OS PAL USER GUIDE

MapuSoft Technologies
1301 Azalea Road
Mobile, AL 36693

Table of Contents

INTRODUCTION TO OS PAL

About OS PAL

OS PAL integrates OS Changer and OS Abstractor with Eclipse's CDT to offer an IDE for developing and porting applications.

OS PAL features include:

- Creation of C and C++ OS PAL projects
- Automatic configuration of any OS Changer and OS Abstractor APIs needed by the application
- Custom configuration of OS resources needed by the application
- Custom configuration of OS Abstractor Resources
- Custom configuration of OS Abstractor for single or multi-application development (Process Feature support)

OS PAL uses OS Abstractor and OS Changer technology to produce optimized target code:

- Up to 9 target configuration tabs to optimize the target code specific for your application
- Generated target code is optimized to contain only the APIs used by the application
- Allows for further optimization by in-lining user selected API's

Contact MapuSoft to receive the components needed for using OS PAL. The steps for using OS PAL are comprehensively described in the following pages. The OS PAL User Guide contains screen shots for many of the steps. Where appropriate, links to the steps in the User Guide are included for more information as well as key Talking Points explaining the features.

HOST Platforms Supported by OS PAL

OS PAL can run primarily on two following platforms:

1. **Windows**: OS PAL for Windows works with any versions of Windows XP.
2. **Linux**: OS PAL for Linux works with Red Hat Linux distributions that use 2.6 and above kernel versions.

Installing OS PAL

You can download an evaluation copy from our website or install ospal via the evaluation cd given by MapuSoft Technologies.

To install OS PAL**:**

1. Insert the CD and run it. A welcome html page will be auto run.
2. Click **Port, abstract and optimize code on a host OS PAL** link.
3. Select the host. For Example: windows or Linux.
4. Click **Click here to begin installation**. This will launch the ospal installer which is called *os-porting-and-abstraction-lab-windows-jvm-installer.exe*.

   NOTE: This *.exe* file is what you have, if you download ospal from the web manually.
5. Double click on the **.exe** file to launch the installer.
6. You can view the installation system requirements displayed the installer wizard.
7. Browse and select the installation directory.

   NOTE: By default, it is c:\MapuSoft\OSPAL.

List of Technical Documentation

Reference manuals can be provided under NDA. Click http:mapusoft.com/contact/ MapuSoft to request for a reference manual.

The document description table lists MapuSoft Technologies manuals.
Using these documents you can:

- Port applications
- Perform OS Abstraction
- Do code optimization
- Generate Standalone OS Abstractor/OS Changer Packages
- Enable profiling

Document Description Table

Document	Description
Programmers Guide	Provides detailed description of how to get started with MapuSoft Abstraction frame work and porting applications. This guide: ▪ Explains how to get started with MapuSoft Technologies products
OS Abstractor Reference Manual	Provides detailed description of how to do abstraction solution. This guide: ▪ Explains how to develop code independent of the underlying OS ▪ Explains how to make your software easily support multiple OS platforms
OS Changer Reference Manual	Provides detailed description of how to get started with OS Changer. This guide: ▪ Explains how to port applications to target platforms
OS PAL User Guide	Provides detailed description of how to use OS PAL. This guide: ▪ Explains how to port applications ▪ Explains how to import legacy applications ▪ Explains how to do code optimization ▪ Explains how to generate standalone OS Abstractor/OS Changer packages
Release Notes	Provides the updated release information about MapuSoft Technologies new products and features for the latest release. This document: ▪ Gives detailed information of the new products ▪ Gives detailed information of the new features added into this release and their limitations, if required

Procuring a License for OS PAL

OS PAL is licensed by the following host and target licenses. A 30 day advanced evaluation license is available for the host licenses.
Click http://mapusoft.com/downloads/ospal-evaluation/ to request an evaluation license.

