Elektrobit

EB corbos Starter Kit user's guide

2.8, Release

Elektrobit Automotive GmbH
Am Wolfsmantel 46
91058 Erlangen, Germany
+49 9131 7701 0
+49 9131 7701 6333
info.automotive@elektrobit.com

Technical support

Europe	Japan	USA
+49 9131 7701 6060	+81 3 5577 6110	+1 888 346 3813

China	Support URL
+86 21 5043 1951 ext 0	https://www.elektrobit.com/support

Legal disclaimer

Table of Contents

1. About this documentation

1.1. Typography conventions

Throughout the documentation you see that words and phrases are displayed in bold or italic font, or in mono space font.

To find out what these conventions mean, please consult the following table. All default text is written in font Arial Regular without any markup.

Convention	Item is used	Example
Arial italics	to define new terms	The *basic building blocks* of a configuration are module configurations.
Arial italics	to emphasize	If your project's release version is mixed, all content types are available. It is thus called *mixed version*.
Arial italics	to indicate that a term is explained in the glossary	...exchanges *protocol data unit*s (*PDU*s) with its peer instance of other ECUs.
Arial boldface	for pop-up windows names	The **Bulk Change** editor enables you to change data in bulk.
Arial boldface	for menus and submenus	Choose the **Options** menu.
Arial boldface	for buttons	Select **OK**.
Arial boldface	for keyboard keys	Press the **Enter** key
Arial boldface	for keyboard key combinations	Press **Ctrl**+**Alt**+**Delete**
Arial boldface	for commands	Convert the XDM file to the newer version by using the **legacy convert** command.
Monospace font (Courier)	for file and folder names, also for chapter names	Put your script in the `function_name\abc-folder`
Monospace font (Courier)	for code	`CC_FILES_TO_BUILD =(PROJECT_-PATH)\source\network\can_node.-c CC_FILES_TO_BUILD += $(PROJECT_-PATH)\source\network\can_config.c`
Monospace font (Courier)	for function names, methods, or routines	The cos function finds the cosine of each array element. Syntax line example is `MLGetVar ML_var_name`

Convention	Item is used	Example
Monospace font (Courier)	for user input/indicates variable text	Enter a `three-digit prefix` in the menu line.
Square brackets [ ]	to denote optional parameters; for command syntax with optional parameters	`insertBefore [<opt>]`
Curly brackets {}	to denote mandatory parameters; for command syntax with mandatory parameters (in curly brackets)	`insertBefore {<file>}`
Three dots …	to indicate further parameters; for command syntax, indicates further parameters	`insertBefore [<opt>…]`
A vertical bar |	to separate parameters in a list from which one parameters must be chosen or used; for command syntax, indicates a choice of parameters	`allowinvalidmarkup {on|off}`
Warning	to show information vital for the success of your configuration	**WARNING**  **This is a warning** This is what a warning looks like.
Note	to give additional important information on the subject	**NOTE**  **This is a note** This is what a note looks like.
Tip	to provide helpful hints and tips	**TIP**  **This is a tip** This is what a tip looks like.
Example	to demonstrate or illustrate information	 **Example 1.1. This is an example** This is what an example looks like.

This is a step-by-step instruction

Whenever you see the bar with step traces, you are looking at step-by-step instructions or how-tos.

Prerequisite:

- This line lists the prerequisites to the instructions.

Step 1
An instruction to complete the task.

Step 2
An instruction to complete the task.

Step 3
An instruction to complete the task.

2. Safe and correct use

2.1. Intended usage of EB corbos Starter Kit

EB corbos Starter Kit is intended to be used by developers to evaluate the development and testing of applications for the AUTOSAR Runtime Environment.

2.2. Content of the EB corbos Starter Kit delivery

The following components are provided with EB corbos Starter Kit:

Component	Description	Abbreviation
EB corbos Studio	The tooling and build environment.	Studio
EB corbos AdaptiveCore Generic	Generic AUTOSAR Adaptive Platform that is independent of controller and operating system (Linux or QNX)	ADG
EB corbos AdaptiveCore Platform	Platform specific AUTOSAR Adaptive Platform	ADP
demo applications	restService, demo, Sensor_handler, Sensor_Preprocessor, Sensor_Manager, Sensor_dataProcessor, Display_Manager, dm_lamp, dm_motor_operation, dm_motor_speed, dm_motor_speed_monitor	

2.3. Possible misuse of EB corbos Starter Kit

WARNING **Possible misuse and liability**

You may use the software only as in accordance with the intended usage and as permitted in the applicable license terms and agreements. Elektrobit Automotive GmbH assumes no liability and cannot be held responsible for any use of the software that is not in compliance with the applicable license terms and agreements.

2.4. Target audience

The target audience of this documentation are automotive software engineers.

2.5. Required knowledge

- Intermediate knowledge in developing with Eclipse
- Background knowledge of Adaptive AUTOSAR
- Experience in programming AUTOSAR- and OSEK/VDX-compliant ECUs

3. Background information

3.1. EB corbos Starter Kit architecture

In order to provide a complete environment to model and develop an Adaptive Application, the following architecture is proposed:

Component	Description
EB corbos Studio	This is the main tool that you are using for modeling a configuration, application development, building and deploying on the target. It provides CDT support. EB corbos Studio runs on a Windows host.
Docker container (ARA SDK platform)	It contains Yocto SDK tool chains for building targets. It is used to build an Adaptive Application
Docker container (Emulated ARA run time)	Two QEMU corbos Linux VMs are running as an ARA image. It uses the AUTOSAR code for ARA run-time. It is launched inside the docker container.
Virtual networking between Windows and QEMU instances	► Shared file system between build system and Windows which allows to write code and build in one file tree ► SSH to trigger builds ► Binaries are copied to ARA run-time system via SCP. ► Remote debugging with GDB is possible. ► Two ECUs connected via SOME/IP are simulated.

The figure below shows QEMU images that run inside the Docker container.

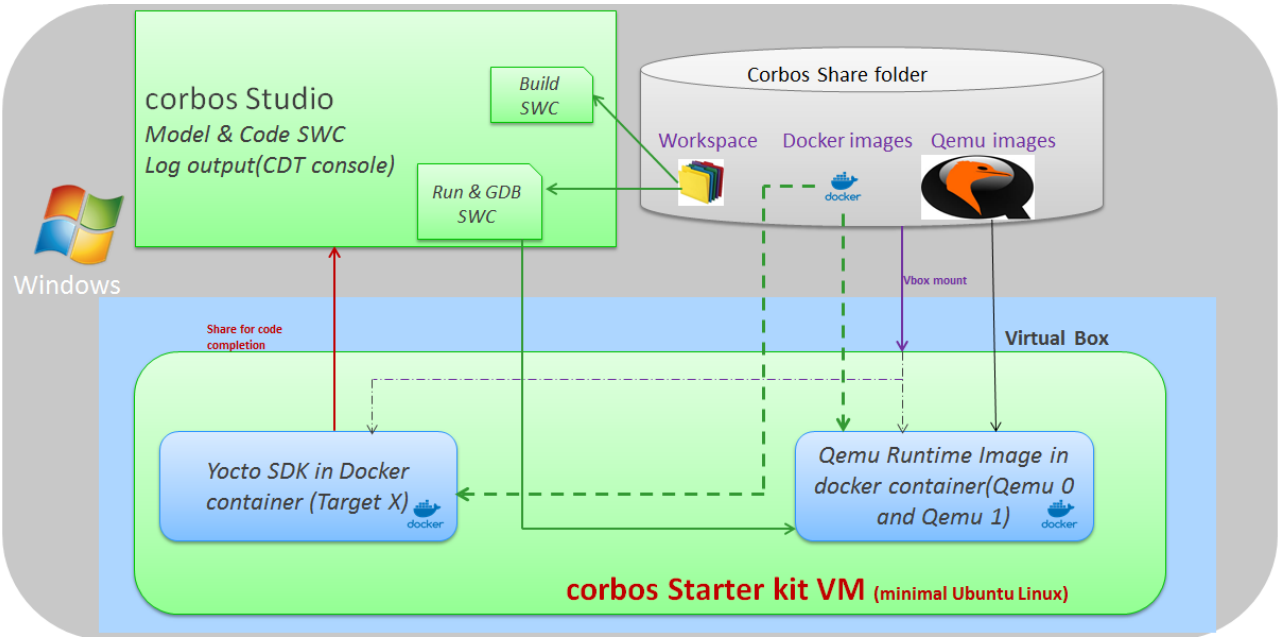


Figure 3.1. StarterKit architecture

See also Section 6.4, “Tuning the RAM usage”, Section 6.5, “Enable multiple cores”

4. Installation

4.1. Prerequisites and system requirements

NOTE

User rights

To install the EB corbos Starter Kit on Windows 7, Windows 8 or Windows 10 systems, you require administrator rights.

Operating system	Windows 7 64 bit, Windows 10 64 bit
Virtualization	Virtualization is enabled in BIOS
RAM	8 GB
Processor	At least a four-core processor
Oracle VM VirtualBox	Version 5.2.22 or higher 5.2.xx and of 32bit

Table 4.1. Recommended settings for EB corbos Starter Kit

4.2. Package content

Component	Version
ADG	1.4.2
EB corbos Studio	2019_R02_01_01
EB Corbos Linux	1.7.0

The EB corbos Starter Kit is delivered as a NSIS installer (starterkit_vx.x.exe) where x.x denotes the version number. Once after installation, you will find an installed folder at `C:\EB\corbos\`, which contains the following major components:

`corbos\common_tools`
: This directory contains tools which are used common for all the starterkit versions, for example Python,md5sum.

`corbos\common_tools\Python27`
: Python 2.7 installation kit.

`corbos\common_tools\Wireshark`
: Wireshark tool for packet tracing.

`corbos\global_config`
This directory is reserved for the global configuration files for EB corbos Starter Kit, for example connection.xml

`corbos\starterkit_vx.x`
This directory contains the data for particular version of EB corbos Starter Kit.

`corbos\target_tools`
This directory contains the target specific tools, for example gdb tools for qemu target.

`corbos\target_tools\qemu\gdb`
GDB tool to debug AUTOSAR/C++ projects for qemu.

`corbos\starterkit_vx.x\studio.cmd`
Starts EB corbos Studio and is use to configure a predefined workspace in order to get access to ARA SDK.

`corbos\starterkit_vx.x\EB_corbos_Studio\EB_corbos_Studio.exe`
EB corbos Studio is a EB solution for modeling ARA configuration and Adaptive Application development.

`corbos\starterkit_vx.x\.vm\corbos-starterkit-vm.ova`
It contains QEMU docker container and SDK toolchain docker container respective to ADG version.

`corbos\starterkit_vx.x\templates`
Contains the configured items that are needed when you create a new AUTOSAR/C++ project.

`corbos\starterkit_vx.x\scripts\python`
Contains the scripts that are needed to install EB corbos Starter Kit.

`corbos\starterkit_vx.x\configure-project.cmd`
Script that is used to configure a new AUTOSAR/C++ project.

`corbos\starterkit_vx.x\switch-workspace.cmd`
Script that is used to switch to a new EB tresos Studio workspace.

`corbos\starterkit_vx.x\documentation\users_guide_qemu.pdf`
The user's guide describes how to create, run and test Adaptive Application.

For details about EB corbos Starter Kit architecture, see Section 3.1, “EB corbos Starter Kit architecture”.

NOTE **License information**

The licensing user's guide for the EB corbos Studio is available in the documentation folder of the starterkit_vx.x package.

4.3. Installing EB corbos Starter Kit

During the installation, a folder is created at C:\Users\%USERPROFILE%\.ssh that contains the known_hosts file with information related to QEMU.

The version postfix has not been added to the *corbos-starterkit-vm* for example *corbos-starterkit-vm-vx.x* where x.x denotes the EB corbos Starter Kit release version number.

NOTE

Every EB corbos Starter Kit installer to be installed at same path

First trigger starterkit_vx.x.exe it will setup the base require to install install_qemu_sdk_-vx.x.exe .

Installing EB corbos Starter Kit

Prerequisite:

- You download the starterkit_vx.x.exe, install_qemu_sdk_vx.x.exe , third_party_software_vx.x.exe and unistaller_starterkit_vx.x.exe from EB command. Also please make sure to have third_party_software_vx.-x.exe in same folder where starterkit_vx.x.exe has been downloaded . The third_party_software_vx.x.exe do not require to install manually, it will be installed automatically during starterkit_vx.x.exe instalation.
- You installed VirtualBox version 5.2.22 (32-bit) or higher 5.2.xx.
- Virtualization is enabled in your BIOS.
- EB corbos Studio now uses the Flexera license mechanism. In order to use EB corbos Studio you either need access to the EB license server or a Wibu dongle with user code 13 and a license file. A configuration for the license server (flexlm.cfg) a license file for the dongle (eb_corbos_studio_dongled_UC-13_until2018-11-01.lic).To use the licenses, copy the respective files into the 'bin' folder of your installation of EB corbos Studio. Alternatively you can set the path to the license in the preferences within the UI.
- **Please Ensure your previous version of starterkit and other registered VMs are closed while instaling new version . For an example , lets say while instaling starterkit_2.3 , previous version of starterkit-vm-2.2 must be in Powered off State as shown below. Also make sure the other VMs must be in Powered off State.**

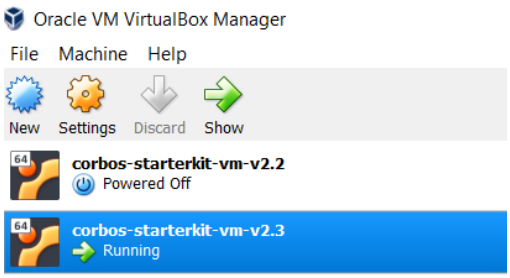


Figure 4.1. Prerequisite Prior to Trigering Starterkit Installation:

Step 1
Double click the starterkit_vx.x.exe.

An installation wizard opens up.

You can choose path and select the components then click next and install the setup.

NOTE

During Installation if following "User Account Control" POP up appears , then click on "Yes".

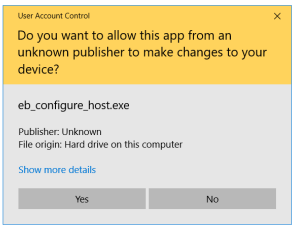


Figure 4.2. eb_configure_host.exe UAC POP UP:

Step 2
Double click on install_qemu_sdk_vx.x.exe and select the same path in which starterkit_vx.x.exe is installed. This will install QEMU and SDK container.

Step 3
Follow the steps in the installation wizard.

The installation continues for up to 10 minutes.

Step 4
Click **Finish** to finalize the installation.

Step 5
The default installation directory for the EB corbos Starter Kit package is `C:\EB\corbos\`.

NOTE

Select **Yes** on the pop-up **Virtual Interface Is Trying To Change**.

4.3.1. Handling installation problems

Problems during the installation::

- Check if virtualization is enabled in your BIOS. It is disabled if the error log displays the following: `VBoxManage.exe: error: VT-x is disabled in the BIOS for all CPU modes.`

Problems related to starting the virtual machine:

- Remove the current artifacts and start the installation:

  1. Shut down your VM in the VirtualBox Manager using command **tools\Python27\python.exe scripts\python\main.py --command stopVBox** run from EB corbos Starter Kit installed directory using command prompt.
  2. Run **tools\Python27\python.exe scripts\python\main.py --command clearVBox** from EB corbos Starter Kit installed directory using command prompt.
  3. Run the setup installer starterkit_vx.x.exe again.

  If the installation still does not work, manually delete the VM artifacts, see Chapter 9, “Troubleshooting“.

4.3.2. Uninstalling EB corbos Starter Kit

Download the unistaller_starterkit_vx.x.exe from EB command.

Steps to uninstall EB corbos Starter Kit

Step 1
Please close the Oracle VM VirtualBox Manager

Step 2
Double click on `unistaller_starterkit_vx.x.exe`. The uninstallation procedure will begin.

Step 3
Select one of the option to uninstall.

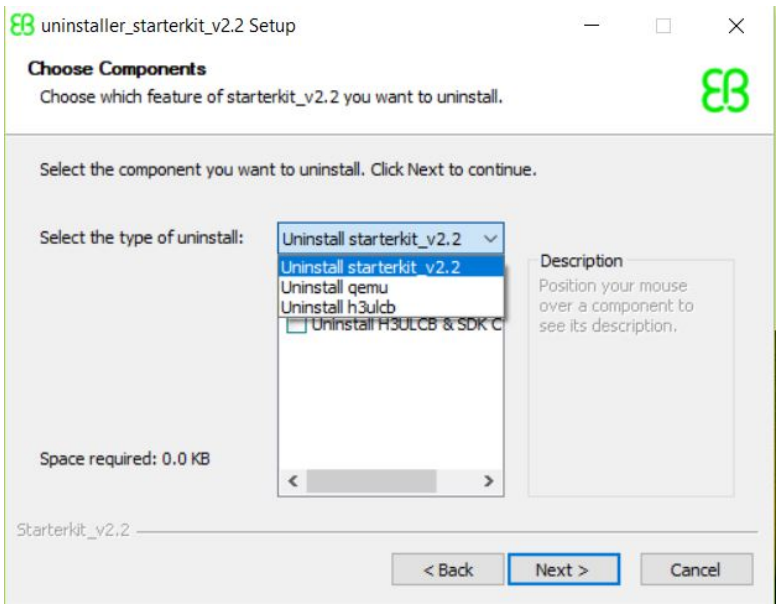


Figure 4.3. Options to uninstall

NOTE If any "User Account Control" POP up appears as below, need to click on "yes"

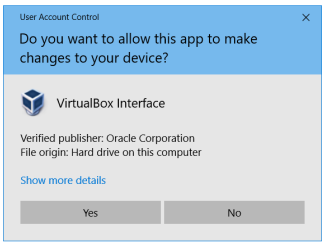


Figure 4.4. User Account Control POP up Virtual Box

Step 4

After the above step, navigate to installation path (`C:\EB\corbos\`, default path, unless changed during installation) using Windows explorer, Select `C:\EB\corbos\` directory and press Shift+Delete on `starterkit_vx.x`. from your keyboard to completely remove the residual files if it exists.

Step 5

Manually delete all the corbos-starterkit-vm* artifacts from your VirtualBox VM storage space, if there exists.

5. Getting started

5.1. Starting EB corbos Studio

Starting EB corbos Studio

Prerequisite:

- EB corbos Starter Kit is installed.
- wizard i.e. `C:/EB/corbos/starterkit_vx.x/workspace_corbos`.

Step 1
Open the directory where you install the EB corbos Starter Kit.

Step 2
Run **studio.cmd**.

The installation wizard opens.

Step 3
Copy the license files *flexlm.cfg* into the *bin* folder of your installation of EB corbos Studio

Alternatively you can set the path to the license in the preferences within the UI.

Step 4
Follow the instructions in the installation wizard.

Step 5
Switch to C/C++ perspective.

