesC is "an extension to the C programming language designed to embody the structuring concepts and execution model of TinyOS" [nesC:URL].

The Cadena team chose to develop plugins to support end-to-end development for TinyOS/nesC for many reasons. The first is as an example of what can be done using Cadena. The second is as a testbed for our research ideas. The final reason is to help support a team of developers at K-State that are currently experimenting with sensor network technologies and applying them in some unique ways. So we decided that developing the tools in Cadena would be very helpful to them and allow them to experiment in more efficient ways (by using product-line development technologies).
Chapter 2. Preparing for the Tutorial

This manual assumes your computer meets the system requirements, has the prerequisite software, and has Cadena with the TinyOS plugins installed. For more details about the system requirements, prerequisite software, and installing Cadena, see the Cadena 2.0: Install Guide. Details on installing the TinyOS plugins are provided below.

TinyOS Plugin Installation

To use this manual, you must have the TinyOS plugins installed into your Eclipse/Cadena environment. We assume you have read the Cadena 2.0: Install Guide and have installed the prerequisite software and Cadena on a system that meets the system requirements. We also assume you know how to install plugins using the Eclipse Update Manager.

This tutorial relies upon the following plugins:

1. edu.ksu.cis.cadena.platform.tinyos
2. edu.ksu.cis.cadena.platform.tinyos.parser

Be sure to install those using the Eclipse Update Manager. For more information on this you can see the Cadena 2.0: Install Guide and the Eclipse web site [Eclipse:URL].
Chapter 3. Create

Overview

This chapter provides details about features that involve the creation of artifacts. This includes tasks related to creating projects, creating files, and creating models. Each section in this chapter is a different feature that will help you create Cadena artifacts.

The sections in this chapter are:

- New TinyOS Project: Create a new TinyOS Project using the Wizard
- New TinyOS Module: Create a new TinyOS Module using the Wizard
- New TinyOS Interface Type: Create a new TinyOS Interface Type using the Wizard
- New TinyOS Component Type: Create a new TinyOS Component Type using the Wizard
- New TinyOS Scenario: Create a new TinyOS Scenario using the Wizard
- New TinyOS Component Instance: Create a new TinyOS Component Instance using the Wizard
- New TinyOS Scenario Instance: Create a new TinyOS Scenario Instance using the Wizard
- New TinyOS Connection: Create a new TinyOS Connection
- New TinyOS Module from a nesC Interface: Create a new TinyOS Module from a nesC Interface using the Wizard
- New TinyOS Module from a nesC Module: Create a new TinyOS Module from a nesC Module using the Wizard
- New TinyOS Scenario from a nesC Configuration: Create a new TinyOS Scenario from a nesC Configuration using the Wizard
- New TinyOS Module from nesC Interfaces and Modules: Create a new TinyOS Module from nesC Interfaces and Modules using the Wizard

Note: There is no ordering to the sections.

New TinyOS Project

One of the first tasks when developing nesC/TinyOS applications in Eclipse is creating a new project. This is made easier through a Wizard for creating a new TinyOS project. This section walks you through the use of that Wizard.

The new TinyOS Project Wizard provides a step-by-step guide to creating a project that will be used for TinyOS application development. The Wizard makes it so easy that you simply have to provide a name and the rest is done for you.

Naming a project has the same restrictions as other Eclipse projects.
New Project Wizard

The first step in creating a new TinyOS project is to start the New Project Wizard dialog. This is done in many ways. The easiest way is using the main menu by selecting "File" | "New" | "Project". This will result in a new dialog being shown that looks like Figure 3.1, “Eclipse New Project Wizard”.

Figure 3.1. Eclipse New Project Wizard

Select the "TinyOS Project" and press "Next". This will result in a page that prompts you for a project name. This can be seen in Figure 3.2, “New TinyOS Project Wizard”. Enter a name and press "Finish".

Figure 3.2. New TinyOS Project Wizard

The end result is that a new Eclipse project will be created in the workspace with the given name. That project will also be set to have the Cadena and TinyOS project natures. Finally, it will create the default directory structure for Cadena projects and set up the Cadena specification path appropriately.

New TinyOS Module

Here you will learn how to create a new Cadena Module using the nesC style. This section assumes you have a nesC/TinyOS project created.
Right click on your open project and go to "New->Other" as seen in Figure 3.3, “New Module Step 1”. This can also be done by pushing "Ctrl-N" or going to “File->New->Other...”.

Figure 3.3. New Module Step 1

Under the Cadena Folder click on "Cadena Module" and press "Next". This can be seen in Figure 3.4, “New Module Step 2”.

Figure 3.4. New Module Step 2

This will bring to you the Cadena Module Wizard. First type a name of your new module. At the moment the module is not associated with any style. Click the "Browse" button under style and select "http://Cadena.projects.cis.ksu.edu/tinyos/nesC.style". This process is illustrated in Figure 3.5, “New Module Step 3”. Now Press "OK".
Your nesC module is now created and should open automatically. It is added to your nesC/TinyOS project as shown in Figure 3.6, “New Module Step 5”. (You can also open it by clicking on your name.module file)

Figure 3.6. New Module Step 5

New TinyOS Interface Type

This section teaches you how to make an "Interface Type". Interfaces are used to define how components will eventually connect together. This section assumes you have an open nesC/TinyOS project and a Cadena module that you can edit.

Open your module file and switch to the table view. To switch to the table view click on the “Table” tab on the lower left hand side of the "Module Overview".