Supported Host Platforms:

- Windows XP
- Linux
- Solaris*
  *Available soon

Supported Development APIs:

- Base OS Abstractor
- POSIX OS Abstractor
- VxWorks OS Changer
- pSOS OS Changer
- Nucleus OS Changer
- micro-ITRON OS Abstractor

For a list of OS PAL supported target operating systems, click here:
http://mapusoft.com/products/offerings/

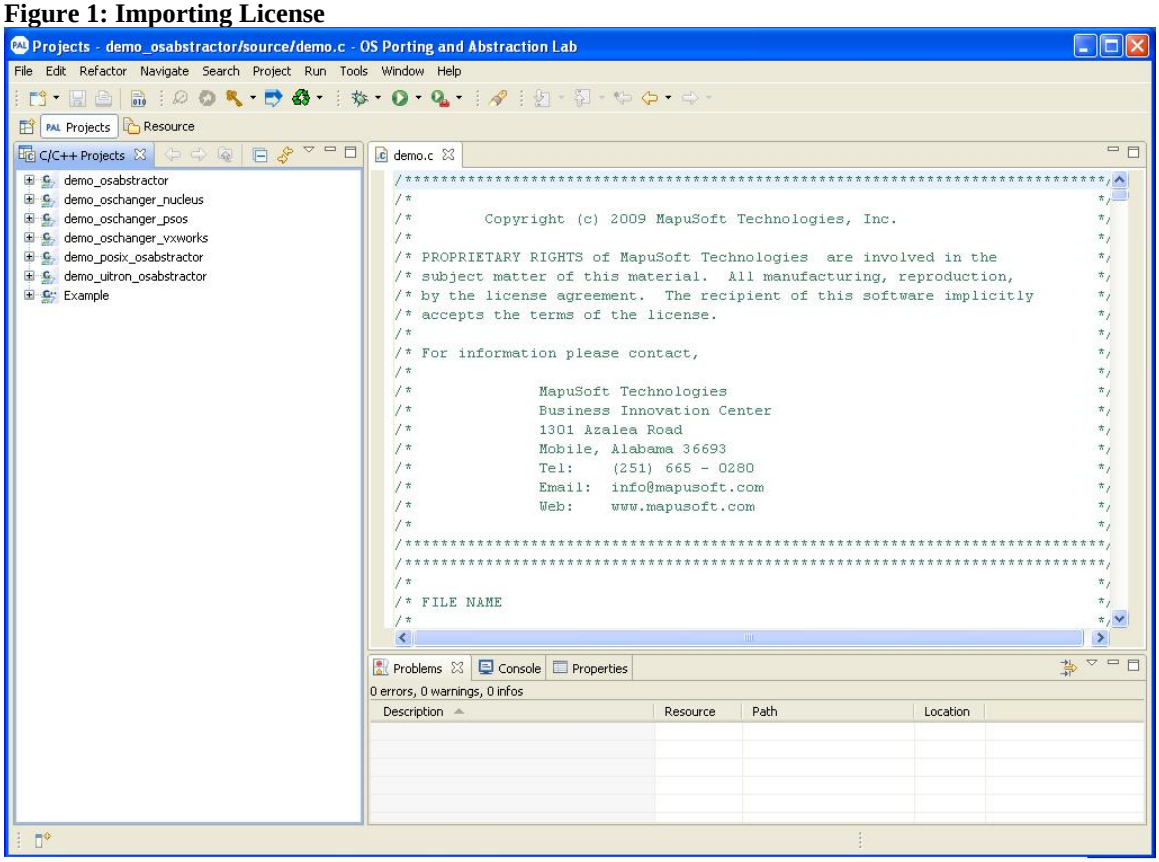
Installing License for OS PAL

MapuSoft provides a license key to the customers. Once the customers provide the Mac Address of their system, MapuSoft Technologies provides a License key for that particular system. This establishes security for the license.