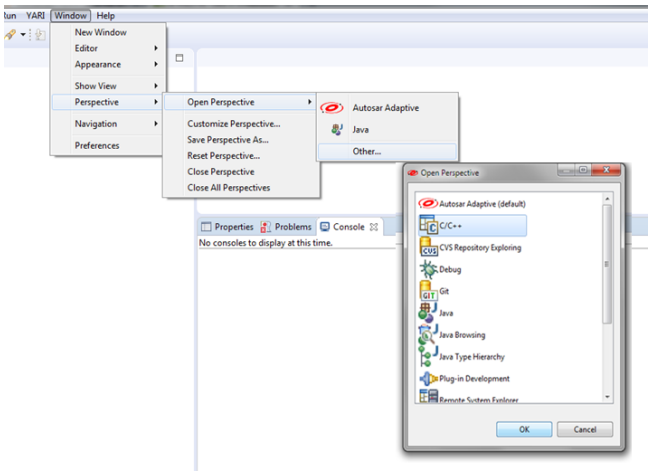


Figure 5.1. Switch to C/C++ perspective

Result:

Demos are available in the workspace:

- `Sensor_handler`
- `Sensor_Preprocessor`
- `Sensor_Manager`
- `Sensor_dataProcessor`
- `Display_Manager`
- `restService`
- `demo`
- `dm_lamp`
- `dm_motor_operation`
- `dm_motor_speed`
- `dm_motor_speed_monitor`

NOTE **Start EB corbos Studio manually**

Do not let EB corbos Studio restart itself.

To have all mandatory parameters, start EB corbos Studio via **studio.cmd**.

5.2. Switching the workspace

The workspace is a directory where projects and settings are stored. The default workspace *workspace_corbos* is available at installed path of EB corbos Starter Kit i.e. default at `C:/EB/corbos/starterkit_vx.x/`.

NOTE **No import of default demo projects into studio**

When a new workspace is created, the default demo projects are not imported automatically.

Switching the workspace

To ensure that a new workspace is configured correctly, switching the workspace must be done within the environment of EB corbos Starter Kit. This guarantees that the new workspace

- is placed on `C:/EB/corbos/starterkit_vx.x/`

- is configured and initialized, i.e. it has connection settings and source code mappings.

Prerequisite:

- EB corbos Starter Kit is installed.

Step 1
Open the command prompt.

Step 2
Navigate to the path where you have installed the EB corbos Starter Kit.

Step 3
Enter command:

switch-workspace.cmd

Step 4
To switch the workspace follow the procedure shown in the figure below.

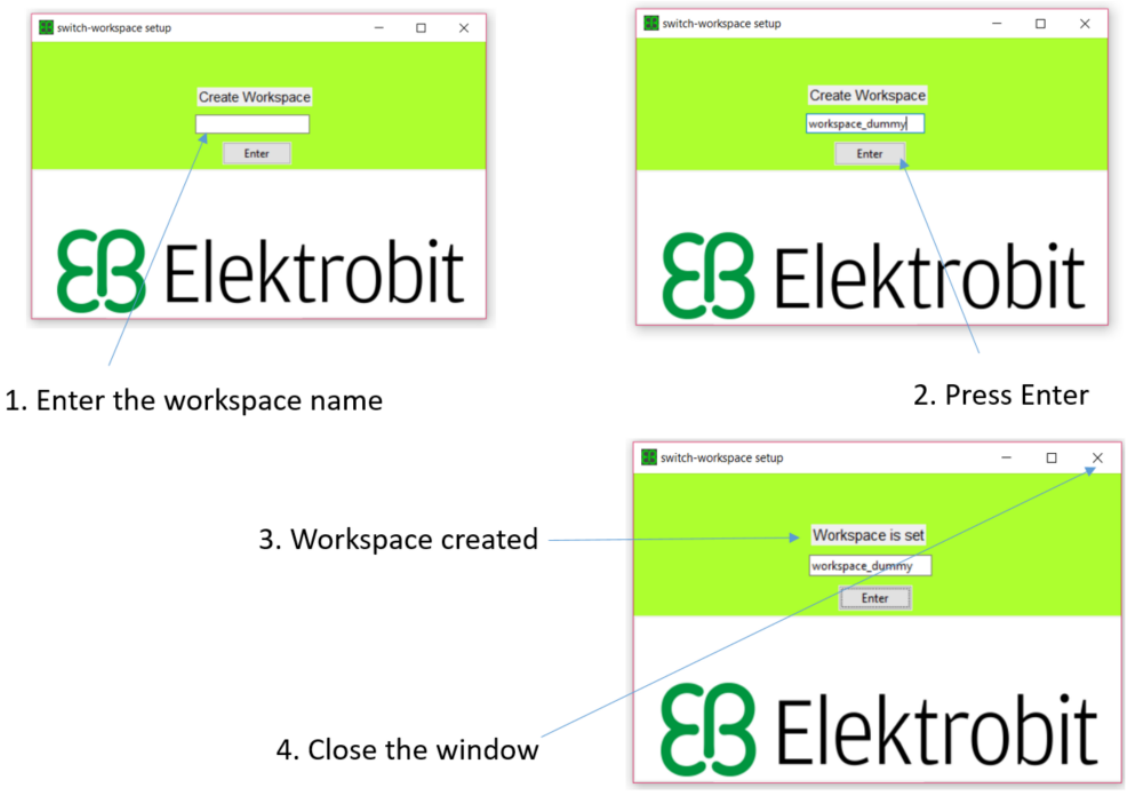


Figure 5.2. switch_workspace

WARNING

Menu items disabled

To prevent misuse, the menu items **Switch-Workspace ...** and **Restart** are disabled per default.

To enable the items, select in EB corbos Studio **Window > Perspective > Customize Perspective > tab Menu Visibility > File** .

5.3. Creating an AUTOSAR/C++ project

Creating an AUTOSAR/C++ project

Step 1
Select **File** → **New** → **Other** → **AUTOSAR Adaptive** → **Configuration project**.

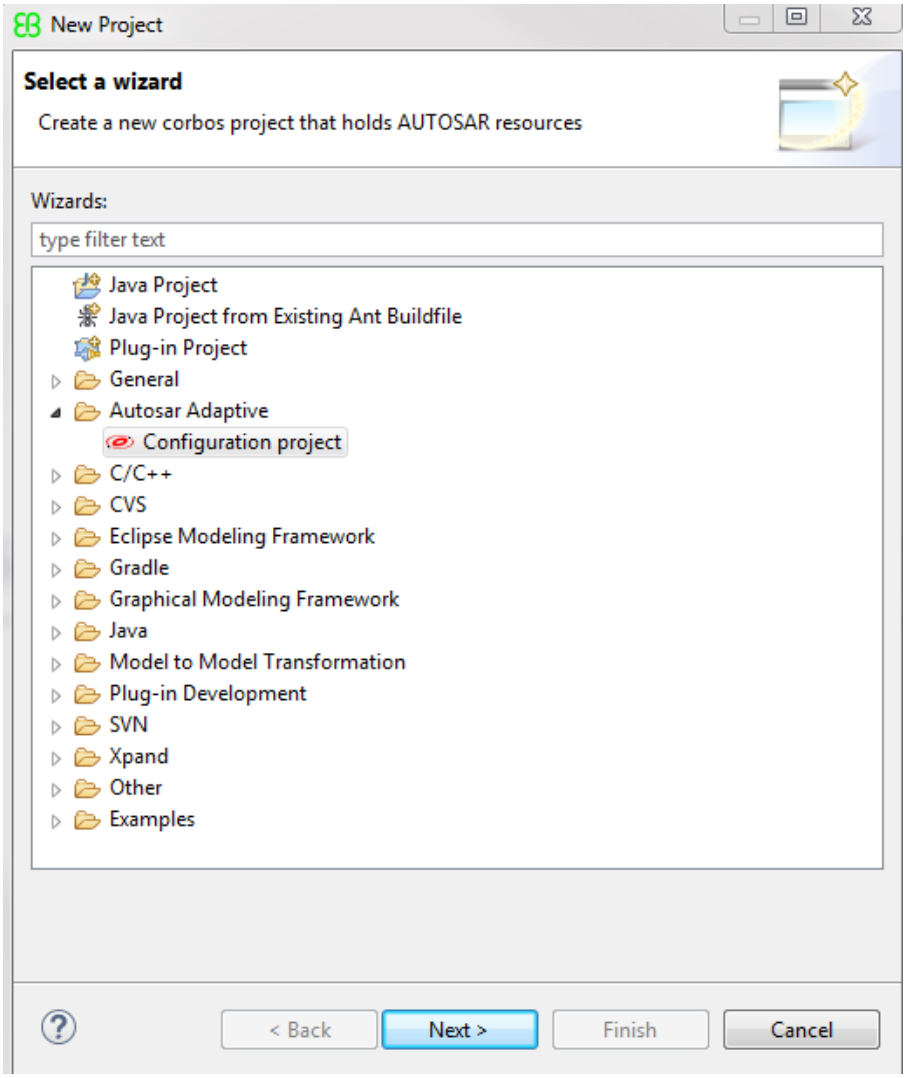


Figure 5.3. New project

Step 2

Enter a **Project name** and click **Finish**. Do not change the default settings.

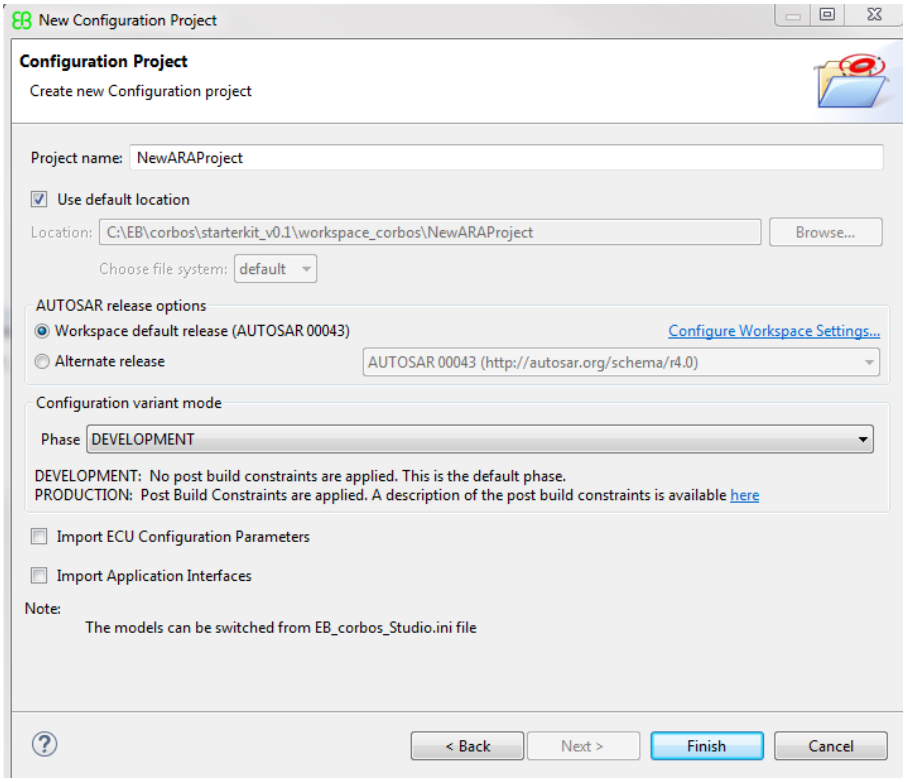


Figure 5.4. New project name

The new project is now visible in the project browser.

Step 3
Configure the new project. First close the EB corbos Studio and then open the command prompt.

Step 4
Navigate to the path where you have installed the EB corbos Starter Kit i.e. default at `C:/EB/corbos/starterkit_vx.x/`.

Step 5
Run the `configure-project.cmd`. Follow the steps as shown in the figure below.

Figure 5.5. configure_project

Step 6

Open the EB corbos Studio again and see the project is now configured. Now you can refresh the project browser and see what was added during the configuration. See also the description below.

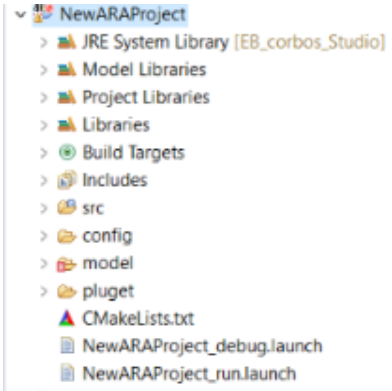


Figure 5.6. A configured new project

Step 7

Right click on the Ant show view window and select **Add Buildfiles...** → **NewARAProject** → **pluget** → **build.ant** → **Ok**.

Step 8

Edit `build.ant` file in your pluget folder inside project according to the project requirement.

During the configuration the following changes are applied:

- Converting to a C++ project.
- Adding build targets.

Build target	Description
0_generate	Runs `jsongen.py` and generate `someip_-${project.name}.json` for ara_Com project.

Build target	Description
1_remote_compile	Cross-compiles your sources with the ARA SDK hosted on the development image.
2_deploy	Copies the binary files, application configurations & container configuration to the QEMU target specified in `config/target_manifest.xml`. Execute this task before running or debugging the application.
3_make_clean	Runs 'make clean' in the projects' build configuration on the development image.
4_clean_deployment	Removes all deployed container files from QEMU.
5_test_coverage_report	Used to generate code coverage report.
6_containers_status	Used to check the running containers status from QEMU target.
7_start_container	Used to start the container if already deployed in QEMU target.
8_stop_container	Used to stop the container if already running in QEMU target.

- Adding directory `config`.

File	Description
target_manifest.xml	You can change this file to configure basic information about the targetType, targetName, machinePath etc.
rootfs.xml	Use and configure this file in order to deploy project specific files into application container and/or target root filesystem and also to configure daemon container restart.

- Adding directory `pluget` which contains the pluget generators for ara-com, ara-em and ara-pm which generates the related configurations out of the AUTOSAR arxml.
- Adding directory `model` which contains default arxml files. Adapt this files to your needs.
- Adding template `CMakeList.txt`. Adapt this file to your needs.
- Updating the launch configuration file `NewARAProject_run.launch` and `NewARAProject_debug.launch` according to the target_manifest.xml during the *2_deploy* task.

TIP

Command to run Pluget from windows command line

call %STARTERKIT_DIR%\EB_corbos_Studio\RunPluget.cmd -data %CORBOS_WORKSPACE_DIR% -project %CORBOS_PROJECT_NAME% -pluget pluget\%PLUGET_NAME%.pluget

For user reference the command looks like for demo project : **call EB_corbos_Studio\RunPluget.cmd -data C:\EB\corbos\starterkit_vx.x\workspace_corbos -project demo -pluget pluget\araPmPhmEmManifestGen.pluget**

TIP

Command to run build.ant from windows command line

call %STARTERKIT_DIR%\EB_corbos_Studioc.exe --launcher.suppressErrors -nosplash -application org.eclipse.ant.core.antRunner -data %CORBOS_WORKSPACE_DIR% -buildfile %CORBOS_WORKSPACE_DIR%\%CORBOS_PROJECT_NAME%\pluget\build.ant

For user reference the command looks like for demo project : **call EB_corbos_Studio\EB_corbos_Studioc.exe --launcher.suppressErrors -nosplash -application org.eclipse.ant.core.antRunner -data C:\EB\corbos\starterkit_vx.x\workspace_corbos -buildfile C:\EB\corbos\starterkit_vx.x\workspace_corbos\demo\pluget\build.ant**

5.4. Developing an Adaptive Application

This chapter describes the relevant steps to create an Adaptive Application from scratch.

1. Creating an AUTOSAR model
2. Editing the target manifest
3. Usage of rootfs.xml
4. Running the generator
5. Developing application code
6. Deploying to a target
7. Running and debugging your application

Some steps are described in more detail in Chapter 7, “Demo applications“.

5.4.1. Creating an AUTOSAR model

To create an AUTOSAR model, the following steps are needed.

1. Create an application design.
2. Describe the system topology.
3. Create and edit `json` files that are not default files generated by the generator.

For more details on how to create a model for a sample application, see Chapter 7, “Demo applications“

5.4.2. Editing the target manifest

The Figure 5.7, “target_manifest.xml” in folder `config`, shows an exemplary content that is used for demo applications in Chapter 7, “Demo applications“.

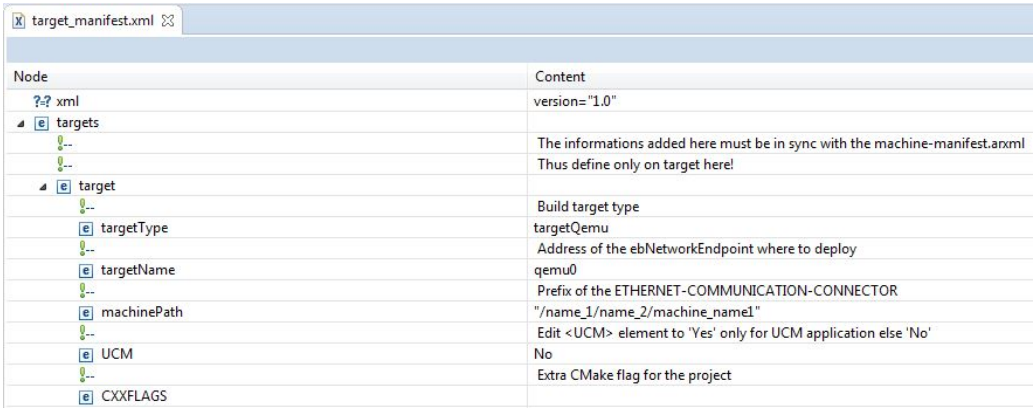


Figure 5.7. target_manifest.xml

NOTE

The `IP` and `machinePath` must be the same as in the `machine.arxml`.

- `machinePath` is necessary for the json generator to generate the SOME/IP configuration file `vsomeip_-{MyProject}.json`.
- `targetName` is the name of the qemu where you want to deploy for example qemu0 or qemu1.

5.4.3. Usage of rootfs.xml file

The Figure 5.8, “rootfs.xml” in folder `config` is used to configure in order to deploy project specific files into application container and/or target root filesystem and also to configure daemon container restart.