In the Module Detail View there are two different sections. The left side deals with Component Types and the right side deals with Interface Types. Right click under the "Interface Types" section and select "Add Interface Type->NesCInterface". This step can be seen in Figure 3.7, “New nesC Interface Step 1".
Create

Figure 3.7. New nesC Interface Step 1

The New Kind Wizard is now open and all you have to do is enter a name. Type in "Leds" and press enter. This step can be seen in Figure 3.8, "New nesC Interface Step 2".

Figure 3.8. New nesC Interface Step 2

A new interface should now be created. It will show up in the "Interface Types" table.

New TinyOS Component Type

This section will teach you how to create a new Component Type. It will go on to teach you how to add ports to a component. Ports are used to connect components together. To be able to add ports to a component you need to have interfaces defined. This is explained in the section called “New TinyOS Interface Type”. At the end of this section you will also find instructions on setting a Port's parameter (or creating a parameterized-interface-instance).
First open up the table view in the module file. Then right click on the "Component Types" side and click "Add Component Type-> NesCComponent". This step can be seen in Figure 3.9, “New Component Type Step 1”.

Figure 3.9. New Component Type Step 1

This opens up a new component wizard. Name the component "BlinkM". You can also see under "Kind" that it is a "NesCComponent". You can see this step in Figure 3.10, “New Component Type Step 2”. When you are done press "Finish".

Figure 3.10. New Component Type Step 2

Now your nesC Component Type has been created! You will see it appear in the "Component Types" box. Now with your "Component Type" created you can add ports to it. Remember you need some interfaces created before you can create ports.

Highlight the "BlinkM" component type and right click. Go to "Add Port->provides". This step can be seen in Figure 3.11, “New Component Type Step 3”
Figure 3.11. New Component Type Step 3

This will open the Add Port wizard. Under "Interface" click browse and click "Leds" and click "Ok". Under "Name" type "Leds". Then click "Finish". This step can be seen in Figure 3.12, "New Component Type Step 4".

Figure 3.12. New Component Type Step 4

When you are done your module file should look like Figure 3.13, “New Component Type Step 5”.

Create
Figure 3.13. New Component Type Step 5

At this point you may need to mark a port as being parameterized. In the nesC Reference Manual (Section 5 "Component Specification" on page 5), they refer to these as interface-parameters and look like the following. interface Timer[uint8_t id] To specify this in Cadena you will need to use the Properties view. The Properties view is an Eclipse-builtin view that shows a table-tree view of properties for the selected object. Cadena uses this view to see the properties of elements in the model. For example, a nesC interface can have operations. The model in Cadena represents these as properties. The same is true for marking interface instances as parameterized.

The begin using the Properties view in Eclipse select the Window -> Show View -> Other from the menus at the top of the Eclipse window. This will result in a dialog being shown that has a tree of views that can be shown. Under General you will find a view named Properties. Select it and it will appear in the current Eclipse perspective.

Now that the Properties view is open, you can select parts of the model and the properties will be shown in that view. In this case, select a Port from the Component Types tree and the properties will be shown. An example of this can be seen in Figure 3.14, “Properties view before setting a parameter”.

Figure 3.14. Properties view before setting a parameter

Once you have the Port selected that will be parameterized, you can start making modifications to the model. Specifically, you will first need to change the hasParameter property from unset to true. To do this, double-click the value side of the table (where it says unset: double click to set). That will bring up the drop-down menu where you can choose true or false. Select true. This denotes your intention to make this a parameterized interface instance. Once that is done you will need to give that parameter a name and a
Create type (both specified as string values). In this case, double-click each and enter the value you wish to set (e.g., the name could be id and the type could be uint8_t). An example of this can be seen in Figure 3.15, “Properties view after setting a parameter”.

**Figure 3.15. Properties view after setting a parameter**

![Properties view](image)

Note: There is currently a bug in the Properties view which forces the user to select something else before the model is updated. So once you set a value, select anything else in the Properties view or anything else in Eclipse. De-selection causes an event so that the model will be updated with your changes.

**New TinyOS Scenario**

This section will teach you how to create a new Scenario using the nesC style. This section assumes you already have an open nesC/TinyOS project.

With your project highlighted, right click and go to "New->Other...". This will open up a window with a list of different wizard you can use. Under the "Cadena" folder click on "Cadena Scenario" and click "Next".

Now the "Cadena Scenario" wizard should be open. Under the "Style" section click the "Browse" button. This should bring up a window with a list of styles. Select "http://Cadena.projects.cis.ksu.edu/tinyos/nesC.style" and click "OK".

Your scenario wizard should look similar to Figure 3.16, “New Scenario Wizard”. Click "Finish". This will create your scenario. You can open it under the "scenario" folder in your project.
New TinyOS Component Instances

This section will teach you how to add a nesC component instance to your Cadena scenario. This section assumes you already have a nesC scenario created, a nesC/TinyOS project open, and a module with some component types created.

Double click to open your scenario file. (You can find it under the "scenario" folder of your project) This will bring you to the "Scenario Overview". The first thing you have to do is add which module files you want to use to your scenario.

To add a module file to your scenario first make sure you are in the "Scenario Overview". Then click the "Add" button under "Imported Modules". This will bring up a window where you can select which modules you want to import. You can see this in Figure 3.17, “Importing Modules to Scenarios”.