To install the license:

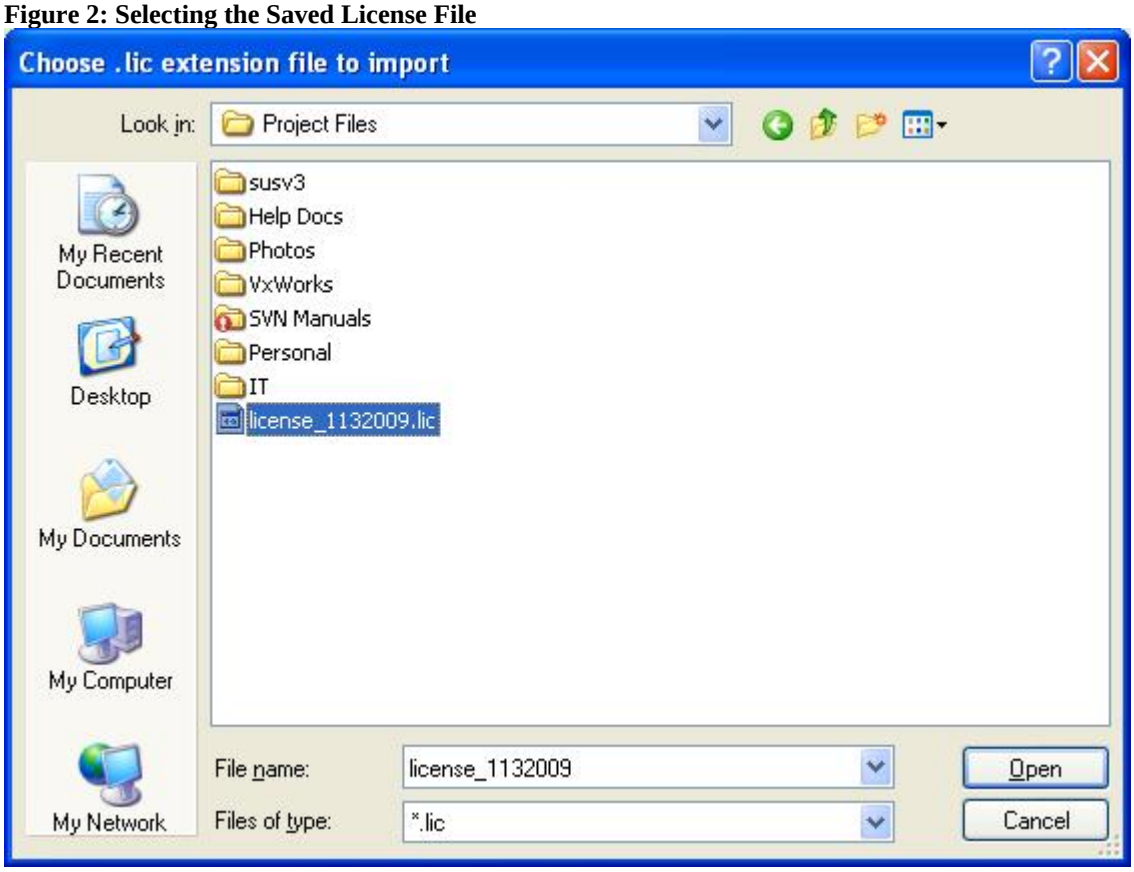
1. Save the license file given to you by MapuSoft.
2. On OS PAL main menu, click the down arrow next to **Key** button and select **Install License** as shown in Figure 1.

Figure 1: Importing License



3. Browse to the location of the saved license file and click **Open** as shown in Figure 2. The license key is installed and now you can work on OS PAL.

Figure 2: Selecting the Saved License File



Uninstalling OS PAL

To uninstall OS PAL:

1. Browse to the installed OSPAL directory and start the **Uninstall** application.
2. You can also uninstall OS PAL by selecting **Control Panel** > **Add/Remove Programs.** Select OS PAL and click **Remove**.
3. There is a possibility of user generated/modified files to be left on your **OSPAL** installation directory. If not necessary, delete the files manually to remove the files.