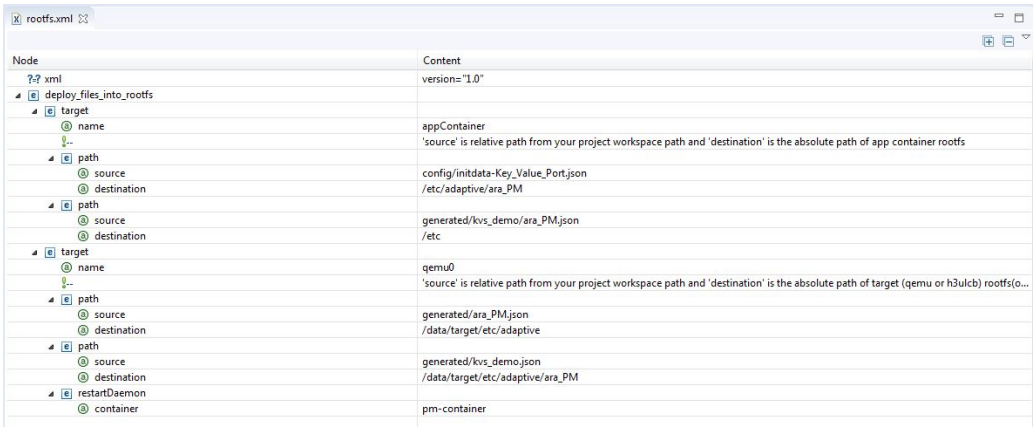


Figure 5.8. rootfs.xml

- `target` is having an attribute `name` which is used to specify where user wants to deploy any project specific files. It should be either in application container root filesystem or in target root filesystem.
  - name="appContainer" : to deploy file into application container root filesystem.
  - name="<targetName>" : to deploy file into target root filesystem. <targetName> can be qemu0, qemu1, etc. and it should be same as defined in Figure 5.7, “target_manifest.xml”
    - `path` is having an atrributes `source` and `destination` used to specify source and destination path
      - source : relative path from your project workspace.
      - destination : absolute path of application container rootfs or target rootfs.
    - `restartDaemon` can be used only under target name is configured to qemu or hardware name. It can not be used under target name configured as appContainer. It can be configured multiple times in order to restart multiple daemon containers.
      - container : daemon container name.

5.4.4. Running the generator

Execute the build target **0_generate** for ara::com related applications. It triggers `jsongen.py` and creates file `someip_{MyProject}.json`.

Execute the other generators using Ant build view from EB corbos Starter Kit.

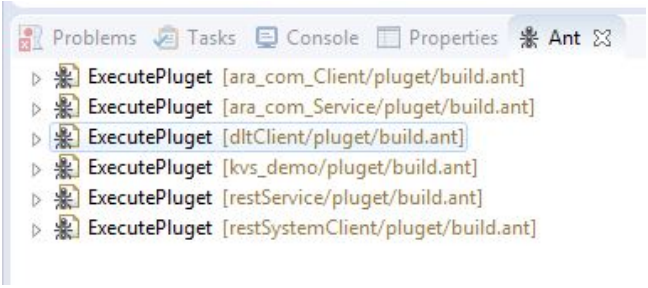


Figure 5.9. Ant build view in EB corbos Starter Kit

AraComBindingGenerator.pluget
: Fed with ara-com related arxml files from directory `model` as well as machinePath from `target_manifest.xml` to create the ara::com related configurations.

araPmPhmEmManifestGen.pluget
: Fed with ara-em, ara-phm and ara-pm related arxml files to create the container configuration file along with persistency(ara-pm) configuration files.

NOTE

All ADG generators are placed at <install dir>\starterkit_vx.x\templates\config\tools \adg_plugets.zip. this will help user to use the ADG generators in the project based on application requirements.

If user is using AraComManifestGenerator.pluget instead of 0_generate, someip manifest json file is generated inside the <project>\generated\config folder. hence AraComManifestGenerator.pluget has to to be copied to the <project>/pluget/ from <install dir> \starterkit_vx.x\templates\config\tools\adg_plugets.zip

5.4.5. Developing application code

1. Edit `CMakeLists.txt` according to your project needs.
2. Implement against the generated binding interfaces.
3. Run build target **1_remote_compile** to cross-compile your application with the provided SDK.

5.4.6. Deploying to a target

Before you deploy to a target, fill out the `target_manifest.xml` as explained in Editing the target manifest

If requires, update the rootfs.xml file as explained in Usage of rootfs.xml

1. To deploy a demo application into the target eg. QEMU, run the script **2_deploy** (as shown in figure below).

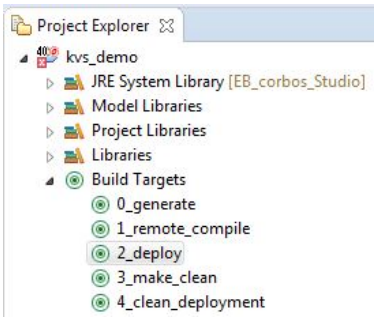


Figure 5.10. Deploy an application

This task copies the binary and configuration files to the target that is defined in `target_manifest.xml`.

NOTE

If QEMU is restarted, then a previously deployed container will not start automatically and you cannot run applications directly from EB corbos Studio. In this case you need to deploy it again using build target 2_deploy from EB corbos Studio or you can manually start the container from the PuTTY terminal that is connected to the respective QEMU using command **systemctl start container-<projectname>_0.service**.

5.4.7. Running and debugging your application

Each project has `{MyProject}_run.launch` and `{MyProject}_debug.launch` that describes the launch configurations in its project explorer.

5.4.7.1. Running the existing demo applications

Running the existing demo applications

Step 1
Go to **Run Configurations** → **C/C++ Remote Application** → {MyProject}_run .

Step 2
Tick check box **Skip download to target path.** → **Apply** → **Run**.

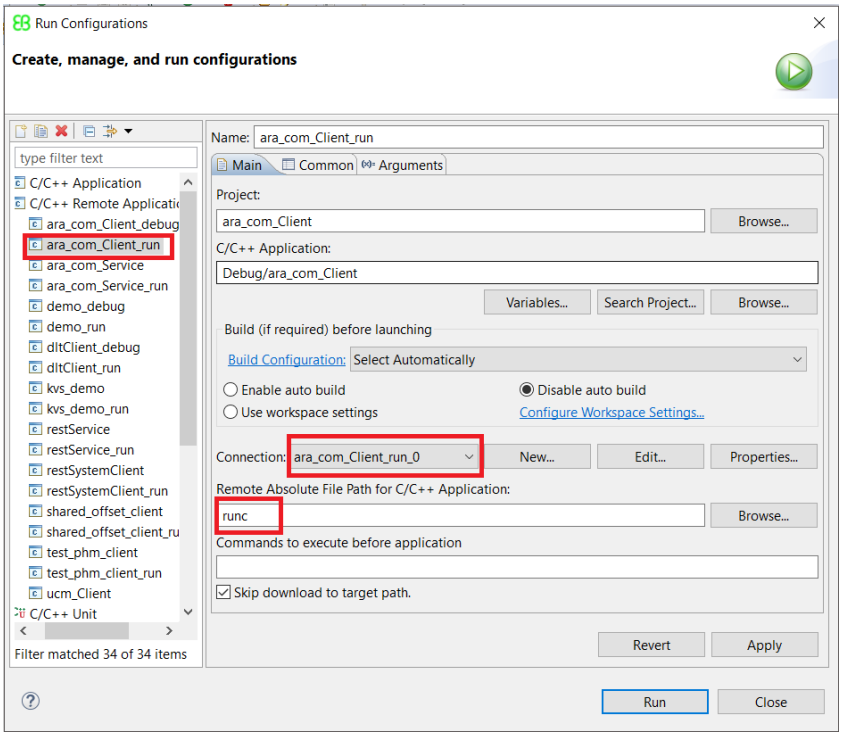


Figure 5.11. Run application: Edit connection

5.4.7.2. Running a new application

Running a new application

Step 1
Go to **Run Configurations** → **C/C++ Remote Application** → {MyProject}_run → **Connection** → **New** → Choose connection type *SSH* → **OK**.

Step 2
Type *Connection name* as your {MyProject}_run → Copy the IP address for targets as mentiones in sections Table 5.1, “Connection settings” based on targetName element containing in target_manifest.xml file. target_manifest.xml file is available at your project workspace config directory → Paste it in Edit Connection *Host* → *User* is *root* → Tick the radio button for *Password based authentication* → **Finish**.

Step 3
Select the {MyProject}_run from the connection dropdown.

Step 4
Tick the check box **Skip download to target path** → **Apply** → **Run**.

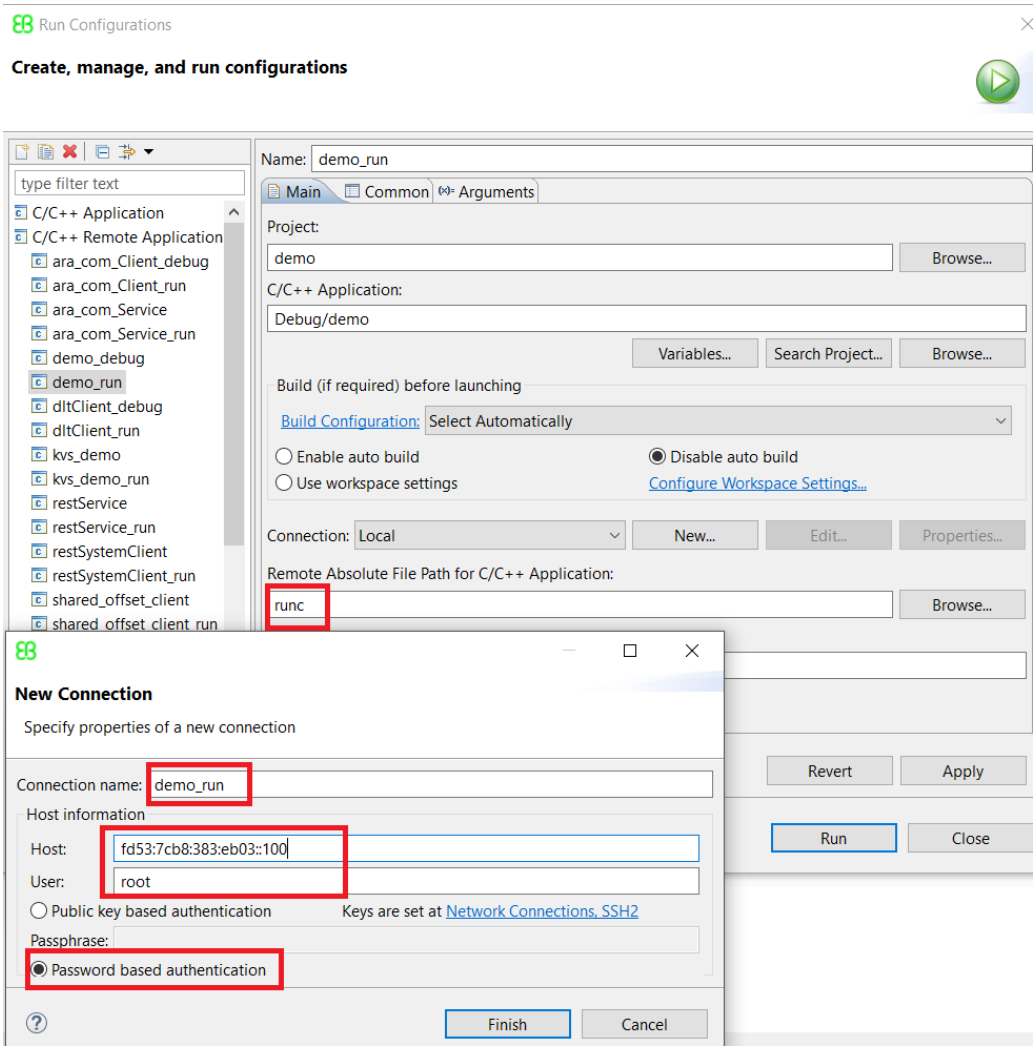


Figure 5.12. Run new application: New connection

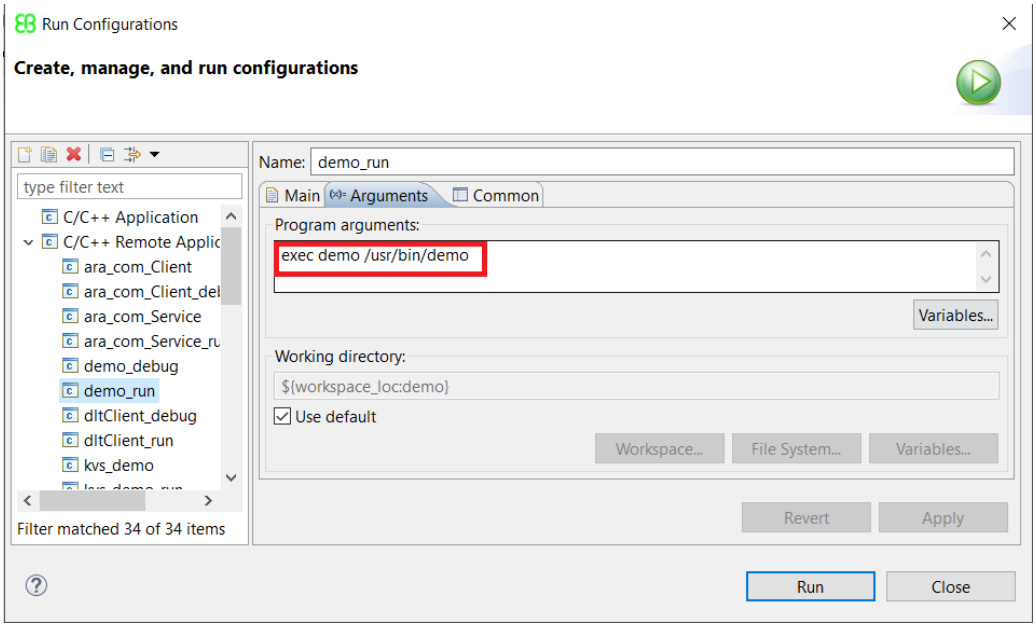


Figure 5.13. **Setting Arguments**

5.4.7.3. Debugging the demo application and New Application

Debugging the demo application

Step 1
Goto **Toolbar** click on **Debug** icon →select **Debug_configurations→{MyProject}_debug** → **edit**→in **host** change IP address according to **containerIP.txt** file. Which is present in project workspace.

NOTE

For New applications do the following :

Goto **Toolbar** click on **Debug** icon →select **Debug_configurations→{MyProject}_debug** → **new** →choose connection type **ssh**→connection name {MyProject}_debug→in **host** change IP address according to **containerIP.txt** file. Which is present in project workspace.→user **root**→click on **Password based authentication**→**Finish**→Now select the connection which you have created just now from **Connecton**.→set the arguments →**Apply**→**Debug**

Step 2
then →**Finish**→**Apply**→**Debug**→**yes**→**yes**.

Step 3
The debug perspective from EB corbos Studio opens up. Now click the **Resume** from Toolbar entries.

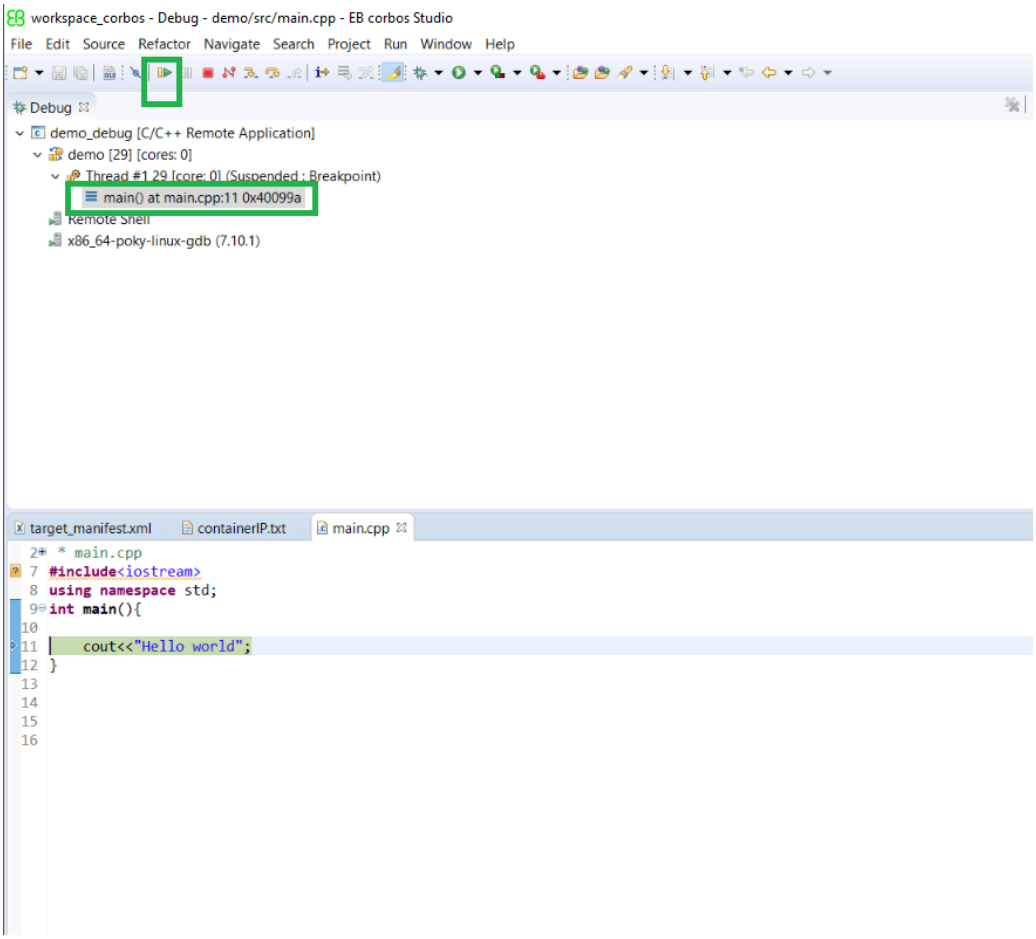


Figure 5.14. Debug Adaptive Application from launch configuration

5.5. Connecting to targets

Use PuTTY to connect to the targets. The Table 5.1, “Connection settings” lists the connection settings for the predefined images.

Targets	Host name (or IP)		Port	login	pass-word
	IPv4	IPv6			
corbos-starterkit-vm	10.10.10.1	fd80:bd06:320b:eb03:00e-b:0010:0000:0001	22	adaptivecore-build	1
qemu_runtime	10.10.10.2	fd80:bd06:320b:eb03:00e-b:0010:0000:0002	22	root	1
qemu-sdk	10.10.10.3	fd80:bd06:320b:eb03:00e-b:0010:0000:0003	22	root	1

Targets	Host name (or IP)		Port	login	pass-word
	IPv4	IPv6			
qemu0	10.10.10.100	fd80:bd06:320b:eb03:00eb:0010:0000:0100	22	root	-
qemu1	10.10.10.101	fd80:bd06:320b:eb03:00eb:0010:0000:0101	22	root	-

Table 5.1. Connection settings

6. Working with EB corbos Starter Kit

6.1. Restoring the default configuration

Restoring the default configuration

Step 1
First close EB corbos Studio then open the directory where you unzipped the EB corbos Starter Kit files.

Step 2
Open the Windows command prompt.

Step 3
Enter the command `configure-project.cmd`. Follow the steps as shown in below figure.

Figure 6.1. configure_project

Step 4
Open EB corbos Studio again and check whether the project is now configured. Now you can refresh the project browser and see what was added during the configuration.

WARNING **Loss of data**

All settings and resources are overwritten by the default settings, including the CMakeList.

6.2. Running the applications directly from PuTTY

Running the applications directly from PuTTY

Step 1
Log in the QEMU target.

Step 2
Start the container if it is not running already.