Figure 3.17. Importing Modules to Scenarios
Put a check the box by every module file you want to import. (In this case the file "module.module" is selected) When you are finished press "OK". In your "Scenario Overview" you should now see what modules you are importing into your scenario. It will look similar to Figure 3.18, “Scenario Overview”.

**Figure 3.18. Scenario Overview**

![Scenario Overview](image)

You can either add new components into the "Table View" or the "Graph View". To get into these click on the tabs at the lower left hand side of the "Scenario Overview". If you are in the "Table View" right click under the "Instances" section and go to "Add Component Instance -> NesCComponent". If you are in the "Graph View" just right click anywhere and go to ""Add Component Instance -> NesCComponent".

The "Component Instance Wizard" should now be open. Under "Type" click "Browse" and select the component type you want to use and click "OK". This can be seen in Figure 3.19, “Component Instance Wizard”. Now give your component instance a name and select "Finish". Your component instance should now be created. If you are in the "Table View" it will show up under the "Instances" list. If you are in the "Graph View" it will show up visually as a rounded-edge rectangle. (How it is visually depicted will depend on your style)

**Figure 3.19. Component Instance Wizard**

![Component Instance Wizard](image)
New TinyOS Scenario Instances

This section will teach you how to add a Scenario Instance to an existing scenario. A Scenario Instance is basically another scenario represented as a box that you can plug into. This useful so you don't have to re-create all the components within a scenario just to use the functionality.

This section assumes an open nesC/TinyOS project with two cadena scenarios using the nesC style. This section also will use the files from nesC tutorial track1 as an example.

First open the the Scenario you want to add an instance to and Open the "Graph" view. (The tab for the "Graph" view can be found on the lower left hand corner of a scenario's overview page)

Right click anywhere and select "Add scenario instance". This will bring up the "Scenario Instance" wizard. Under the scenario drop down box select the scenario you want to use. Then give your scenario instance a meaningful name. This process is illustrated in Figure 3.20, “New Scenario Instance Wizard". When you are done click "Finish".

Figure 3.20. New Scenario Instance Wizard

Now a scenario has been added. This will look like Figure 3.21, “New Scenario Instance Wizard". You can tell them apart from components by their different apperance. If you have exposed ports in the scenario you can actually connect to the scenario instace and use its functionality.
This section will teach you how to create connections between instances in a Scenario. It will go on to describe how to set parameter values for connections involving parameterized-interface-instances. This section assumes you have a TinyOS project that includes a Scenario with at least two instances.

More information on this topic can be found in the Cadena manual (section 5, sub-section 6 "Connections").

The basic idea is to create a connection between two port instances (each port instance is associated with a single instance - either a scenario or component instance). There are many ways to do this but we will only detail one way here (all 4 methods are similar so once you know one, you will be able to do them all).

To start with, move to the Graph view. Once there, select one of the ports that you wish to connect. After selecting the port, right-click to bring up a context-menu which will include an item for "New Connection for Port" which has a sub-menu item based upon the type of port (it will either be a NesCInterfaceConnector.serverSide or a NesCInterfaceConnection.clientSide). Select that menu item and a new dialog will be shown (Bind Connector Roles). This dialog presents you with half of the connection already specified (it took the selection you made in the graph view as the first half of the connection). You should now select the "empty" side of the connection and use the Binding drop-down menu to select the other half of the connection. Once selected, press Finish and the connection will be created for you.

You have now successfully created your first connection. In some cases, you will now need to set a parameter value for a parameterized-interface-instance connection. To do this, you must switch to the table view. After doing so, find the connection that you wish to set the parameter for and select the side upon which to set it. If you double-click it, the Properties view will be shown.

Once selected, you should double-click to set the value of parameterValue to your specific need. For example, you may want to set the value using the unique function in nesC (unique("Timer")).

This section will teach you how to create a new Module from an nesC interface. This section assumes you already have a nesC/TinyOS project open and have access to a nesC interface file. This section will use a nesC interface file named "leds.nc" but will work with any valid nesC interface file.
First go to "File->Import..." and select "Import nesC Interface File". This will bring you to the first page of the import wizard.

Click the "New" button to specify that you want to create a new module file instead of adding to an existing one. Then type in a meaningful name for your new module. Then click "Browse" and select the "module" folder of the project you want to import the interface to. This step can be seen in Figure 3.22, "Import Interface Wizard: Select New Module Location". Select "OK" then Click "Next".

**Figure 3.22. Import Interface Wizard: Select New Module Location**

Now click the "Browse" button and find the nesC interface file you want to import. When you import your nesC interface file it will populate the commands of your interface in the "Operations" table. This will look like Figure 3.23, “Import Interface Wizard: Select Interface File”. If you accidently select a file that is not a nesC interface the wizard will tell you the error with the file. This is one way that Cadena helps you work with nesC files. When you are done click "Next".
Figure 3.23. Import Interface Wizard: Select Interface File

The next page of the wizard is the "Confirmation Page" and lets you make sure that you are importing the interface that you wanted. This can be seen in Figure 3.24, “Import Interface Wizard: Confirmation Page”. Double check you are importing the correct file. When you are done press "Finish" and your nesC interface will be imported into a new module.
New TinyOS Module from a nesC Module

This section will teach you how to create a new Cadena module file from a nesC module file. This section assumes that you have already imported or created the interface types needed for the nesC module and that you have an open nesC/TinyOS project. This section will use files from "track1" of the nesC tutorial as an example but you can use any nesC module file.