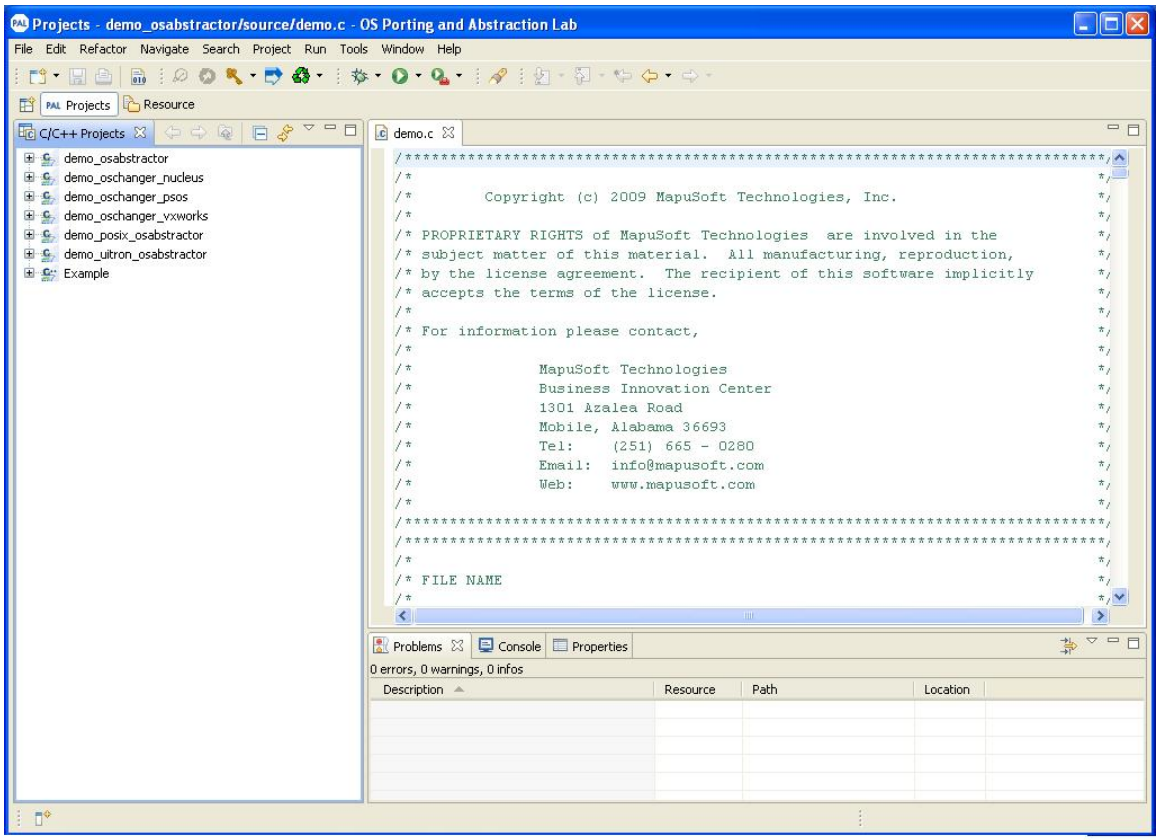
Getting Started With OS PAL

OS PAL Projects Perspective

When you select the OS PAL Projects perspective, you will see the following OS PAL projects as shown in Figure 3:

- Application developed using OS Abstractor APIs (demo_osabstractor)
- Application developed using POSIX APIs (demo_oschanger_posix)
- Application developed using VxWorks APIs (demo_oschanger_vxworks)
- Application developed using Nucleus APIs (demo_oschanger_nucleus)
- Application developed using pSOS APIs (demo_oschanger_psos)
- Application developed using micro-ITRON APIs (demo_osabstractor_uitron)

Figure 3: OS PAL Projects Perspective Page



NOTE: If you want to go to the default Perspective of OS PAL, from the main menu select **Window > Reset Perspective**.

OS PAL Profiler Perspective

NOTE: This feature requires a license. Click http://mapusoft.com/downloads/ospal-evaluation/ to request an evaluation license.

The OS PAL Profiler is an add-on to the established OS PAL Eclipse based code migration and API optimization technology and is designed to enable data collection.