TIP **Commands to run from qemu (for containers)**

`runc list`	Checks running containers
`systemctl start container-<projectname>_0.-service`	Starts containers
`/sbin/cont-ctrl start container_name`	Starts containers
`systemctl stop container-<projectname>_0.-service`	Stops containers
`runc kill container_name`	Stops containers
`/sbin/cont-ctrl stop container_name`	Stops containers
`runc exec container_name ifconfig`	Checks the container IP address

Step 3
Connect to the container via SSH. `ssh root@container_ip`. When asked enter **yes**.

TIP The container IP address should be available in the workspace project location in file `containerIP.txt`

Step 4
To run the application inside a container, execute the application binary.

6.3. Restart QEMU base docker container using PuTTY

Restart QEMU base docker container using PuTTY

Step 1
Delete `built_images` directory from `C:\EB\corbos\starterkit_vx.x\qemu\qemu0` and `C:\EB\corbos\starterkit_vx.x\qemu\qemu1` path.

TIP

Here `C:\EB\corbos\starterkit_vx.x` is default install path of EB corbos Starter Kit

Step 2
Use PuTTY to connect **corbos-starterkit-vm**. User **adaptivecore-build** password **1**

Step 3
run the following commands:

docker kill qemu_runtime

echo y | docker system prune

docker run --restart=always -itd --name qemu_runtime -v /media/sf_share:/media/sf_share --sysctl net.ipv6.conf.all.disable_ipv6=0 --privileged qemu_base_image:<version> bash -c "/sbin/init"

Step 4
Check the container is started or not by giving command **docker ps** . It will show qemu_runtime container details.

TIP

In above command <version> is starterkit version Ex:1.0 and qemu_runtime is the qemu base container name.

6.4. Tuning the RAM usage

You can change the RAM according to the usage or general performance. Usually 2GB of RAM are to the corbos-starterkit-vm image.

Tuning the RAM usage

Step 1
Use **htop** command in corbos-starterkit-vm to check whether your system is running normal or you need to increase the performance or vice-versa. Check whether your image is using swap memory. In this case increase the RAM.

Step 2
Change the settings in VirtualBox. Go to VirtualBox → **Settings** → **System** → **Motherboard** → **Base Memory**.

6.5. Enable multiple cores

You can change the core according to the usage or general performance. Usually 4 cores are assign to the corbos-starterkit-vm image.

Tuning the core usage

Step 1
Use **htop** command in corbos-starterkit-vm to check whether your system cores utilization is normal or you need to increase the performance or vice-versa. In this case increase or decrease the core.

Step 2
Change the settings in VirtualBox. Go to VirtualBox → **Settings** → **System** → **Processor** → **Processor(s)**.

NOTE

If user sytem goes slow after installing EB corbos Starter Kit and core is fully utilize then decrease the cores in VirtualBox uses by corbos-starterkit-vm but this will impact the EB corbos Starter Kit performance.

6.6. Parallel console for Client-Server based Applications

Steps to enable Parallel Console

Step 1
Click on the console button to create new console like given below picture.

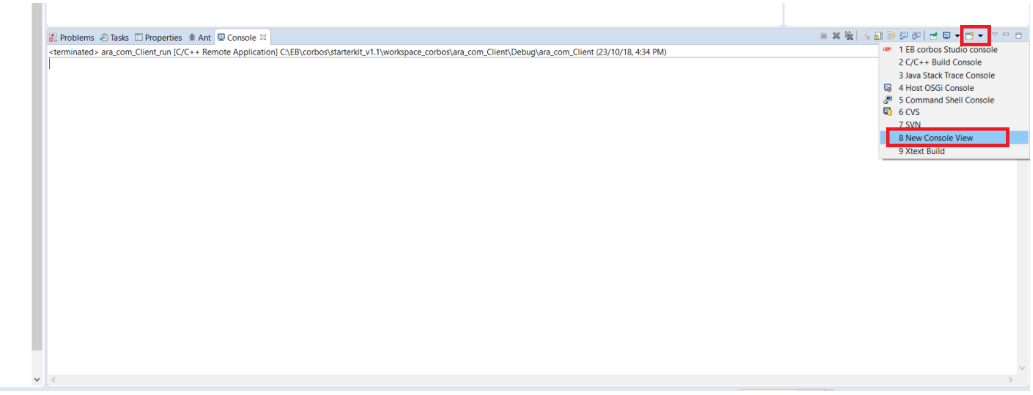


Figure 6.2. New console Creation

Step 2
Two consoles will get opened and placed one after the other, Now Drag and Drop the newly created console to Downward direction, you will get to see two consoles opened in parallelly.

Step 3
Run client and server applications, Goto console select the running application for each console. Like shown below.

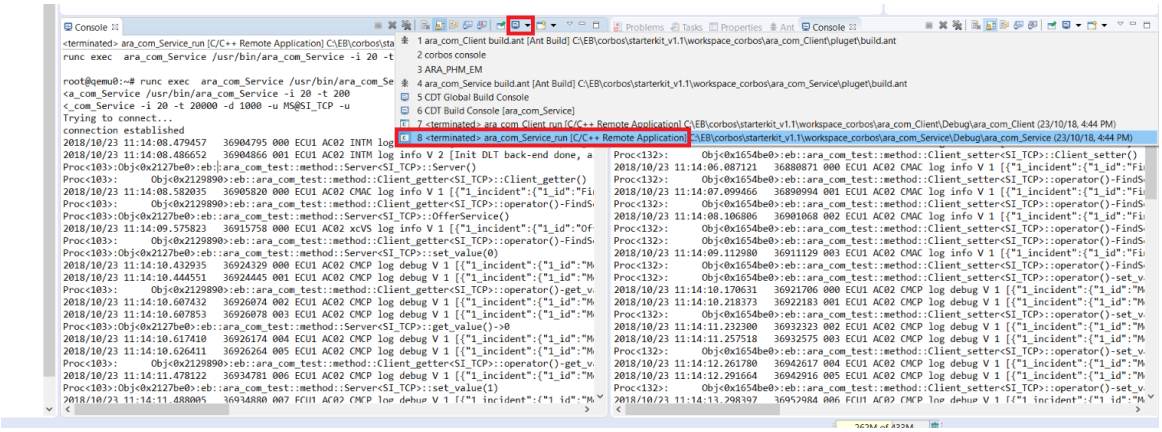


Figure 6.3. Parallel console

6.7. Configure project to use Gtest and Gmock

User has to follow the below steps to use gtest and gmock in project.

Gtest And Gmock Usage

Step 1
User has to use the gtest and gmock API's in the source code of the project.

Step 2
Add find_package(GTest REQUIRED) line in `CMakeLists.txt` and ${GTEST_LIBRARIES}, gmock and pthread in the TARGET_LINK_LIBRARIES section of the `CMakeLists.txt` of the project.

Step 3
Compile and run the project, output will be shown on the EB corbos Studio console.

6.8. Integration support for Doxygen

Doxygen is integrated into EB corbos Studio as an Eclipse plugin `Eclox`. The plugin includes the following features:

- Graphical editing and creation of doxyfile
- Easy generation of documentation with right mouse click

Demo application `demo` contains predefined Doxyfile named `demo.doxyfile`. Double-clicking `demo.doxyfile` opens a Doxyfile configuration editor view. To build documentation, right click on `demo.doxyfile` and select @ Build Documentation. The documentation is created into defined output directory demo\doxygen\.

NOTE

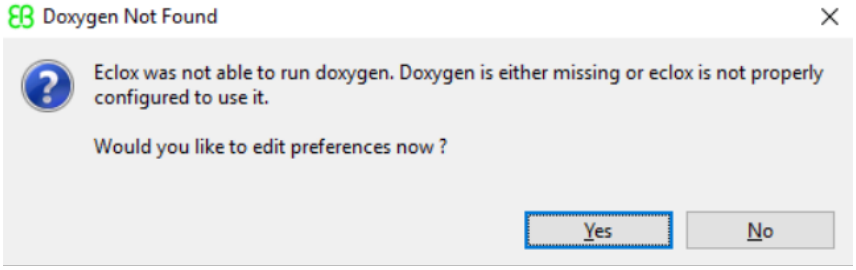


Figure 6.4. Choose among available doxygen versions

To solve this, please click on yes to go to the Preferences. Here go to Doxygen and select bundled instead of unknown. After click on OK, you can use doxygen.

Here's how to add Doxygen documentation build support to your project:

Step 1
Open right-click mouse menu for your project's root directory → select New → Other... → Other → Doxyfile.

Step 2
Give name for the new Doxyfile.

Step 3
Run client and server applications, Goto console select the running application for each console. Like shown below.

Step 4
Refresh your project and Double-click the created Doxyfile to open the Doxyfile configuration editor view.

Step 5
Set the configuration as desired, then save the file.

Step 6
To add support for Doxygen comments into C/C++ editor, go to Windows → Preferences → C/C++ → Editor → on the right side down select Workspace default - Doxygen → Apply → OK.

Step 7
Now when you open a source file into C/C++ editor and add a new comment by typing /** and hitting ENTER then Doxygen type of commenting is created. Create the desired Doxygen commenting in all your source files.

Step 8
Build documentation by open right mouse button menu for Doxyfile and select @ Build Documentation. The documentation is created into the output directory defined in Doxyfile configuration.

6.9. Integration support for code coverage

Tools integrated for code coverage support are gcov,lcov and genhtml.

User can make use of these tools by implementing test cases for the source code. For reference you can use `demo` project which has implemented test cases using gtest.

Prerequisite:

- Run the application before running the build target 5_test_coverage_report.

To calculate and display code coverage, a build target 5_test_coverage_report is used and as a result you will find a folder name code_coverage under project directory. You can double click on `index.html` to view report in EB corbos Studio or you can view in any internet browser.

6.10. Integration support for ReqM2 tool

ReqM2 is a tool for generating requirement specification reports. For reference you can use "demo" project which showcases the usage of ReqM2.

To generate report run build target 6_reqm2_generate. Which calls the `reqm2_generate.bat` script present inside `starterkit_vx.x\tools` folder. This script create ReqM2 folder inside project directory which contains generated report. Double click on `cfg.html` which present inside ReqM2 folder to view the report in EB corbos Studio or you can view in any internet browser.

Here's how to add ReqM2 report generation support to your project:

Step 1
Put the configuration file in project directory and rename it to cfg.xml

Step 2
Open EB corbos Studio select project. Right on Build Target -> Create... and to add new build target refer the below image

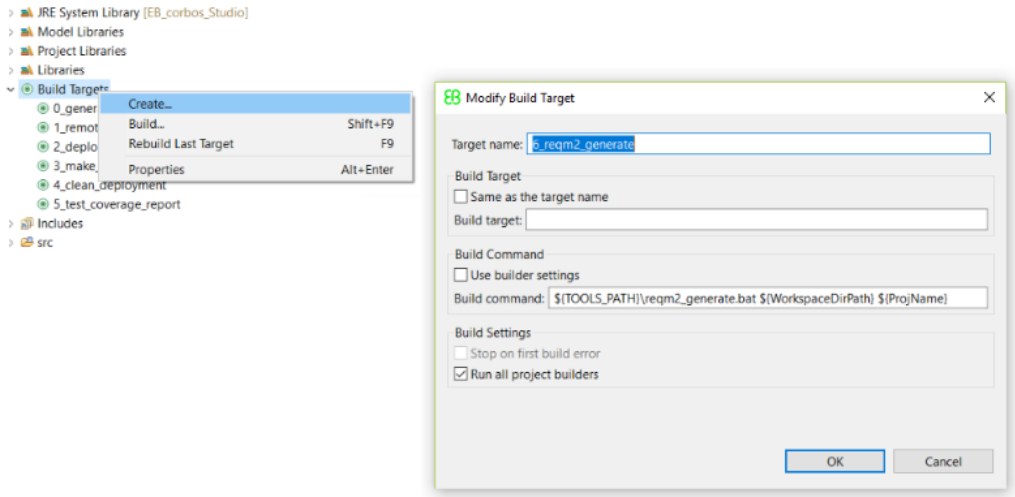


Figure 6.5. Add ReqM2 support

NOTE

Change the `reqm2_generate.bat` script present inside `starterkit_vx.x\tools` according to the requirement

After Adding build target double click on 6_reqm2_generate and it will generate report inside ReqM2 folder. Double click on cfg.html which present inside ReqM2 folder to view the report.

For more example go to `corbos\common_tools\ReqM2\demos`

6.11. Core_dump file generation for Adaptive Applications

Adaptive applications in corbos_studio_starterkit are compiled in debug and release mode. In debug mode, applications are compiled with debug symbols (using '-g' flag in compilation).

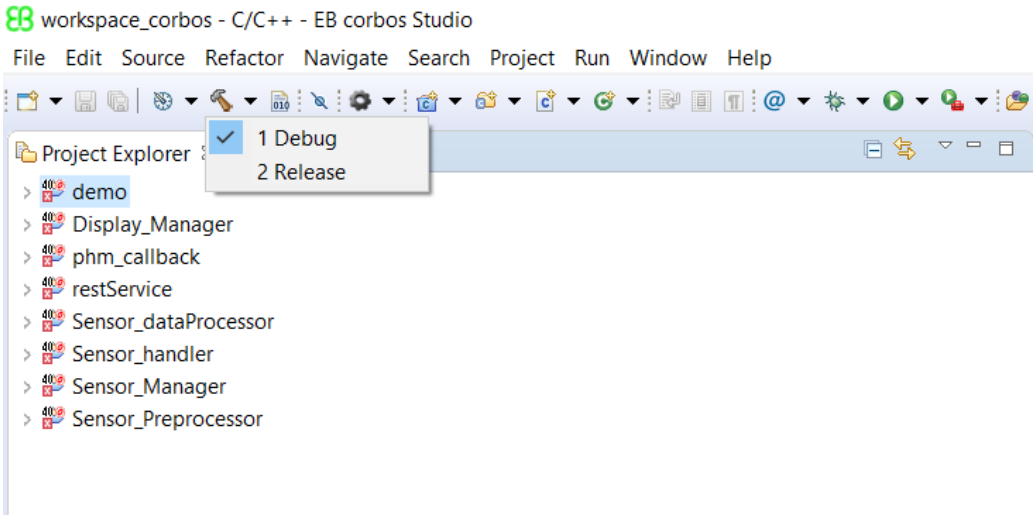


Figure 6.6. Compilation modes

Core_dump file is generated for crashed application that are compiled in debug mode only. It can be found at /data/coredump/current path of the respective application container. Connect to Application container through Putty or from Qemu using following command **ssh root@<application_container_ip>**.

NOTE

If the folder structure "/data/coredump/current" is not present in application container, user has to create the missing folders in the folder structure to get the coredump files.

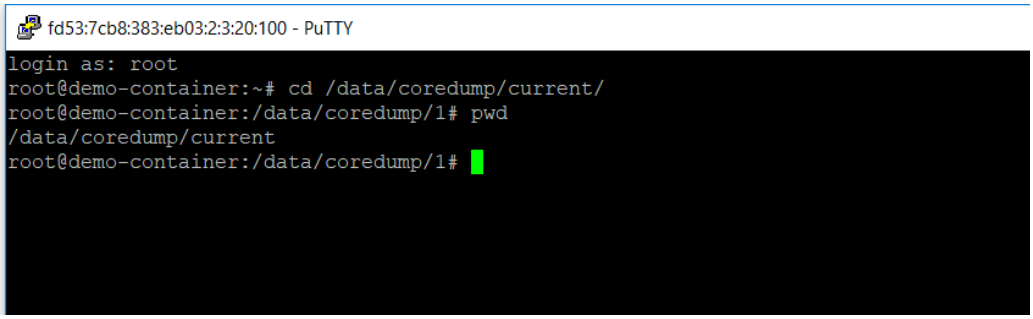


Figure 6.7. Coredump location

6.12. Running containers status show from EB corbos Studio

User can the see the daemon containers status and deployed Adaptive applications status by just double click on **x_container_status** from the build targets of the project. see below image for reference image.

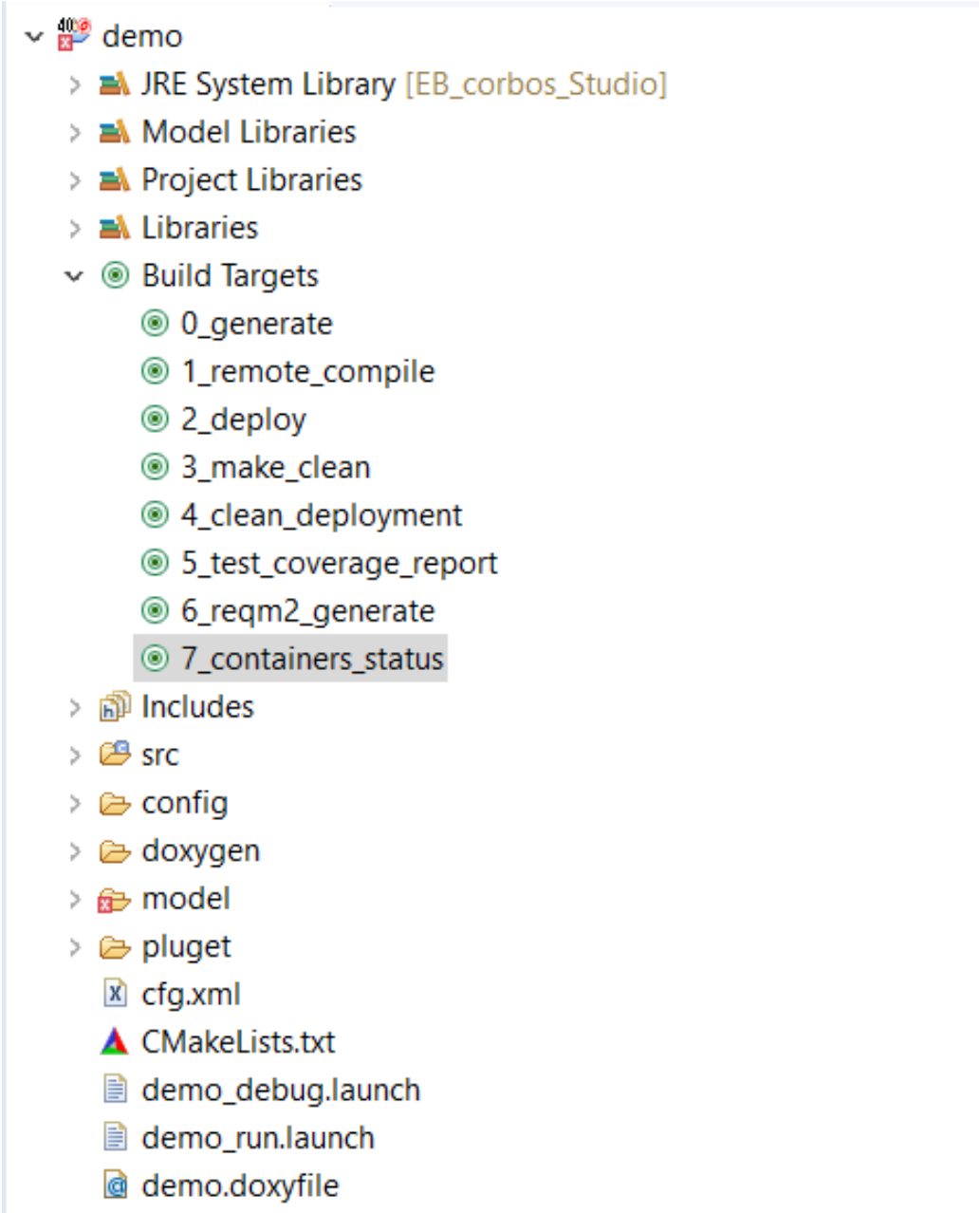


Figure 6.8. Build targets

After clicking double click on **x_containers_status** like following output will be shown on studio console.