First go to "File->Import..." and select "Import nesC Module File". This will bring you to the first page of the import wizard.

Click the "New" button to specify that you want to create a new module file instead of adding to an existing one. Then type in a meaningful name for your new module. Then click "Browse" and select the "module" folder of the project you want to import the interface to. This step can be seen in Figure 3.22, “Import Interface Wizard: Select New Module Location “. Select "OK". Then Click "Next".

Now click the "Browse" button and find the nesC module file you want to import. When you import your nesC module file it will populate the ports of your module in the "Ports" table. If you accidently select a file that is not a nesC module the wizard will tell you the error with the file. When you are done click "Next".

The next page of the wizard helps you choose what interface types you want to use with your module. You need to have your interface types already defined in a module file. Click the "+" button to open up a window with a list of your different module files in your project. Put a check mark by the modules you want to use and press "OK". This process is illustrated in Figure 3.25, “Import nesC Module “. (This example shows a nesC module file named "BlinkM" being imported. It requires the interface types "StdControl", "Timer", and "Leds" to continue. Because they are already defined in a module file named "blinkModule" we simply import that module and continue )

When you are done press "Next".
The last step is the confirmation page to make sure you are importing the correct nesC module with the interfaces you want. When you are done press "Finished". A new Cadena module with your nesC module imported as a component type will be created.

New TinyOS Scenario from a nesC configuration

This section will teach you how to create a new Cadena scenario from a nesC configuration file. This section assumes you have an open nesC/TinyOS project and that you have the interface and component types that the nesC configuration file uses already defined in a Cadena module. This section will use files from "track1" of the nesC tutorial as an example but you should be able to use any valid nesC configuration file.

First go to "File->Import..." and select "Import nesC configuration File". This will bring you to the first page of the import wizard. (You can also right click on your project or anywhere within the package manager)

The first page of the import wizard asks you to specify wether you will import the nesC configuration file to an existing Cadena scenario file or wether you create a new file. In this case we will create new Cadena scenario. First give your scenario a meaningful name. Then click "Browse" and select the "scenario" folder of the project you want to import the interface to and press "OK". This process is illustrated in Figure 3.25, "Import nesC Module ". When you are finished press "Next".
The next page of the import wizard will ask you to specify your nesC configuration file. Under "File" click the "Browse" button and select your file. The wizard will now populate information about your nesC configuration file in various places. The different components used will show up under the "Components" section. The different connections will show up under "Connections". If you are scenario has any ports they will show up in the "Ports" section. This is illustrated in Figure 3.27, “Select Configuration File”. When you are finished press "Next".

Figure 3.27. Select Configuration File
The next page of the wizard will let you know what types are missing and what needs to be imported. We need to import the Cadena module files that have the component and interface types defined in them. Press the "+" button and put a check mark by the module files that you want to use. This process is illustrated in Figure 3.28, “Scenario Imports”. After you import the correct module files no missing types will remain. When you are finished with this page click "Next".

**Figure 3.28. Scenario Imports**

The last step is the confirmation page to make sure you are importing the correct nesC configuration with the interfaces types you want. When you are done press "Finished". A new Cadena scenario with your nesC configuration will be created.

**New TinyOS Module from nesC interfaces and modules**

This section will teach you how to create a Cadena module from nesC interfaces and modules. This section assumes that you have an open nesC/TinyOS project and access to nesC files. This section will use the files from track1 of the Cadena nesC tutorial. It will show an example of importing a nesC module "blinkM.nc" and its interfaces "stdControl.nc", "leds.nc", and "timer.nc".

First go to "File->Import..." and select "Import types from nesC files" and select "Next".

Click the "New" button to specify that you want to create a new module file instead of adding to an existing one. Then type in a meaningful name for your new module. Then click "Browse" and select the "module" folder of the project you want to import the interface to. This step can be seen in Figure 3.22, “Import Interface Wizard: Select New Module Location ”. Select "OK". Then Click "Next".

This next page of the wizard is the section where you actually import your nesC module and interface files. Under the "nesC module" section click the "+" button to open the file select window. Now click
"Browse" and select the nesC module file you want to import. This window will show you what interfaces are needed for the module file. Click ok and now your module nesC is setup to be imported.

If you don't know remember exactly what interface files you need to import then you can click the "Next" button. This will bring you to a page where it lists all the missing interfaces. This can be seen in Figure 3.28, “Scenario Imports”. (In this case the wizard shows what interfaces are missing for the BlinkM module). Press "Back" to go the previous screen.

**Figure 3.29. Missing Imports**

![Import Cadena Types Wizard](image)

Now under the interface section push the "+" button and select the remaining interfaces. You can keep checking whether or not you have all the correct interfaces imported by repeating the last step. When you are done click "Next" and this will take you to a confirmation page. On this page you can make sure all your imports are correct. Click "Finish" when you are done and your new module will be created.
Chapter 4. Import

Overview

This chapter provides details about features that involve the importing of artifacts. This includes tasks to import projects and nesC files into Cadena. Each section in this chapter is a different feature that will help you import artifacts.

The sections in this chapter are:

- Import Interface Type into an Existing Module: Import a nesC interface file into a Cadena Module
- Import Component Type into an Existing Module: Import a nesC module file into a Cadena Module

*Note: There is no ordering to the sections.*

Import Interface Type into an Existing Module

Importing an Interface Type into an existing Module from a nesC interface file can be done in many ways. Each of them has a different starting point but all of them have the same result.

The following instructions assume that there is an existing TinyOS project that has an existing nesC module in the Cadena Module path. The instructions further assume that you have that Module open in the Cadena Module Editor.

There are three starting points for importing a nesC interface file into the currently selected Module.

- Menu: "TinyOS Actions" | "Import nesC Interface"
- Tool-bar: "Import nesC Interface" button
- Context-Menu: Right-click in the Editor | "TinyOS Actions" | "Import nesC Interface"

Each of these results in a dialog being shown that looks like Figure 4.1, “nesC Interface File Import Dialog: Empty”.

**Figure 4.1. nesC Interface File Import Dialog: Empty**

To find the nesC file to import, press the "Browse" button and locate the file. Once you have selected the file, the dialog will be populated with the information from the interface (as seen in Figure 4.2, “nesC Interface File Import Dialog: StdControl”). Check that the interface name and list of operations are correct.
and once you have confirmed this, press the "Finish" button. This will import the selected nesC interface into the current Module as an Interface Type.

**Figure 4.2. nesC Interface File Import Dialog: StdControl**

![Import nesC Interface File Dialog](image)

### Import Component Type into an Existing Module

Importing a nesC module file into an existing Module can be done in many ways. Each of them has a different starting point but all have the same result.

The following instructions assume that there is an existing TinyOS project that has an existing nesC module in the Cadena Module path. The instructions further assume that you have that Module open in the Cadena Module Editor.

There are three starting points for importing a nesC module file into this Module.

- **Menu**: "TinyOS Actions" | "Import nesC Module"
- **Tool-bar**: "Import nesC Module" button
- **Context-Menu**: Right-click in the Editor | "TinyOS Actions" | "Import nesC Module"

Each option results in a dialog that looks like Figure 4.3, “nesC Module File Import Dialog: Empty”.

**Figure 4.3. nesC Module File Import Dialog: Empty**

![Import nesC Module File Dialog](image)
Use that dialog to browse the filesystem to find the nesC module file you want to import. This can be done by pressing the "Browse" button and using the file system browser. Once the file is selected, the name of the Module and the ports declared on that Module will be shown. Check the information for accuracy and when satisfied, press "Finish". As a result, the specified nesC module will be imported into the currently selected Module as a Component Type.

**Figure 4.4. nesC Module File Import Dialog: TimerM**

You should note that the dialog does not help you resolve any missing Interface Types that are referred to in that Component Type. It will warn you of this fact and won't permit you to proceed. This means that you must have them available in that Module prior to using that dialog (either directly or through the nesC module import functionality).
Chapter 5. Generate

Overview

This chapter provides details about features that involve the generation of artifacts. This includes code generation as well as other features related to generation. Each section in this chapter is a different feature that will help you to generate artifacts.

The sections in this chapter are:

• Configure Auto Generation: Shows how to configure the nesC source generation engine for a project

• Generate nesC Configuration: Generate a nesC configuration from a Scenario

• Generate nesC Module: Generate a nesC module from a Component Type

*Note: There is no ordering to the sections.*

Configure Code Generation

Every TinyOS project now has the ability to generate nesC source code automatically when the model changes. [Note: To be more precise, it will be automatically generated when the model is saved.] To make this happen, we provide the ability to configure if it is turned on, if it will delete older portions of the model, and where it will store the generated code. The three sections that follow detail each of these configurations.

Turn Auto Generation On/Off

Auto-generation of nesC source code can be turned on or off at any time by the user using the Project Properties. To do this, you should select the project you want to configure, right-click to bring up the context-menu, and select Properties. This will bring up a dialog that looks like Figure 5.1, “nesC Code Generation Project Properties for Surge”. Switch to the nesC Code Generation page and you will see the on/off switch which is a checkbox labeled "Automatically generate nesC code when the Cadena model is saved". If this is checked, code generation is turned on, otherwise code generation is turned off.
Delete Old Model Artifacts

When auto-generation occurs, it might be the case that a portion of the model has been removed. When this happens, the auto-generation engine can do one of two things. It can either ignore it or it can remove the code that it generated for that part of the model. In most cases, you won’t want to turn this on but rather delete these portions manually. The danger here is that you will accidentally delete a portion of the model and lose any code that you had written in the nesC source file. So be cautious with this setting and make sure you know what you are doing.

To configure the auto-generation engine to automatically delete old artifacts simply select the checkbox labeled "Automatically delete nesC code when the corresponding Cadena artifacts are deleted". An example of this can be seen in Figure 5.1, “nesC Code Generation Project Properties for Surge”.

Source Code Destination

The auto-generation feature allows you to set a location for the source to be stored when it is generated. You can use the default location (a folder named src in the root of the project) or any folder that exists in that project. An example of this dialog can be seen in Figure 5.1, “nesC Code Generation Project Properties for Surge”. You can see a radio box that allows you to select the default (radio button labeled with Default) or a user-defined folder in the project (radio button labeled Project and then select the folder using the Browse button).

Generate nesC Configuration

An important feature of the Cadena TinyOS platform plugin is the ability to generate nesC source from the model. Specifically, a user will need to generate a nesC configuration file from a Scenario that they have created using Cadena and the TinyOS platform plugin. This section will walk you through this process, which can be accomplished in 4 different ways.

Before working through this section you will need to have a Cadena/TinyOS project that has a Scenario that uses the nesC style.