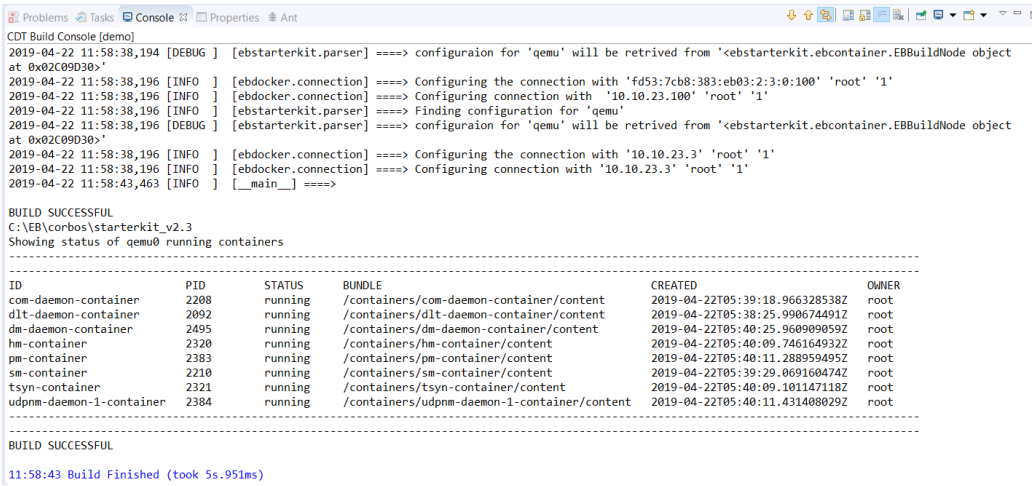


Figure 6.9. Containers status output

7. Demo applications

7.1. Demo applications scope

The purpose of the demo applications is to provide an Adaptive AUTOSAR application example which offers a service using ARA APIs and intra/inter-ECU communication via SOME/IP. All the demo applications are developed using the EB implementation of Adaptive AUTOSAR SDK (APIs) and configurations.

7.2. Traffic Sign Scenario

Traffic sign scenario described using six applications :

- Sensor_handler
- Sensor_Preprocessor
- Sensor_Manager
- Sensor_dataProcessor
- restService
- Display_Manager
- phm_callback(Not in image below)

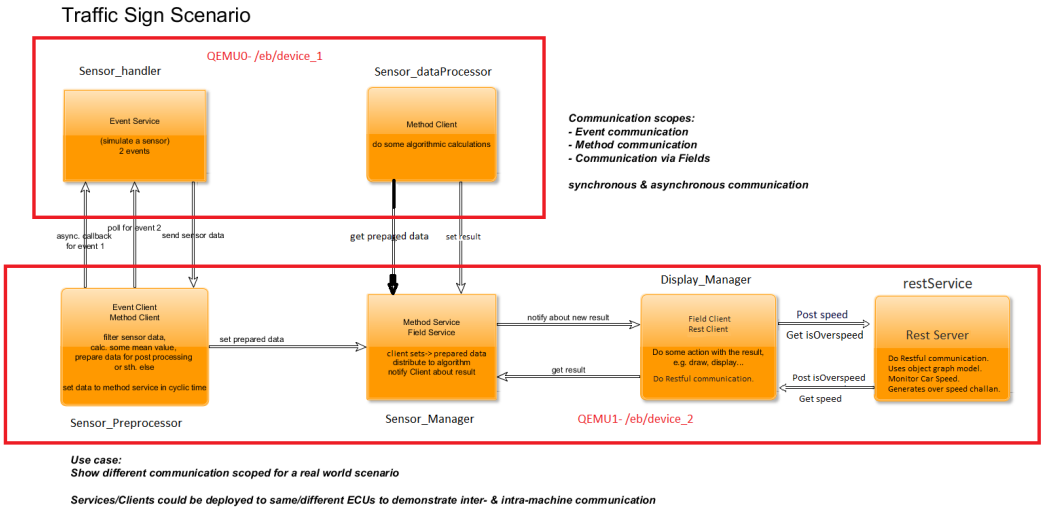


Figure 7.1. Traffic Sign Scenario - demo applications

- Sensor_handler : Event service is sending data to Sensor_Preprocessor. It stores camera calibration data into persistency database and also implements the alive supervision use case of HM. it has synchronized master and offset master tsync features implemented.
- Sensor_Preprocessor : It has event client which receives data from Sensor_handler and finds the status based on data received and forwards the status to Sensor_Manager through method client. It also implements the logical supervision use case of HM.
- Sensor_Manager : Method server receives data from Sensor_Preprocessor. Field server will notify the data availability and send the data to Display_Manager when queried. it has pure local tsync feature implemented.
- Sensor_dataProcessor : Method client queries the data from Sensor_Manager and shares it with field client. field client set the data in Sensor_Manager(field server). It also stores vehicle high speed data into persistency database. it has synchronized slave and offset master tsync features implemented.
- Display_Manager : Field client receives the notification from sensor_Manager and queries for data. it has rest client implementation.
- restService : Rest Service will receive data from Display_Manager(rest client) and monitor the data. if the car speed is exceeded speed limit, it will generate the challan and send it to the Display_Manager(rest client).
- phm_callback: It registers the callback function with PHM using the PhmCallbackClient interface. Whenever the the Supervised app, Sensor_Manager in this case(refer section: 'ara::phm configuration and use case'), fails to meet the timing deadlines defined in application manifest (PHM_Sensor_Manager.json), PHM will invoke the callback function in phm_callback application.

There are three types of communication processes:

- Intraprocess communication: Instantiate in same process
- Interprocess communication: Instantiate in different processes connected to the same com_daemon
- SOME/IP communication (according to the daemon's manifest): Instantiate in different processes connected to different com_daemon instances (here or in a different QEMU)

Servers must select a unique <instance> string. Clients specify the <instance> string of the server they want to talk to.

If communication shall take place over SOME/IP on network, then the <instance> string must match an SOME/IP instance deployment entry in the SOME/IP manifest file provided to the com_daemon during start-up. This file provides the mapping to the related SOME/IP protocol elements.

7.2.1. ara::com event communication

- Event Server – Transmits events periodically.

- Event Client – Receives events transmitted by event server and prints them to console.

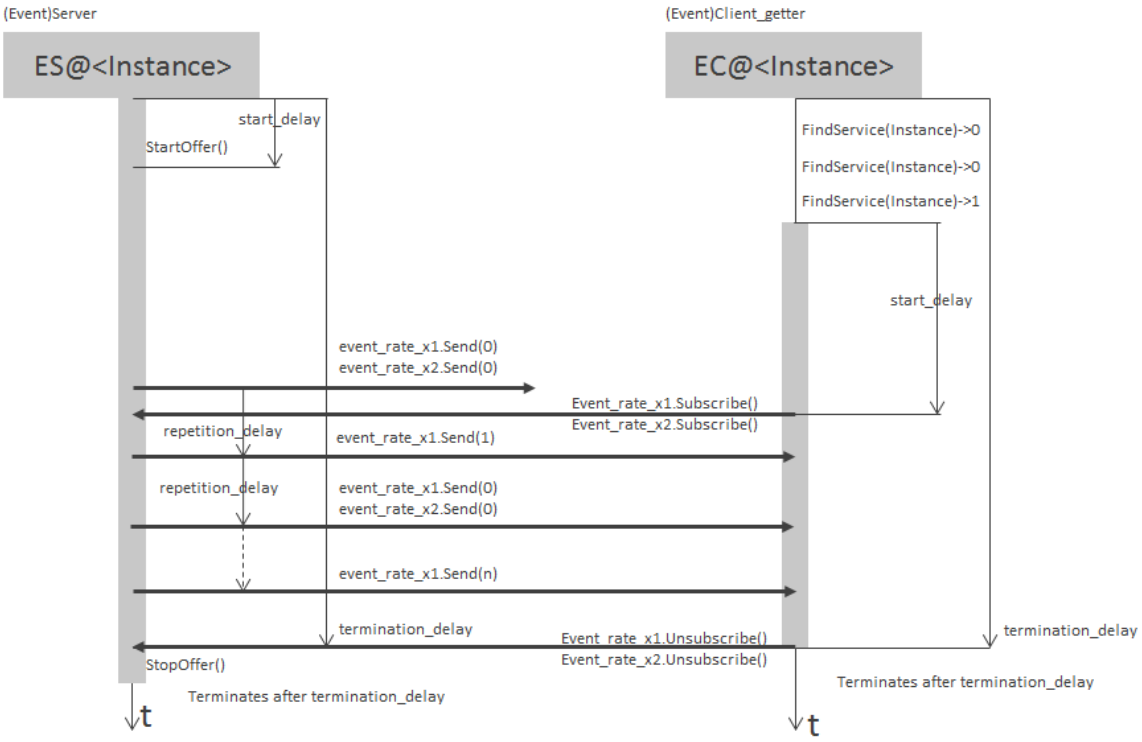


Figure 7.2. ara::com: Event

7.2.2. ara::com method communication

- Method Server – Provides a getter and a setter method for storing/retrieving a value.
- Method Client Setter – Periodically calls the setter method of Method Server with an increasing value.
- Method Client Getter – Periodically calls the getter method of Method Server and prints the result to console.

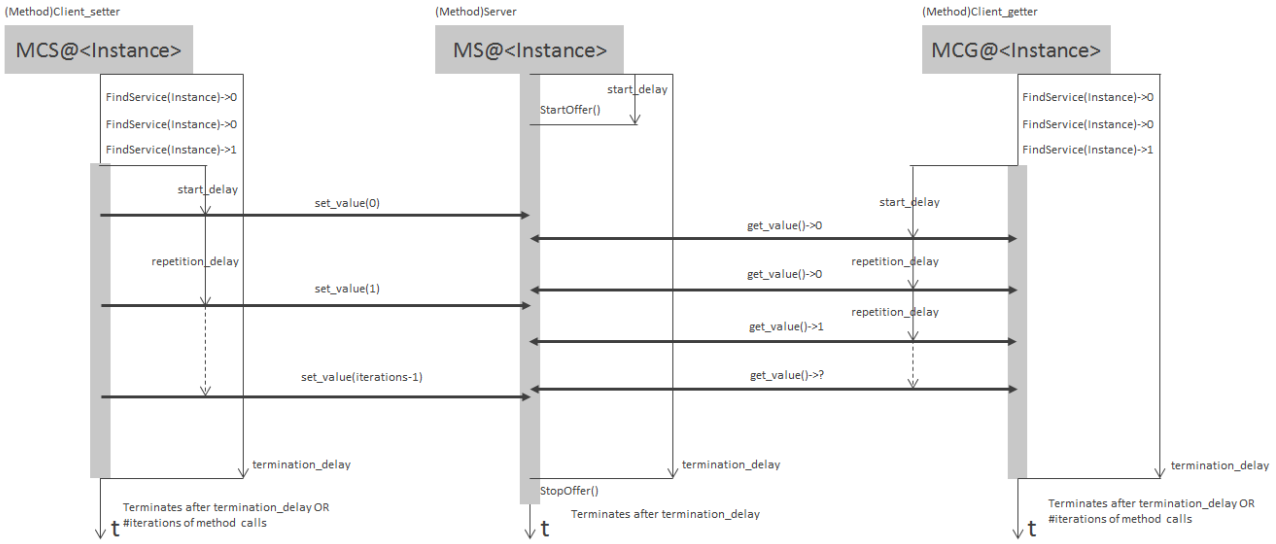


Figure 7.3. ara::com Method

7.2.3. ara::com field communication

- Field Server – Same behavior as Event Server.
- Field Client – Same behavior as Event Client.

7.2.4. ara::pm configuration and use case

7.2.4.1. Sensor_handler

The arxml file `Sensor_handler_em.arxml` avaiable in model folder of Sensor_handler project is configured of PM configuration along with EM(Execution Manager) configuration.

You need to run the pluget `araPmPhmEmManifestGen.pluget` in order to generate Persistency configuration json files into generated folder. As a result of this you will find a generated folder having :

- Sensor_handler\generated\ara_PM.json will get deploy at PM daemon container at path /etc/adaptive.
- Sensor_handler\generated\Sensor_handler.json will get deploy at PM daemon container at path /etc/adaptive/ara_PM.
- Sensor_handler\generated\Sensor_handler_Key_Value_Port will get deploy at PM daemon container at path /data/target/adaptive/ara_PM.
- Sensor_handler\generated\Sensor_handler\ara_PM.json will get deploy to Sensor_handler application container at path /etc.

Initial database configuration file made available at Sensor_handler\config\initdata-Key_Value_Port.json and it will get deploy into Sensor_handler application container at path /etc/adaptive/ara_PM.

Source and destination path to deploy into Sensor_handler application container can be defined in file `rootfs.xml`.

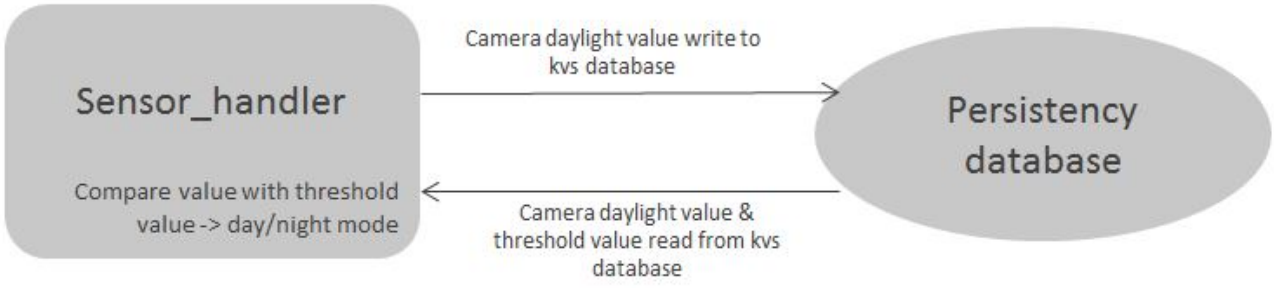


Figure 7.4. ara::pm usecase in Sensor_handler

In Sensor_handler, random generated camera daylight value gets overwritten into persistency database and then use the same database to read threshold value and daylight value and then compare it and decides camera day mode or night mode calibration.

7.2.4.2. Sensor_dataProcessor

The arxml file `Sensor_dataProcessor_em.arxml` avaiable in model folder of Sensor_handler project is configured of PM configuration along with EM(Execution Manager) configuration.

You need to run the pluget `araPmPhmEmManifestGen.pluget` in order to generate Persistency configuration json files into generated folder. As a result of this you will find a generated folder having :

- Sensor_dataProcessor\generated\ara_PM.json will get deploy at PM daemon container at path /etc/adaptive.
- Sensor_dataProcessor\generated\Sensor_dataProcessor.json will get deploy at PM daemon container at path /etc/adaptive/ara_PM.
- Sensor_dataProcessor\generated\Sensor_dataProcessor_Key_Value_Port_2 will get deploy at PM daemon container at path /data/target/adaptive/ara_PM.
- Sensor_dataProcessor\generated\Sensor_dataProcessor\ara_PM.json will get deploy to Sensor_dataProcessor application container at path /etc.

  Initial database configuration file made available at Sensor_dataProcessor\config\initdata-Key_Value_Port_2.json and it will get deploy into Sensor_dataProcessor application container at path /etc/adaptive/ara_PM.

Source and destination path to deploy into Sensor_dataProcessor application container can be defined in file `rootfs.xml`.

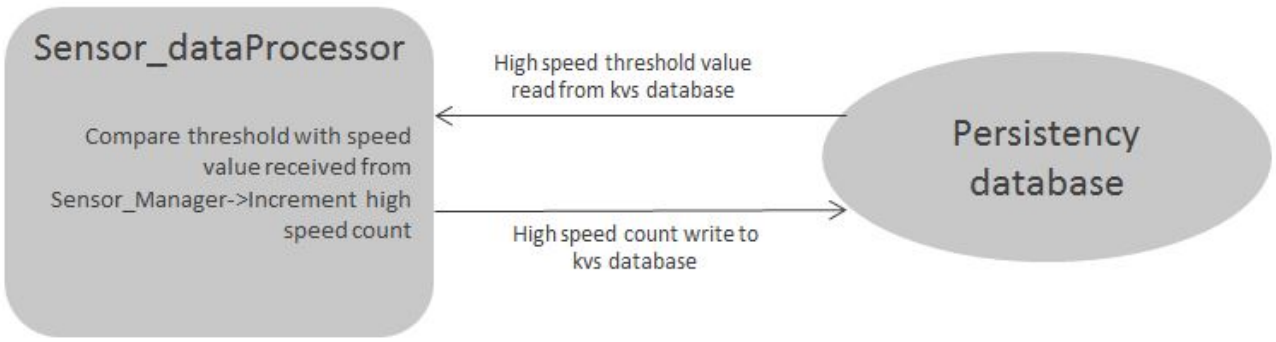


Figure 7.5. ara::pm usecase in Sensor_dataProcessor

In Sensor_dataProcessor, first it reads vehicle speed threshold value from persistency database and compared it with present speed value received from Sensor_Manager. if present speed value exceeds the threshold value then update the high speed count into persistency database and send alert log message.

7.2.5. ara::dlt use case

The Log and Trace functional cluster provides interfaces in the namespace ara::log for applications to forward logging information onto the communication bus, the console, or to the file system. Each of the provided logging information has its own severity level. For each severity level, a separate method is provided to be used by applications or other Adaptive Platform functional clusters. In addition, utility methods are provided, e.g. to convert decimal values into the hexadecimal numeral system, or into the binary numeral system.

ara::dlt is used for the tracing the working logs of the running applications on the console and to convert decimal values into the hexadecimal numeral system.

7.2.6. ara::tsync use case

Time Synchronization between different applications and/or ECUs is of paramount importance when correlation of different events across a distributed system is needed, either to be able to track such events in time or to trigger them at an accurate point in time. For this reason, a Time Synchronization API is offered to the Application, so it can retrieve the time information -synchronized with other Entities / ECUs. The Time Synchronization functionality is then offered by means of different " Time Base Resources" (from now on referred to as TBR) which are present in the system via a pre-build configuration.