Generate nesC Configuration using Editors Context Menu

One way to generate a nesC configuration is to use the context-menu in the Scenario Editor. This might be the most convenient method when you are working in the editor and don’t want to move the mouse very far.

To generate the nesC configuration in this way you must have the Scenario opened and be in either the table or graph view. Once in either view right-click to bring up the context menu. On that menu you will see a sub-menu named TinyOS Actions. Expanding that sub-menu will display a single action named "Generate Configuration File". Selecting this will cause the code generation engine to generate the configuration and place it into the configured location (see the section called “Source Code Destination” for more on configuring this location).

Generate nesC Configuration using Navigator Context Menu

One way to generate a nesC configuration is to use the context-menu in the Navigator or Package Explorer views. This might be the most convenient method when you are not working in the Scenario Editor (i.e., you don't have the Scenario opened).
To generate the nesC configuration in this way you must have the Scenario file selected in the Navigator or Package Explorer. Once selected, right-click to bring up the context-menu. In that menu you will see a sub-menu named TinyOS Actions which contains a single action named "Generate Configuration File". Selecting this will cause the code generation engine to generate the configuration and place it into the configured location (see the section called “Source Code Destination” for more on configuring this location).

**Generate nesC Configuration using Toolbar Button**

One way to generate a nesC configuration is to use the toolbar button that is available when the Scenario editor is open. This is the most user-friendly method since it is easy to find.

To generate the nesC configuration in this way you must have the Scenario opened. When it is, you will see the button next to the other toolbar buttons and has a tooltip of "Export nesC Configuration File for this Scenario". Selecting this will cause the code generation engine to generate the configuration and place it into the configured location (see the section called “Source Code Destination” for more on configuring this location).

**Generate nesC Configuration using Menubar**

One way to generate a nesC configuration is to use the menubar action that is available when the Scenario editor is open. This is the used by people who prefer menu systems.

To generate the nesC configuration in this way you must have the Scenario opened. When it is, you will see the menu at the top of the screen named "TinyOS Actions". In that menu you will see an action named "Export nesC Configuration File". Selecting this will cause the code generation engine to generate the configuration and place it into the configured location (see the section called “Source Code Destination” for more on configuring this location).

**Generate nesC Module**

An important feature of the Cadena TinyOS platform plugin is the ability to generate nesC source from the model. Specifically, a user will need to generate a nesC module file from a Component Type that they have created using Cadena and the TinyOS platform plugin. This section will walk you through this process, which can be accomplished in 5 different ways.

Before working through this section you will need to have a Cadena/TinyOS project that has a Module that uses the nesC style.

**Generate nesC Module using Editors Context Menu**

One way to generate a nesC module is to use the context-menu in the Module Editor. This might be the most convienent method when you are working in the editor and don't want to move the mouse very far. It will also be the only way to generate a single nesC module file (as opposed to generating nesC module files for all Component Types in a Module file).

To generate the nesC module in this way you must have the Module opened and be in the table view. Once there you should select the Component Type that you want to generate code for. Right-click it and the context-menu will be shown that has an action named "Generate nesC Module File". Selecting this will cause the code generation engine to generate the module file and place it into the configured location (see the section called “Source Code Destination” for more on configuring this location).
Generate nesC Modules using Editors Context Menu

One way to generate a nesC modules is to use the context-menu in the Module Editor. This might be the most convienent method when you are working in the editor and don't want to move the mouse very far.

To generate the nesC modules in this way you must have the Module opened and be in the table view. Once there you can right-click anywhere to bring up the context-menu. On that menu you will see a sub-menu named TinyOSActions which contains an action named "Generate All Module Component Code". Selecting this will cause the code generation engine to generate the module files for each Component Type in this Module file and place it into the configured location (see the section called “Source Code Destination” for more on configuring this location).

Generate nesC Modules using Navigator Context Menu

One way to generate a nesC modules is to use the context-menu in the Navigator or Package Explorer views. This might be the most convienent method when you are not working in the Module Editor (i.e., you don't have the Module opened).

To generate the nesC modules in this way you must have the Scenario file selected in the Navigator or Package Explorer. Once selected, right-click to bring up the context-menu. In that menu you will see a sub-menu named TinyOS Actions which contains an action named "Generate All Module Component Code". Selecting this will cause the code generation engine to generate the module files for each Component Type in this Module file and place it into the configured location (see the section called “Source Code Destination” for more on configuring this location).

Generate nesC Modules using Editors Toolbar Button

One way to generate a nesC modules is to use the Toolbar button available in the Module Editor. This is the most user-friendly method since it is easy to find.

To generate the nesC modules in this way you must have the Module opened. When it is opened a toolbar will be available that has a button on it with a tooltip saying "Export the nesC Module code for the Components in this Module". Selecting this will cause the code generation engine to generate the module files for each Component Type in this Module file and place it into the configured location (see the section called “Source Code Destination” for more on configuring this location).

Generate nesC Modules using Menubar

One way to generate a nesC modules is to use the menubar available in the Module Editor. This is the used by people who prefer menu systems.

To generate the nesC modules in this way you must have the Module opened. When it is opened a menu will be available that has an action in it named "Export nesC Module Code". Selecting this will cause the code generation engine to generate the module files for each Component Type in this Module file and place it into the configured location (see the section called “Source Code Destination” for more on configuring this location).
Chapter 6. Share

Overview

This chapter provides details about features that involve sharing artifacts. This includes sharing through version control and through project/artifact exporting. Each section in this chapter is a different feature that will help you to share artifacts.

The sections in this chapter are:

- Share using archive: Exporting
- Share using archive: Importing
- Share using CVS

*Note: There is no ordering to the sections.*

Share using archive: Exporting

One way to share a project is to export it into an archive file. This allows you to share your project with a colleague or move it to a different computer with no hassle. To ensure compatibility between computers make sure you have the same JDK version and Cadena versions.

The first step is to get into the Package Explorer where you can see all your projects. Then right click on the project you want to share and click "Export...". This will look like Figure 6.1, “Export”.

**Figure 6.1. Export**

This brings up the export wizard in Eclipse and shows the multiple ways a project can be exported. Select "Archive file" and click "Next".
The last part of the export wizard lets you select where to export your archive. This can be done by clicking the "Browse" button. Once you have done this click "Finish" and an archive of your project will be made in the location specified. This can be seen in Figure 6.2, “Export Wizard Page 2”.

**Figure 6.2. Export Wizard Page 2**

![Export Wizard Page 2](image)

**Share using archive: Importing**

Importing an existing project into Eclipse from an archive is very easy. You will make use of the Eclipse import wizard for existing projects.

To start the process, right click in the Package Explorer and select "Import". This can be seen in Figure 6.3, “Import”.

**Figure 6.3. Import**

![Import](image)
This is the Eclipse import wizard. Select “Existing projects into workspace”. This can be seen in Figure 6.4, “Import Wizard Page1”. Then click Next.

**Figure 6.4. Import Wizard Page1**

Now check the radio button next to ”Select Archive File”. Click the ”Browse” button and select the archive of a project you want to import. Click the finish button and the project will be imported into Eclipse. You can see this step in Figure 6.5, “Import Wizard Page2”.

**Figure 6.5. Import Wizard Page2**

---

**Share using CVS**

CVS stands for Concurrent Versions System and is a way to share projects among team members. It allows multiple users to "check out" a project, make changes, and commit those changes. This is a very basic explanation that only scratches at the surface of what is possible with CVS.

This section will show you how to setup CVS within Eclipse and share a project. This part assumes you have access to a CVS server and have an account already setup.

This first step is to open the CVS perspective in Eclipse. You can do this by going to the "Window" menu and selecting "Open Perspective->Other".
Figure 6.6. Open Perspective

The "Select Perspective window" opens up showing you the different Perspectives that Eclipse has available. Click on "CVS Repository Exploring" and press OK. This can be seen in Figure 6.7, “Select a Perspective”. The Eclipse UI will change as you are taken to the new perspective. The CVS Repositories view is similar to the Package Explorer but it is used for connecting browsing and creating connections to CVS repositories.

Figure 6.7. Select a Perspective

Now you can add a CVS server and start sharing your projects. Right click in the CVS Repositories space and select "New->Repository location". This can be seen in Figure 6.8, “Add new location”.

Now you can add a CVS server and start sharing your projects. Right click in the CVS Repositories space and select "New->Repository location". This can be seen in Figure 6.8, “Add new location”.
Figure 6.8. Add new location

In the "Add CVS Repository" Window you can fill out the information for your CVS server. This can be seen in Figure 6.9, “New CVS location”. Normally the repository path is "/cvsroot/mylocation" but this could be different for your CVS server. There are different types of connections. Many CVS repositories use the "pserver" connection type. Some use the "ext-ssh" which is an encrypted connection. Once again this will vary on the server you have.

Figure 6.9. New CVS location

Once you are done click Finish and the new CVS location will be added. Now you can share a project on your CVS site.

Switch back to the perspective you were using to manage the project. Once there, right click on the project you want to share and select "Team->Share Project". The next screen you will see will ask you which CVS
location you want to use. You can see this in Figure 6.10, “Sharing your project”. Select your repository location and push the next button.

**Figure 6.10. Sharing your project**

Choose to "Use Project name as module name". This means that on the CVS server your project name will be what people see. Click next through until finished.

Now that your project is shared someone else on your team can connect to your CVS server and "check out" your project. This is done by right clicking on a project in your CVS and selecting "Check out". When they are done modifying the project they can commit it by right clicking on the project and go to "Team->Commit...”.

To make sure you have the most recent version of your project click on your project, go to "Team->Update". This will make sure you have the current version of the project that's on your CVS server.
Chapter 7. Explore

Overview

This chapter provides details about features that involve exploring artifacts. This includes viewing and searching features to allow the user to learn about their projects and models. Each section in this chapter is a different feature that will help you explore the artifacts.

The sections in this chapter are:

• ?

Note: There is no ordering to the sections.
Chapter 8. Maintain

Overview

This chapter provides details about features that involve maintaining artifacts. This includes configuration and settings for projects and models. Each section in this chapter is a different feature that will help you in maintaining artifacts.