Three Time Bases are currently available:

- Pure Local Time Base
- Synchronized Slave Time Base

- Offset Master Time Base

User can access the Time Base Resources via the Master/SlaveTimeBaseResourceProxy in application. For an instance of a Master/SlaveTimeBaseResourceProxy you need an ara::time::InstanceIndetifier. To access the several Time Bases you have to pass the input parameters as the following strings in the InstanceIndetifier constructor:

- "pure_local" or "local" or InstanceIdentifier::Any
- "synchronized"
- "offset"

  Sensor_handler is having synchronized master and offset master tsync features implemented in it. offset master will set the offset time. synchronized master will just print the current time of the time base resource.

  Sensor_dataProcessor is having synchronized slave and offset master tsync features implemented in it. offset master will get the offset time which is set by Sensor_handler. synchronized slave will just print the current time of the time base resource.

  Sensor_Manager is having pure local time base tsync feature implemented in it. it will just print the current time and checks the synchronized status. if it is synchronized with master base time it will continue further execution part of the application.

7.2.7. ara::phm configuration and use case

7.2.7.1. Sensor_handler

NOTE

User needs to restart the HM daemon everytime he restarts the Sensor_handler application, except for the first time the application is run after deploying it or user needs to re-deploy the application container itself.

To resatrt HM-daemon container log in qemu0/1 and run: **systemctl restart hm-container.service**

You need to run the pluget `araPmPhmEmManifestGen.pluget` in order to generate PHM configuration json files into generated folder. As a result of this you will find a generated folder having :

- Sensor_handler\generated\PHM_Sensor_handler.json will get deploy at PHM daemon container at path /etc/adaptive/ara_PHM.
- Sensor_handler\manifests\ara_phm.json will get deploy at PHM daemon container at path /etc/adaptive.
- Initial database configuration file made available at Sensor_handler\manifests\sh_phm_client.json and it will get deploy into Sensor_handler application container at path /etc/adaptive.

Source and destination path to deploy into Sensor_handler application container can be defined in file `rootfs.xml`.

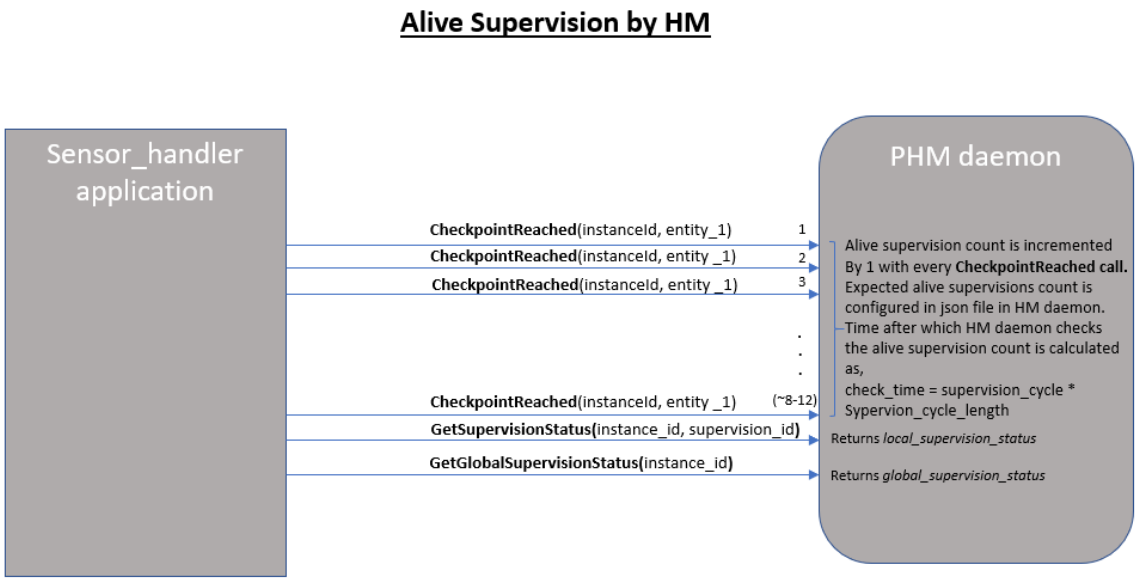


Figure 7.6. ara::phm usecase in Sensor_handler

In Sensor_handler, alive supervision use case of phm is demonstrated by sending CheckpointReached notifications to phm daemon at regular intervals. These notifications are sent every 600msec (configured as interval in `sh_phm_client.json` file) inorder to align with the configurations in `sample.arxml` under model folder. This value is set as 600msec to send the expected amount of alive indications (CheckpointReached) notified to phm daemon.

The format of the configuration file *PHM_Sensor_handler.json* is as follows:

Example 7.1.
PHM: Example of a alive supervision configuration file

```
{
  "Application": {
 "Version_info": {
   "Pluget_build_git_version": "RFI_ADG-1325_1-6-gcd76be7",
   "Pluget_build_date": "21-03-2019 07:36:13"
 },
 "Instance_id": "sensor_handler",
 "Supervision_entities": [
   {
  "Process": "sensor_handler",
  "Supervision_entity_id": 1,
  "Alive_supervisions": [
```

```
          {
          "Checkpoint_id": 1,
          "Expected_alive_indications": 10,
          "Max_margin": 3,
          "Min_margin": 3,
          "Supervision_cycle": 50
          }
        ],
        "Deadline_supervisions": [],
        "Logical_supervisions": [],
        "Failed_alive_supervision_cycle_tolerance": 0
        }
      ],
      "Expired_supervision_cycle_tolerance": 0,
      "Health_channels": [],
      "Health_channel_supervisions": [
        {
        "Supervision_condition": 3,
        "Channel_name": "HealthChannelSupervision",
        "Supervision": 1
        }
      ],
      "Rules": [
        {
        "Name": "Rule",
        "Evaluable_expression": {
          "Name": "LogicalExpression",
          "Sub_expressions": [
          {
            "Condition_type": "isStopped",
            "Instance_id": "sensor_handler",
            "Expression_type": "Mode_condition"
          }
          ],
          "Expression_type": "Logical_expression",
          "Logical_operator": "and"
        },
        "True_action_list": {
          "Execution": "TRIGGERED-ON-CHANGE",
          "Actions": [
          {
            "Callback_function": "safety_app_for_sh",
            "Type": "Callback",
            "Order": 0
          }
          ]
        },
```

```
        "False_action_list": {
          "Execution": "TRIGGERED-ON-EVALUATION",
          "Actions": [
         {
           "Control_type": "RESTART",
           "Type": "Application_control",
           "Order": 1
         }
          ]
        }
         }
       ]
        }
      }
```

7.2.7.2. Sensor_Preprocessor

NOTE

User needs to restart the HM daemon everytime he restarts the Sensor_Preprocessor application, except for the first time the application is run after deploying it or user needs to re-deploy the application container itself.

To resatrt HM-daemon container log in qemu0/1 and run: **systemctl restart hm-container.service**

You need to run the pluget `araPmPhmEmManifestGen.pluget` in order to generate PHM configuration json files into generated folder. As a result of this you will find a generated folder having :

- Sensor_Preprocessor\generated\PHM_Sensor_Preprocessor.json will get deploy at PHM daemon container at path /etc/adaptive/ara_PHM.
- Sensor_Preprocessor\manifests\ara_phm.json will get deploy at PHM daemon container at path /etc/adaptive.
- Initial database configuration file made available at Sensor_Preprocessor\manifests\sp_phm_client.json and it will get deploy into Sensor_Preprocessor application container at path /etc/adaptive.

Source and destination path to deploy into Sensor_Preprocessor application container can be defined in file `rootfs.xml`.

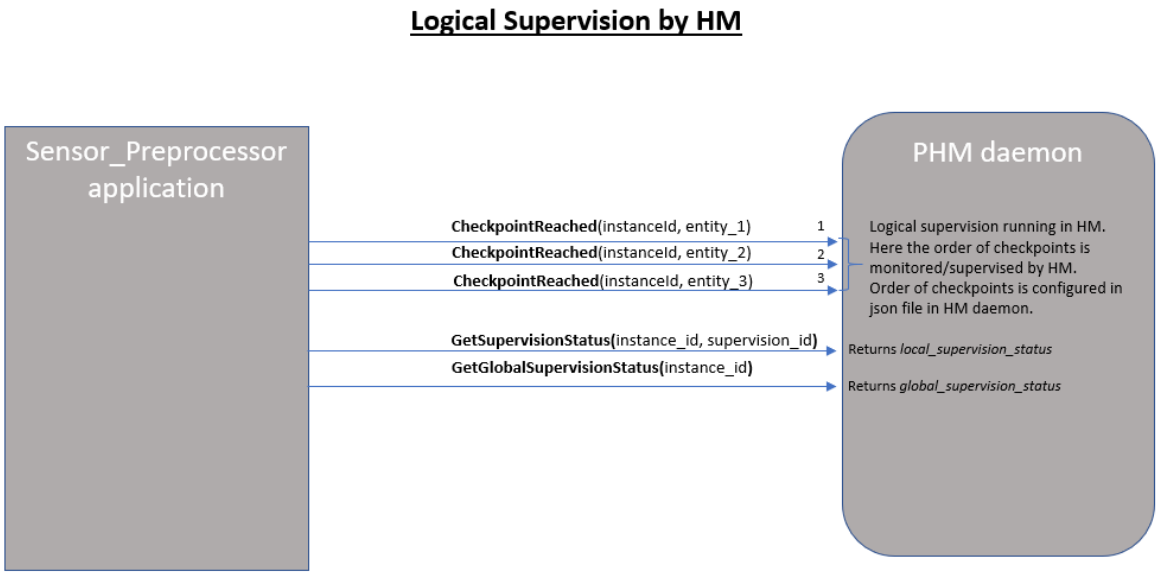


Figure 7.7. ara::phm usecase in Sensor_Preprocessor

In Sensor_Preprocessor, logical supervision use case of phm is demonstrated. Sensor_Preprocessor sends CheckpointReached notifications according to the configuration in `sample.arxml` under model folder.

The format of the configuration file *PHM_Sensor_Preprocessor.json* is as follows:

Example 7.2.
PHM: Example of a logical supervision configuration file

```
{
  "Application": {
 "Version_info": {
   "Pluget_build_git_version": "RFI_ADG-1325_1-6-gcd76be7",
   "Pluget_build_date": "21-03-2019 07:36:13"
 },
 "Instance_id": "sensor_preprocessor",
 "Supervision_entities": [
   {
  "Process": "sensor_preprocessor",
  "Supervision_entity_id": 1,
  "Alive_supervisions": [],
  "Deadline_supervisions": [],
  "Logical_supervisions": [
    {
   "Initial_checkpoint_ids": [
```

```
          0
        ],
        "Final_checkpoint_ids": [
          6
        ],
        "Transitions": [
          {
         "Source_checkpoint_id": 0,
         "Destination_checkpoint_id": 1
          },
          {
         "Source_checkpoint_id": 1,
         "Destination_checkpoint_id": 6
          }
        ]
         }
      ],
      "Failed_alive_supervision_cycle_tolerance": 0
       }
    ],
    "Expired_supervision_cycle_tolerance": 0,
    "Health_channels": [],
    "Health_channel_supervisions": [ ],
    "Rules": [
      {
     "Name": "Rule",
     "Evaluable_expression": {
       "Name": "LogicalExpression",
       "Sub_expressions": [
      {
        "Condition_type": "isStopped",
        "Instance_id": "sensor_preprocessor",
        "Expression_type": "Mode_condition"
      }
       ],
       "Expression_type": "Logical_expression",
       "Logical_operator": "and"
     },
     "True_action_list": {
       "Execution": "TRIGGERED-ON-CHANGE",
       "Actions": [
      {
        "Callback_function": "safety_app_for_sp",
        "Type": "Callback",
        "Order": 0
      }
       ]
```

```
        },
        "False_action_list": {
          "Execution": "TRIGGERED-ON-EVALUATION",
          "Actions": [
         {
           "Control_type": "RESTART",
           "Type": "Application_control",
           "Order": 1
         }
          ]
        }
         }
       ]
        }
      }
```

7.2.7.3. Sensor_Manager

NOTE

User needs to restart the HM daemon everytime he restarts the Sensor_Manager application, except for the first time the application is run after deploying it or user needs to redeploy the application container itself.

To resatrt HM-daemon container log in qemu0/1 and run: **systemctl restart hm-container.service**

You need to run the pluget `araPmPhmEmManifestGen.pluget` in order to generate PHM configuration json files into generated folder. As a result of this you will find a generated folder having :

- Sensor_Manager\generated\PHM_Sensor_Manager.json will get deploy at PHM daemon container at path /etc/adaptive/ara_PHM.
- Sensor_Manager\manifests\ara_phm.json will get deploy at PHM daemon container at path /etc/adaptive.
- Initial database configuration file made available at Sensor_Manager\manifests\sm_phm_client.json and it will get deploy into Sensor_Manager application container at path /etc/adaptive.

Source and destination path to deploy into Sensor_Manager application container can be defined in file `rootfs.xml`.

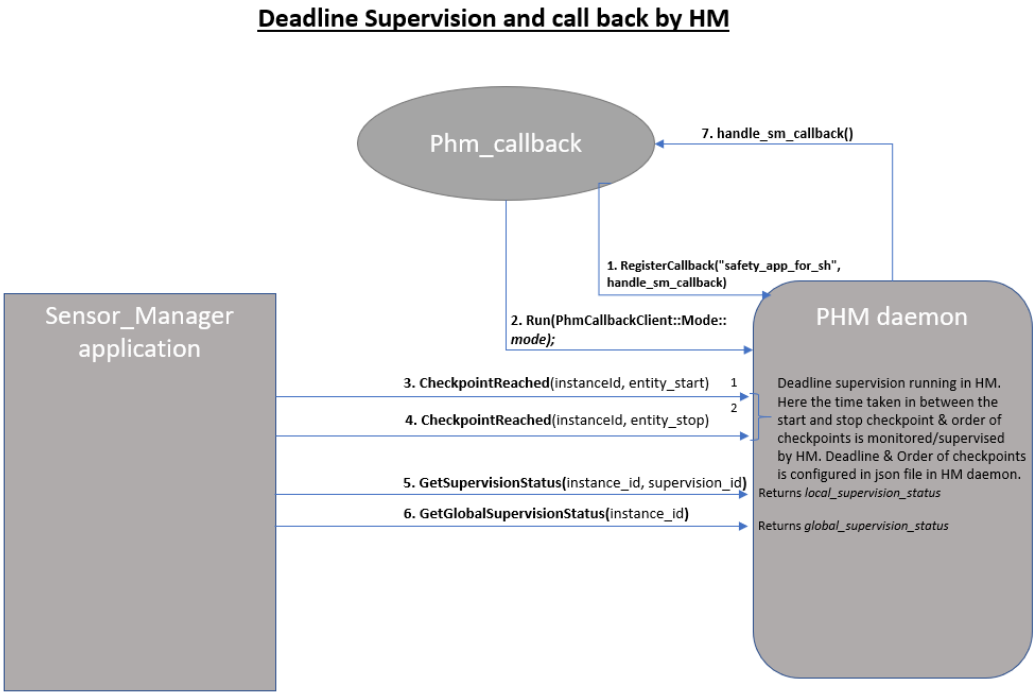


Figure 7.8. ara::phm usecase in Sensor_Manager

In Sensor_Manager, deadline supervision use case of phm is demonstrated. Also demonstrates call_back functionality using phm_callback application Sensor_Manager sends CheckpointReached notifications according to the configuration in `sample.arxml` under model folder.

This configuration file is generated from sample.arxml *PHM_Sensor_Manager.json* is as follows:

Example 7.3.
PHM: Example of a logical supervision configuration file

```
{
  "Application": {
 "Version_info": {
   "Pluget_build_git_version": "RFI_ADG-1325_1-6-gcd76be7",
   "Pluget_build_date": "21-03-2019 07:36:13"
 },
 "Instance_id": "sensor_manager",
 "Supervision_entities": [
   {
  "Process": "sensor_manager",
  "Supervision_entity_id": 1,
  "Alive_supervisions": [],
  "Deadline_supervisions": [
```

```
        {
       "Deadline_max": 600,
       "Deadline_min": 40,
       "Start_checkpoint_id": 1,
       "Stop_checkpoint_id": 2
        }
     ],
     "Logical_supervisions": [],
     "Failed_alive_supervision_cycle_tolerance": 0
      }
    ],
    "Expired_supervision_cycle_tolerance": 0,
    "Health_channels": [],
    "Health_channel_supervisions": [
      {
     "Supervision_condition": 3,
     "Channel_name": "HealthChannelSupervision",
     "Supervision": 1
      }
    ],
    "Rules": [
      {
     "Name": "Rule",
     "Evaluable_expression": {
       "Name": "LogicalExpression",
       "Sub_expressions": [
      {
        "Condition_type": "isStopped",
        "Instance_id": "sensor_manager",
        "Expression_type": "Mode_condition"
      }
       ],
       "Expression_type": "Logical_expression",
       "Logical_operator": "and"
     },
     "True_action_list": {
       "Execution": "TRIGGERED-ON-CHANGE",
       "Actions": [
      {
        "Callback_function": "safety_app_for_sm",
        "Type": "Callback",
        "Order": 0
      }
       ]
     },
     "False_action_list": {
       "Execution": "TRIGGERED-ON-EVALUATION",
```

```
          "Actions": [
         {
           "Control_type": "RESTART",
           "Type": "Application_control",
           "Order": 1
         }
          ]
        }
         }
      ]
       }
     }
```

7.2.8. ara::rest configuration and use case

ara::rest can establish communication paths between Adaptive applications. client can send and receive data from server using URI and Object graph model. The Object Graph Model is a protocol-binding independent tree-like data structure which is the cornerstone of all ara::rest communication. Its purpose is to map to a protocol format such as JSON as well as to C structs. REST libraries parse the configuration file `rest.json` by default at the application root folder. The application may introduce another configuration file location in the restService and Display_Manager constructors. Methods used in restful communication is kGet and kPost in current Traffic Sign Scenario application. configuration id configuration file must be set in rest client and rest server applications.

The format of the configuration file *rest.json* is as follows:

Example 7.4.
REST: Example of a configuration file

```
[
  {
    "id" : "conf1",
    "bindings" : [
      {
        "host" : "127.0.0.1"
        "port" : 9095,
        "use_dynamic_port" : false,
        "dynamic_port_min" : 9000,
        "dynamic_port_max" : 9095
      },
      {
        "host" : "::1",
        "port" : 9096
```

```
        },
        {
          "host" : "127.0.0.1",
          "use_dynamic_port" : true,
          "dynamic_port_min" : 9097,
          "dynamic_port_max" : 9099
        }
      ],
      "compression" : false,
      "compression_min_length" : 890,
      "caching" : false,
      "default_headers" : {
        "X-Powered-By" : "Ara Rest 1.1.0"
      }
    },
          {
            "id" : "conf2",
            "bindings" : [
            {
              "host": "127.0.0.1",
              "port" : 9095
            },
            {
              "id" : "ipv6",
              "host" : "::1",
              "port" : 9096
           }
          ]
         }
        ]
```

Field	Description	Mandatory or not
id	Configuration ID. Is used in Client/Server constructors.	
bindings	List of network bindings	At least one item is mandatory.
host	IP address to listen or connect. Supports both IPv4 and IPv6 addresses.	Mandatory
port	Port to listen to or connect to.	Mandatory, if the dynamic port is not used.
use_dynamic_port	Configures whether dynamic port range is used (finding an available port between ranges). Defaults false. Only available for server.	