The sections in this chapter are:

- 

*Note: There is no ordering to the sections.*
# Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cadena</td>
<td>An Eclipse-based extensible integrated modeling and development framework for component-based systems.</td>
</tr>
<tr>
<td>TinyOS</td>
<td>An open-source operating system designed for wireless embedded sensor networks. It features a component-based architecture which enables rapid innovation and implementation while minimizing code size as required by the severe memory constraints inherent in sensor networks.</td>
</tr>
<tr>
<td>nesC</td>
<td>An extension to the C programming language designed to embody the structuring concepts and execution model of TinyOS.</td>
</tr>
<tr>
<td>Eclipse</td>
<td>An open source community whose projects are focused on building an open development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle. When we refer to Eclipse it is usually as an IDE or platform and not the project or community.</td>
</tr>
<tr>
<td>workspace</td>
<td>An Eclipse term that refers to the central hub for all user data. This is a specific folder/directory. A good quote from the Eclipse website is &quot;you can think of the platform workbench as a tool that allows the user to navigate and manipulate the workspace&quot;.</td>
</tr>
<tr>
<td>project</td>
<td>An Eclipse term that refers to a specific type of resource in the workspace. To be more specific, a workspace contains a collection of projects. Projects contain files and folders.</td>
</tr>
<tr>
<td>Module File</td>
<td>A Cadena term that refers to a file that contains a Cadena Module.</td>
</tr>
<tr>
<td>Scenario File</td>
<td>A Cadena term that refers to a file that contains a Cadena Scenario.</td>
</tr>
<tr>
<td>Scenario</td>
<td>A Cadena term that refers to a collection of instances (component, scenario, and connector) that define a modeled application.</td>
</tr>
<tr>
<td>Module</td>
<td>A Cadena term that refers to the description of the types available in the model which will be used at the Scenario tier. Modules contain definitions of Types that are used to define Scenario instances.</td>
</tr>
<tr>
<td>Style</td>
<td>A Cadena term that refers to the description of the platform that will be modeled at the other tiers of Cadena (module and scenario tiers). In other words, the style helps define a language to use in the Module tier. Styles contain definitions of Kinds (and Meta-Kinds) that are used to define Module Types.</td>
</tr>
<tr>
<td>nesC Interface</td>
<td>A TinyOS/nesC term that refers to a collection of methods (or method signatures) with a name. In nesC, components (modules and configurations) provide and use interfaces.</td>
</tr>
<tr>
<td>nesC Module</td>
<td>A TinyOS/nesC term that refers to a component that holds logic. This uses and provides interfaces, commands, and events. It also holds the logic that maps to the defined interfaces, commands, and events.</td>
</tr>
<tr>
<td>nesC Configuration</td>
<td>A TinyOS/nesC term that refers to a component that does not hold logic. A configuration defines a collection of components (modules and configurations)</td>
</tr>
</tbody>
</table>
and connectors as well as an optional collection of interfaces, commands, and events that it uses and provides. This holds no logic.

<table>
<thead>
<tr>
<th>Nature</th>
<th>An Eclipse term that refers to flags set on Eclipse projects. These flags help Eclipse behave in a prescribed way. For example, certain actions, features, and builders are only available in projects with certain natures. For example, the Cadena Specification Path can only be defined in a project with a Cadena nature.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification Path</td>
<td>A Cadena term that refers to the path Cadena uses to find the model specifications available in a project. This includes three distinct paths for styles, modules, and scenarios.</td>
</tr>
<tr>
<td>Interface Type</td>
<td>…</td>
</tr>
<tr>
<td>Component Type</td>
<td>…</td>
</tr>
<tr>
<td>Component Instance</td>
<td>…</td>
</tr>
<tr>
<td>Scenario Instance</td>
<td>…</td>
</tr>
<tr>
<td>TinyOS Module</td>
<td>A Cadena/nesC term that refers to a Cadena Module that is set to use the nesC style.</td>
</tr>
<tr>
<td>TinyOS Scenario</td>
<td>A Cadena/nesC term that refers to a Cadena Scenario that is set to use the nesC style.</td>
</tr>
<tr>
<td>Architectural Definition Language (ADL)</td>
<td>…</td>
</tr>
<tr>
<td>Product-Line Development</td>
<td>…</td>
</tr>
<tr>
<td>Software Product Lines (SPL)</td>
<td>…</td>
</tr>
<tr>
<td>Middleware</td>
<td>…</td>
</tr>
<tr>
<td>Type</td>
<td>…</td>
</tr>
<tr>
<td>Service</td>
<td>…</td>
</tr>
<tr>
<td>Meta Model</td>
<td>…</td>
</tr>
<tr>
<td>Component</td>
<td>…</td>
</tr>
<tr>
<td>Interface</td>
<td>…</td>
</tr>
<tr>
<td>Connector</td>
<td>…</td>
</tr>
<tr>
<td>Meta Kind</td>
<td>…</td>
</tr>
<tr>
<td>Kind</td>
<td>…</td>
</tr>
<tr>
<td>Platform</td>
<td>…</td>
</tr>
<tr>
<td>Port Option</td>
<td>…</td>
</tr>
<tr>
<td>Role</td>
<td>…</td>
</tr>
<tr>
<td>Interface Kind</td>
<td>…</td>
</tr>
<tr>
<td>Glossary</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Component Kind</td>
<td>...</td>
</tr>
<tr>
<td>Connector Kind</td>
<td>...</td>
</tr>
<tr>
<td>Instance</td>
<td>...</td>
</tr>
<tr>
<td>Level</td>
<td>...</td>
</tr>
<tr>
<td>Layer</td>
<td>...</td>
</tr>
<tr>
<td>Assembly</td>
<td>...</td>
</tr>
</tbody>
</table>
Bibliography


[nesC:URL] “nesC Web Site”.

[TinyOS:URL] “TinyOS Web Site”.

[Cadena:URL] “[\sc Cadena] Web Site”.