Field	Description	Mandatory or not
dynamic_port_max	Dynamic port range end. Defaults to 65535.	
compression	Configures whether HTTP compression is used or not. Defaults to true.	
compression_min_length	Minimum number of bytes that will be compressed. Defaults to 1024.	
caching	Configures whether caching via HTTP ETags is used. Defaults to true.	
default_headers	List of default headers that will be added to requests/responses.	

Table 7.1. Description of JSON fields

- restService is the rest server application, it will maintain the car information using object graph model data structure. it will send the pointer of Object graph model to the Display_Manager(rest client). Display_Manager will set the speed tag value and send it to the restService. restService will read the speed tag value and based on some logic, Modify the isOverspeed tag from car information and give a standard output Over Speed Challan Generated.
- Display_Manager will parse the isOverspeed tag from server using pointer of Object graph model. will give a standard output Over Speed Challan Received if the isOverspeed tag value is true. After that it will set the recent speed tag value and send it to the restService.

7.2.9. Running the Traffic Sign Scenario applications

restService→phm_callback→Sensor_Manager → Sensor_handler → Sensor_Preprocessor → Sensor_dataProcessor → Display_Manager

NOTE

User needs to run Sensor_Manager application and wait for the following log to appear in Sensor_Manager application and then proceed with running other Traffic Sign Scenario apps (Sensor_handler, Sensor_Preprocessor etc..)

"Sensor Manager: Manager initialized. Proceed with running other Apps ... "

Make sure that restService is deployed before Display_Manager, because Display_Manager is dependent on the restService application container IPv6 address.

Figure 7.9. Run Traffic Sign Scenario app sequence

7.3. demo Application

The demo application describes usage of gtest , gmock , doxygen, code coverage and ReqM2 within the EB corbos Starter Kit. This application is having four test cases, in which two test cases will pass and two will fail intentionally.

NOTE

If user wants to use gtest and gmock functionality in any other project then refer section gtest_and_gmock_usage

7.4. Motor Speed Scenario

Motor Speed scenario described using the below applications and it explains the functionality of diagnostic management.

- Motor Speed
- Motor Speed Monitor
- Lamp
- Operation State

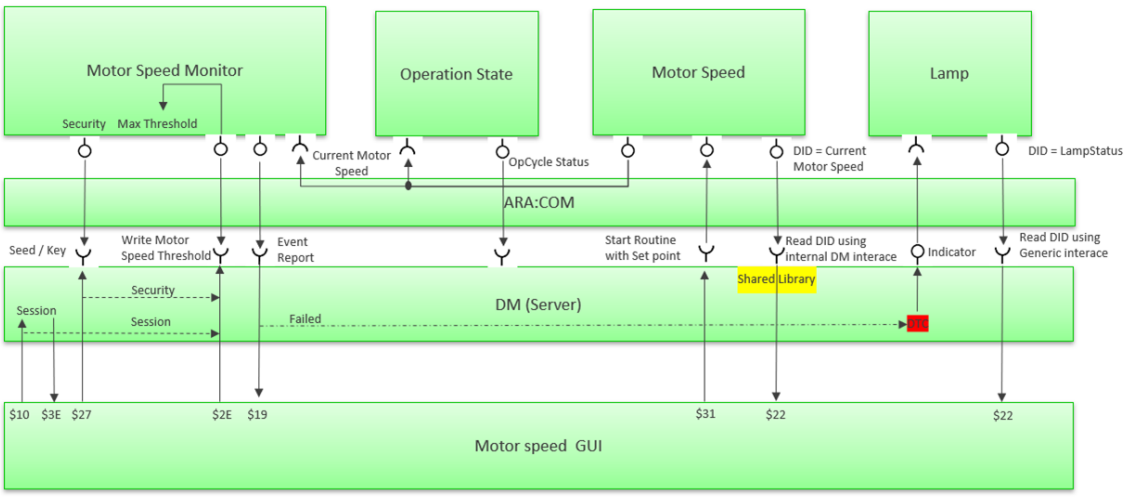


Figure 7.10. Motor speed scenario - demo applications

- Motor Speed : It provides the motor speed and default startup is 0 rpm. The Motor speed can be varied using a Routine start test where you can provide a setpoint value. The Motor speed is incremented in

steps of 200 rpm per second until the setpoint is reached. Hold 10 seconds before ramp down. Routine Id is 0x200. The Motor speed is set back to default after hold time. The speed is reduced in steps of 200rpm until it reaches back 0 rpm. Current Motor speed can be read with DID 0x0100 (2 bytes)

- Motor Speed Monitor : It monitors the current speed against a threshold. The Max threshold at the start is set to 2500 rpm. And the max threshold can be written using DID 0x201 in extended session with security level = 2. If the current speed is less than or equal to the threshold then Ok is reported else Failed is reported. DTC stored is 0x100000. Motor speed resolution 1rpm = 1 bit.
- Lamp : This controls the MIL lamp. If the DTC is confirmed then MIL lamp is On else Off . Status of the MIL Lamp can be read as DID 0x0200 (1 Byte -> 0 = Off, 1 = On).
- Operation State : Thsi controls the operation cycle of the system. If Motor speed > 0 rpm then Operation Cycle = True else Operation Cycle = False.

7.4.1. Running the motor speed applications sequence

User needs to run the dm applications in the below order.

- dm_motor_speed
- dm_motor_speed_monitor
- dm_motor_operation
- dm_lamp

7.4.1.1. Motor speed scenario

- User who are testing the applications on the Qemu please use corbos\starterkit_vx.x\utility\moror_speed_-scenario\motor_speed_scenario_qemu.py to test

7.4.1.2. Using the CLI to run the application

- Start the applications mentioned in the above order
- Use the sctipt present in the <Installdir>\starterkit_vx.x\utility\moror_speed_scenario\motor_speed_scenario_qemu.py for demonstration of the scenario
- To start the demo you can use the python if it is installed in your PC or you can use the one present in the <Installdir>\common_tools\Python27\python.exe
- To get the command line help options please use <Installdir>\common_tools\Python27\python.exe <Installdir>\starterkit_vx.x\utility\moror_speed_scenario\motor_speed_scenario_qemu.py --command help

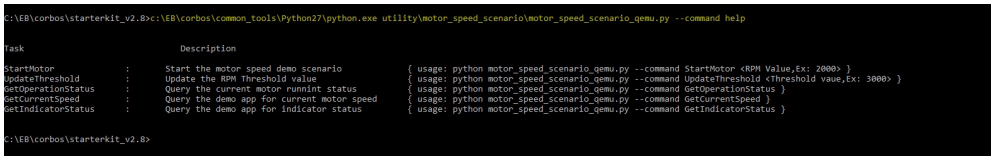


Figure 7.11. motor_speed_scenario_help_window

- To start the motor please run <Installdir>\common_tools\Python27\python.exe <Installdir>\starterkit_vx.x\utility\moror_speed_scenario\motor_speed_scenario_qemu.py --command StartMotor (Set point value ex : 2000)

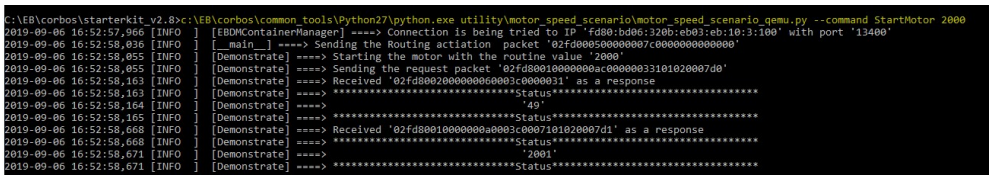


Figure 7.12. motor_speed_scenario_StartMotor_window

- To set the thresold please run <Installdir>\common_tools\Python27\python.exe <Installdir>\starterkit_vx.x\utility\moror_speed_scenario\motor_speed_scenario_qemu.py --command UpdateThreshold (Threshold value ex : 3500)

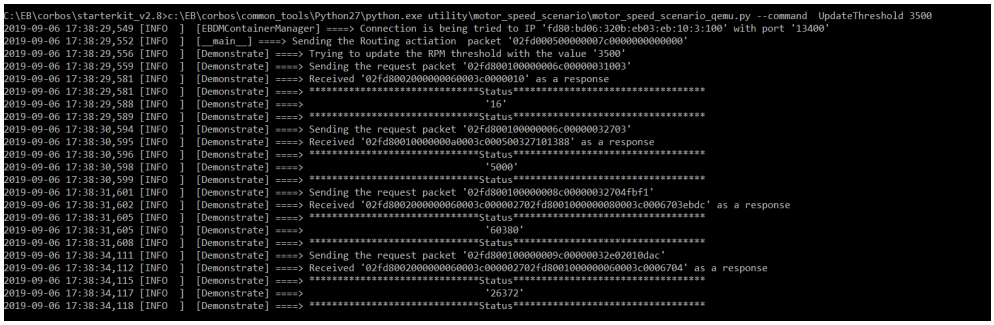


Figure 7.13. motor_speed_scenario_UpdateThresold_window

- To get indicator status please run <Installdir>\common_tools\Python27\python.exe <Installdir>\starterkit_vx.x\utility\moror_speed_scenario\motor_speed_scenario_qemu.py --command GetIndicatorStatus

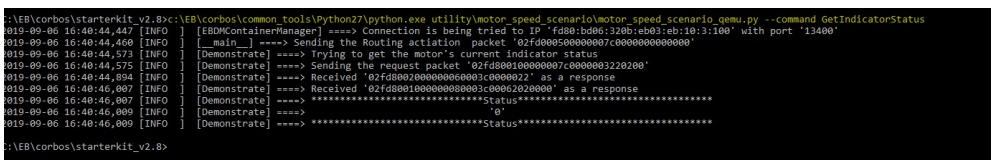


Figure 7.14. motor_speed_scenario_indicatorstatus_window

- To get curent speed please run <Installdir>\common_tools\Python27\python.exe <Installdir>\starterkit_vx.x\utility\moror_speed_scenario\motor_speed_scenario_qemu.py --command GetCurrentSpeed

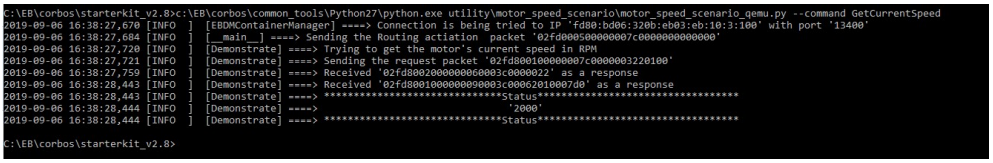


Figure 7.15. motor_speed_scenario_currentspeed_window

- To get operational status please run <Installdir>\common_tools\Python27\python.exe <Installdir>\starterk-it_vx.x\utility\moror_speed_scenario\motor_speed_scenario_qemu.py --command GetOperationalStaus

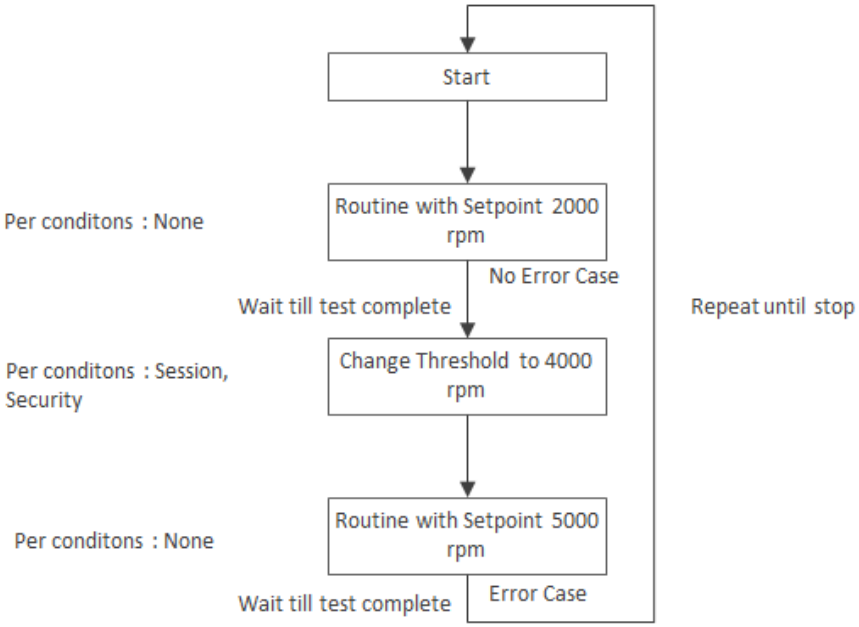


Figure 7.16. Dianostic Management usecase (sequence flow)

8. Wireshark: SOME/IP tracing

8.1. Installing Wireshark and SOME/IP dissectors

Installing Wireshark and SOME/IP dissectors

Step 1
Locate the directory were you install the EB corbos Starter Kit.

Step 2
Run **wireshark.cmd**.

When asked whether you want to install Wireshark enter `Y`.

The Wireshark installation wizard opens up.

Step 3
Follow the installation instructions in the wizard.

In the **Choose Components** dialog, in the options tree, go to **Tools** and enable the checkbox for **SSHdump**.

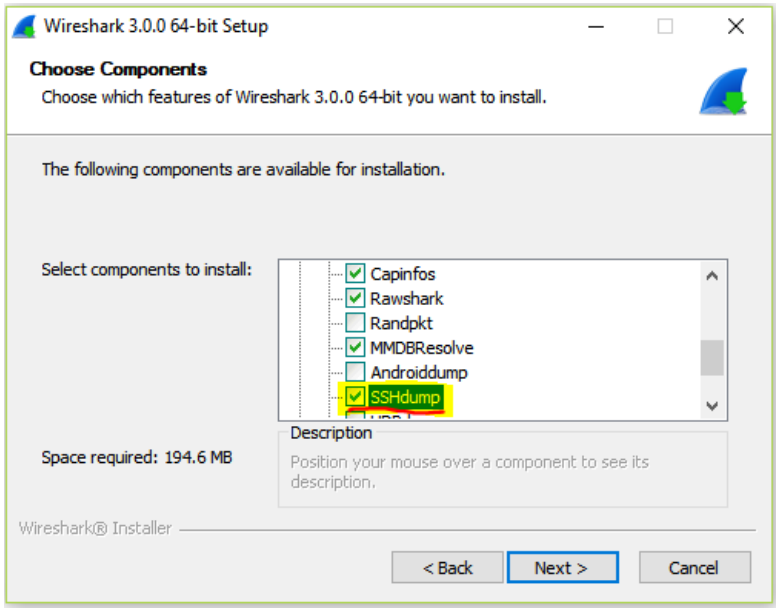


Figure 8.1. Choose SSHdump

Step 4
Click **Next** and continue the installation.

This will also install SOME/IP dissectors plugins at path `C:\Users\%USERPROFILE%\AppData\Roaming\Wireshark\plugins`.

8.2. Analyzing SOME/IP traffic with Wireshark

Analyzing SOME/IP traffic with Wireshark

Step 1
Log in via Putty SSH into the qemu base container (IP: 10.10.10.2)

- Log in as: root
- Password: 1

Step 2
Use the tcpdump and start network capture using the following command:

tcpdump -i br0 port not 22 -w trace_br0.dump

Step 3
Run ara_com applications in EB corbos Studio.

Step 4
Stop *tcpdump* capture with **Crtl+C** via PuTTY.

Step 5
Copy the capture file via pscp (Putty command line tool) back to Windows.

"C:\Program Files\PuTTY\pscp.exe" root@10.10.10.2:/root/trace_br0.dump <destination_path>

Step 6
Analyse *trace_br0.dump* with Wireshark on Windows (EB internal Wireshark SOME/IP Testability Dissectors).

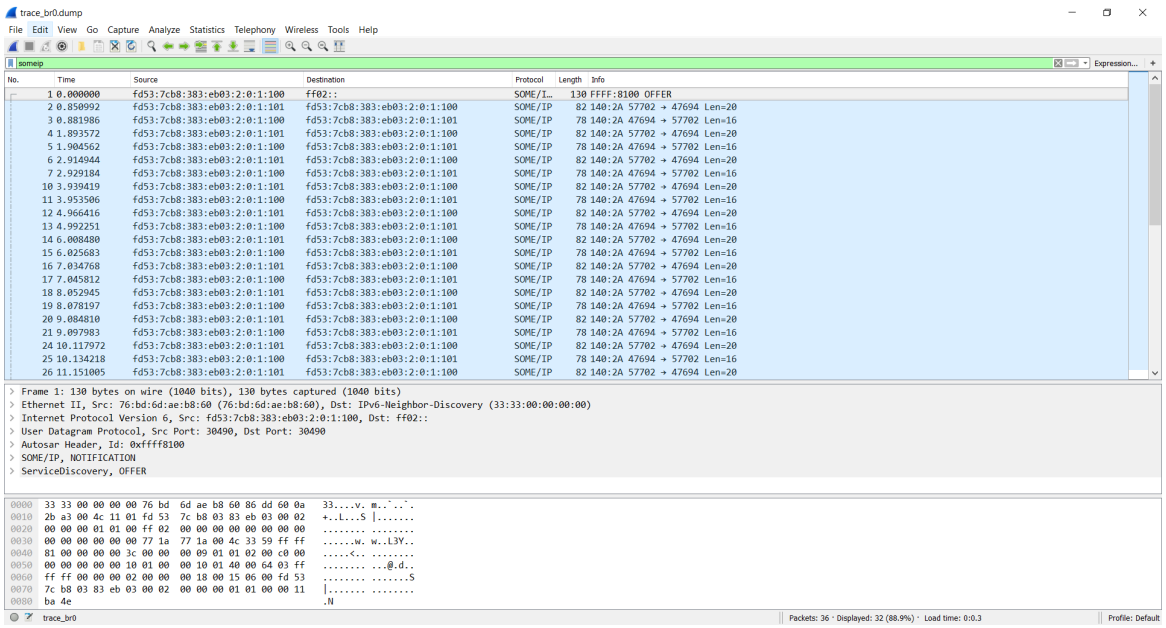


Figure 8.2. tcpdump file Capture

8.2.1. Remote live capture

There are two variants of remote live capture:

- Section 8.2.1.1, “Remote live capture with sshdump and GUI”
- Section 8.2.1.2, “Remote live capture with Wireshark and PuTTY”

8.2.1.1. Remote live capture with sshdump and GUI

Remote live capture with sshdump and GUI

Step 1
Run Wireshark and start the SSH remote capture:

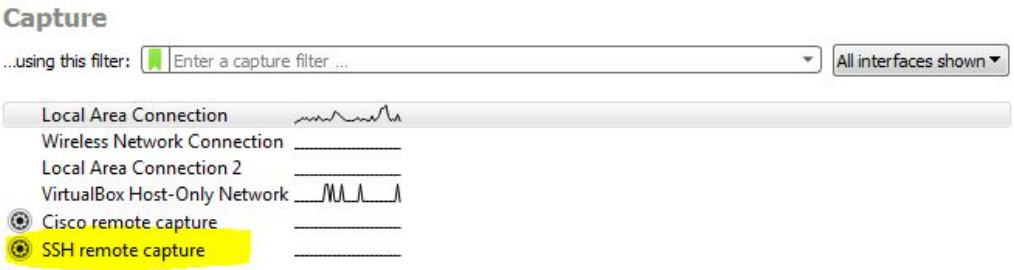


Figure 8.3. Start ssh remote capture

Step 2
Configure the SSH remote capture:

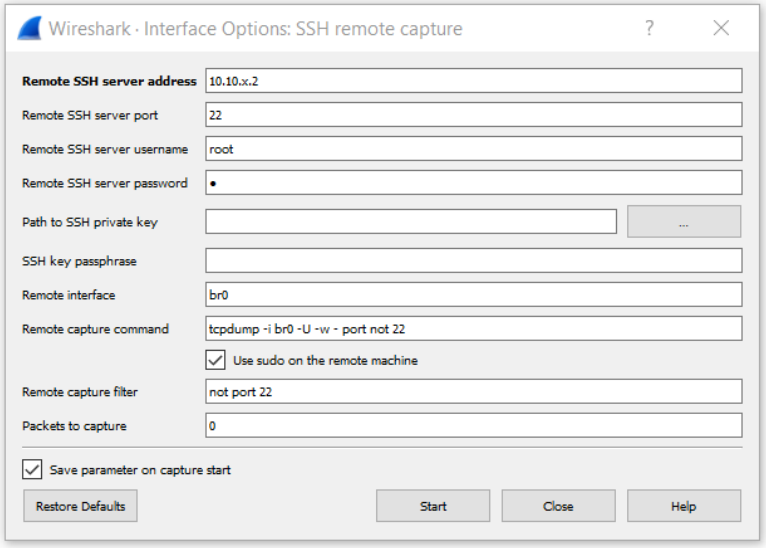


Figure 8.4. Configure ssh remote capture

Step 3
Click **Start**.

Step 4
Run ara_com applications in EB corbos Studio.

8.2.1.2. Remote live capture with Wireshark and PuTTY

Remote live capture with Wireshark and PuTTY

Step 1
Execute the following command using *cmd*

"C:\Program Files\PuTTY\plink.exe" -ssh -pw 1 root@10.10.10.2 "tcpdump -ni br0 -l -s 0 -w - not port 22" | "C:\Program Files\Wireshark\Wireshark.exe" -i -

The Wireshark window opens up. Press the capture button.

Step 2
Run ara_com applications in EB corbos Studio.

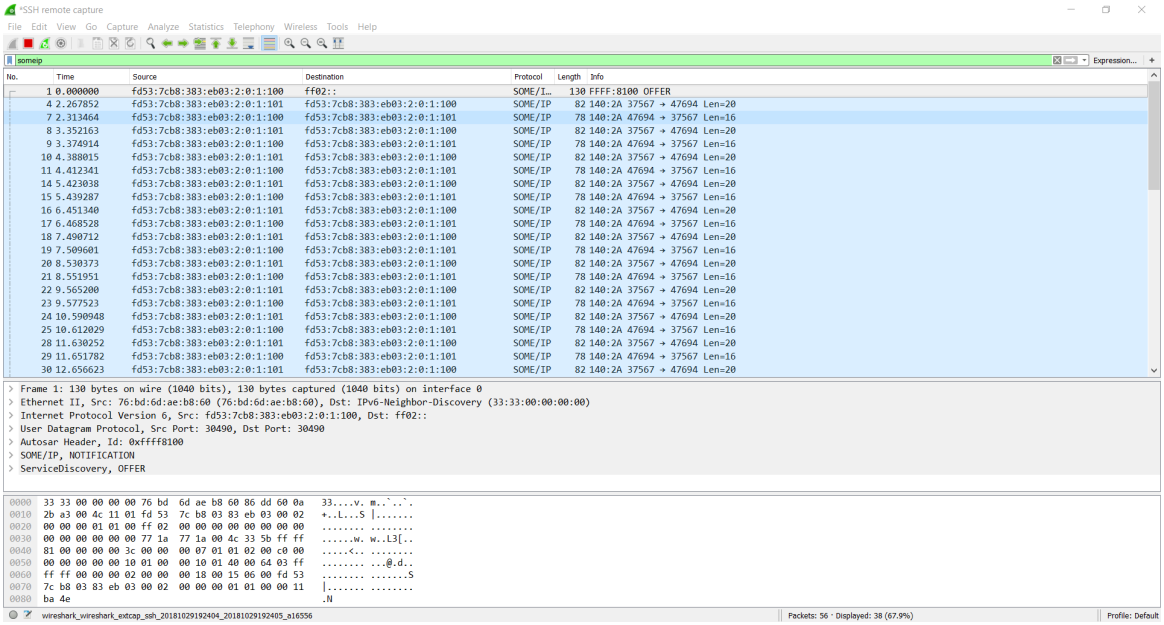
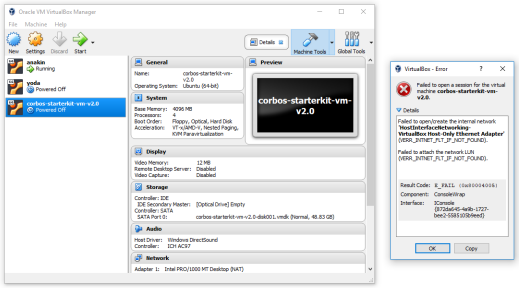


Figure 8.5. Remote live capture

9. Troubleshooting

Summary	**EB Corbos Starter kit installation failure due to mulitple installation of virtual box hos-tonly network adapter**
Description	There is a possible case of installation failure due to multiple VirtualBox Host-Only Network adapter has been installed in your pc. To confirm this possible scenario, please start the cor-bos-starterkit-vm manually. If the below error is observed, then follow the workaround steps.  Figure 9.1. Starterkit vm error
Workaround	1. Poweroff Corbos starter kit VM. 2. Remove the VM from Virtual Box by right clicking on the specific VM. 3. Remove current installed version of starterkit_vx.x folder form installed location (default location will be C:\EB\corbos\starterkit_vx.x) 4. Unistall the Virtual Box from **ControlPanel -> Program and Features** 5. Install the Virtual Box again (suggested version for current release) 6. Initaite the starter kit installation again Note: On re-installing Virtual Box ,existing VMs will be retained and appears back in the registerd VM list.

Summary	**QEMU runtime image(s) not accessible**
Description	It is possible that one or both targets are not accessible via SSH. Possibly an unclean shut-down caused a corrupted QEMU runtime image.
Workaround	1. Stop and restart the qemu docker container as mentioned in the section Section 6.3, “Restart QEMU base docker container using PuTTY”

Summary	**Host key error while SSH connection to QEMU**
Description	Runtime images, i.e. QEMU, are not able to connect via SSH and there is a host key error.
Workaround	1. Connet the SDK container via putty. Refer Table 5.1, “Connection settings” for connec-tion settings

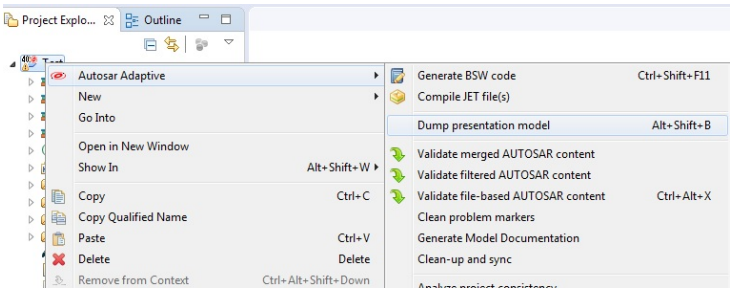
Summary	**Demo projects are not imported into EB corbos Studio**
Workaround	Import the demo projects manually from EB corbos Studio: 1. In the **File** menu, select **Import... → General → Existing Projects into Workspace**. 2. Select the workspace directory workspace_corbos default located at `C:/EB/corbos/starterkit_vx.x/workspace_corbos`. 3. Select the projects you want to import.

Summary	**Demo projects are not imported into workspace `C:/EB/corbos/starterkit_vx.x/workspace_corbos`**
Workaround	Import the demo projects manually from EB corbos Studio: 1. Manually delete the folder workspace_corbos at `C:/EB/corbos/starterkit_vx.x/` 2. Execute the python task **tools\Python27\python.exe scripts\python\main.py --command copyDemos** and **tools\Python27\python.exe scripts\python\main.py --command installTemplates** using cmd from the EB corbos Starter Kit installation directory. 3. Check the workspace again.

Summary	**GDB: Cannot bind address: Address already in use.**
Description	Cause: The gdbserver process is still running on the deployed application container.
Workaround	1. Log in to the application container using PuTTY 2. Find out the process ID (PID) via ps. 3. Kill processed: kill $affectedPID

Summary	**Installation problems**
Description	If you have installation problems and you want to manually delete the VM artifacts.
Workaround	1. Shut down the VM in the VirtualBox Manager. 2. Delete *corbos-starterkit-vm* via **CTRL+R** or in the context menu. 3. Press **CTRL+D**. If you still have any *corbos-starterkit-vm-** images, delete them. 4. If existent, delete the directory `C:\Users\%USERPROFILE%\VirtualBox VMs\corbos-starterkit-vm` 5. Install EB corbos Starter Kit again.

Summary	**Cannot create AUTOSAR configuration file**
Description	When you create a file, a StackOverFlow error is thrown, unless you already have a file inside the project.
Workaround	1. Create a project.

Summary	Cannot create AUTOSAR configuration file
	2. In EB corbos Studio go to the **Project Explorer**. 3. Right-click the project. In the context menu select **AUTOSAR Adaptive → Dump presentation model**.  Figure 9.2. Menu **Dump presentation model**

Summary	Program 'gcc/g++' not found in the path
Description	In EB corbos Studio, the following error messages occur in the **Problems** tab: Program "g++" not found in PATH Program "gcc" not found in PATH This error message appears because EB corbos Studio does not use a native cross-compiler for Windows but a cross-compiler for Linux. If you have already installed mingw/cygwin compiler in your host system, the error does not appear. EB corbos Studio looks for the default path where the compiler is installed. The error is issued if the compiler is not found in the default path. This is a known issue. This issue will be solved by implementing a native cross-compiler for Windows in EB corbos Studio.
Workaround	-

Summary	VBoxManage.exe: error

Description	`VBoxManage.exe: error: VT-x is disabled in the BIOS for all CPU modes (VERR_VMX_MSR_ALL_VMX_DISABLED)`
	`VBoxManage.exe: error: Details: code E_FAIL (0x80004005), component ConsoleWrap, interface IConsole`
	Cause: Virtualization is not enabled on your computer.

Workaround	Enable virtualization in your BIOS. For more details, see the user's manual of your computer.

Summary	**Installation problems**
Description	If you have installation problems such as *importvm FAILED* or *clearvm FAILED* and you want to manually delete the VM artifacts.
Workaround	1. Shut down the VM in the VirtualBox Manager. 2. Delete *corbos-starterkit-vm* via **CTRL**+**R** or in the context menu. 3. Press **CTRL**+**D**. If you still have any *corbos-starterkit-vm-** images, delete them. 4. If the directory exists, delete directory `C:\Users\%USERPROFILE%\VirtualBox VMs\corbos-starterkit-vm` 5. Install EB corbos Starter Kit again. In case the above steps have not helped with fixing the issue, contact the EB support team and provide the log file `importvm-log.log, inatalltemps_install-log.log` found in the EB corbos Starter Kit installation path. If you did not change the directory during installation the default directory is `C:\EB\corbos\starterkit_vx.x\installLog`.

Summary	**Installation problems**
Description	`Error starting VM or VM not reachable. Exit code: 0` or *startvm FAILED*
Cause	VM is not able to connect to the host machine.
Workaround	Set IPv4 address of VirtualBox Host-Only Network to 10.10.11.2 and IPv4 Network Mask to 255.255.0.0 1. Click **Start** → **Oracle VM VirtualBox**. 2. Click **File** → **Preferences** → **Network** → **Host-only Networks**. 3. Edit VirtualBox Host-only Ethernet Adapter #<Number>. To know exact Adapter #<Number> ; refer `C:\EB\corbos\starterkit_vx.x\log\vm_adapter_number.log`(Default Installation Path). 4. Set *IPv4 Address* to 10.10.11.2 and *IPv4 Network Mask* to 255.255.0.0. 5. Click **OK**. In case the above steps have not helped with fixing the issue, contact the EB support team and provide the log file `importvm-log.log` found in the EB corbos Starter Kit installation path. If you did not change the directory during installation the default directory is `C:\EB\corbos\starterkit_vx.x\installLog`.

Summary	**Remove red markers from code**
Description	EB corbos Studio shows red markers in a code
Workaround	Red markers from the code can be removed by following these steps:

Summary	**Remove red markers from code**
	1. Execute the AraComBindingGenerator.pluget because in the code some include path and API depends on the generated folder. 2. Right click on project → Index → Rebuild.

Summary	**If qemu0, qemu1 and corbos-starterkit-vm are not pingable**
Description	The installation fails and gives the error that qemu0, qemu1 and corbos-starterkit-vm are not pingable.
Workaround	Check the windows firewall option. If it is enabled, turn it off.

Summary	**If SOME/IP communication is not working**
Description	If someip communication is not working, because com-daemon crashed.
Workaround	Restart the com-daemon container using this command from the qemu PuTTY terminal **/sbin/cont-ctrl start com-daemon-container**

Summary	**Error VT-x is not available (VERR_VMX_NO_VMX)**
Description	If Hyper-V is enable in users system then disable that before installing the EB corbos Starter Kit
Workaround	Open Command Prompt as administrator and run **dism.exe /Online /Disable-Feature:Microsoft-Hyper-V**

Summary	**Overcoming Deletetion of virtual box folder after System restart**
Description	Sometimes Organizational specific IT Policy deletes ".VirtualBox" directory located in `C:\Users\%USERPROFILE%\` after each system restart
Workaround	Before each restart, and after restart of PC, this folder should be restored back in the `C:\Users\%USERPROFILE%\` before using the EB corbos Starter Kit

Summary	**Deploy and Running Application takes Too Long time(More than 10 Minutes)**
Description	Deploying and Running Application from corbos studio takes too long time due to Laptop Power Option is set to “Power Saver Mode”.
Workaround	User Need to change Laptop Power option to “High Performance” mode as shown below.

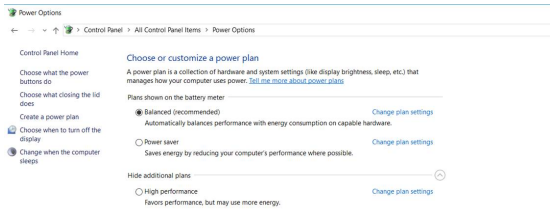
Summary	Deploy and Running Application takes Too Long time(More than 10 Minutes)
	

Figure 9.3. Power seetings options

Summary	Debugging appliations encountered problem with connection timeout
Description	There is a possible for the user to face the below error when user debugging an applications.

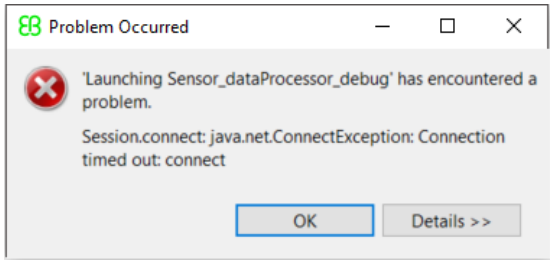


Figure 9.4. Starterkit vm error

Workaround	1. Go to debug configurations and edit connection

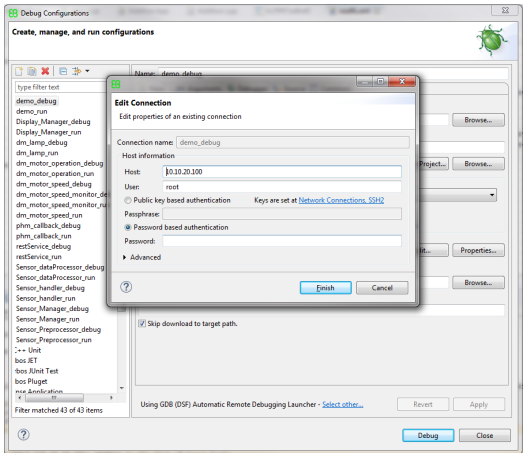


Figure 9.5. Starterkit vm error

2. **Click Network Connections SSH2 link** and then under **known Hosts** tab clear all the existing registered hostname IPs

Summary	Debugging appliations encountered problem with connection timeout

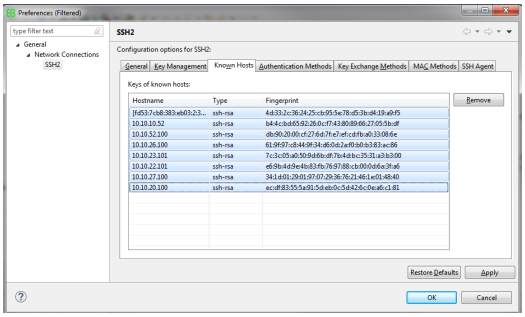


Figure 9.6. Starterkit vm error

3. Click **Apply -> OK -> Finish**. And then start debugging applications.

Glossary

A

AA	Adaptive Application
ARA	AUTOSAR runtime for Adaptive Applications
AP	AUTOSAR Adaptive Platform
API	Application programming interface
AUTOSAR	AUTomotive Open System Architecture

B

Build target	An Eclipse feature

D

datadisk	Name of the mounted SMB drive from the development-image to provide a common drive accessible from Windows and Linux
development image	A basic Ubuntu is used to compile adaptive applications for runtime images

E

EM	Execution Manager

G

GDB	GNU Project Debugger
Python	Scripts in the background are implemented using Python. For details, see https://www.python.org/

M

Manifest	A manifest represents a piece of AUTOSAR model description that is created to support the configuration of an AUTOSAR Adaptive Platform product and which is uploaded to the AUTOSAR Adaptive Platform product, potentially in

	combination with other artifacts (like binary files) that contain executable code to which the manifest applies.

Q

QEMU	QEMU is a generic and open source machine emulator and virtualizer. For details, see http://www.qemu.org/

R

runtime image	Target QEMU image which contains ARA modules

S

SCP	Secure copy protocol
SDK	Software development kit
SMB	Server message block protocol
SSH	Secure shell

T

task	Python task that performs an action.

V

VM	Virtual machine

Appendix A. Open source license

NOTE

License Information

EB corbos Starter Kit includes software with several types of licenses. For details, see the Product licensing user's guide.