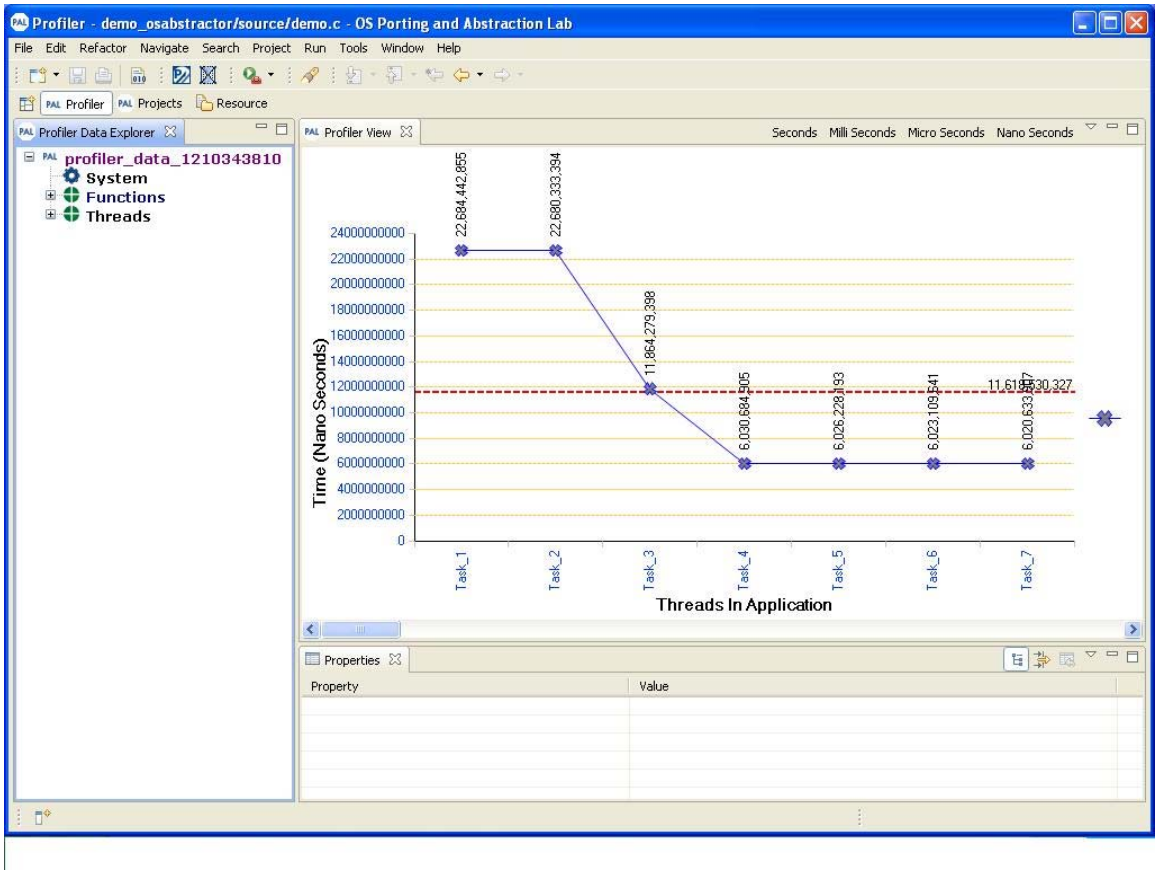
OS PAL Profiler offers the following:

- The data collected by the Profiler provides feedback concerning the utilization of MapuSoft's APIs in the project.
- The reports allow for performance impact analysis by detailing specific API execution time during a particular time period as well as the average and total API execution times.
- Users can analyze the data with the included OS PAL Profiler graphical viewer.This is done offline now and can be used for analysis purpose only. Online data collection of OS PAL Profiler will be incorporated in the future releases. The data collected by the Profiler provides feedback concerning the utilization of MapuSoft's APIs in the project. These reports enable the performance impact analysis by detailing specific API execution time during a particular time period as well as the average and total API execution times. Users can analyze the data with the included OS PAL Profiler graphical viewer which offers area, bar, line, pie, and scatter charts as shown in

Figure 4.

For more information on how to use Profiler, refer to Using OS PAL Profiler section in the manual.

Figure 4: OS PAL Profiler Page



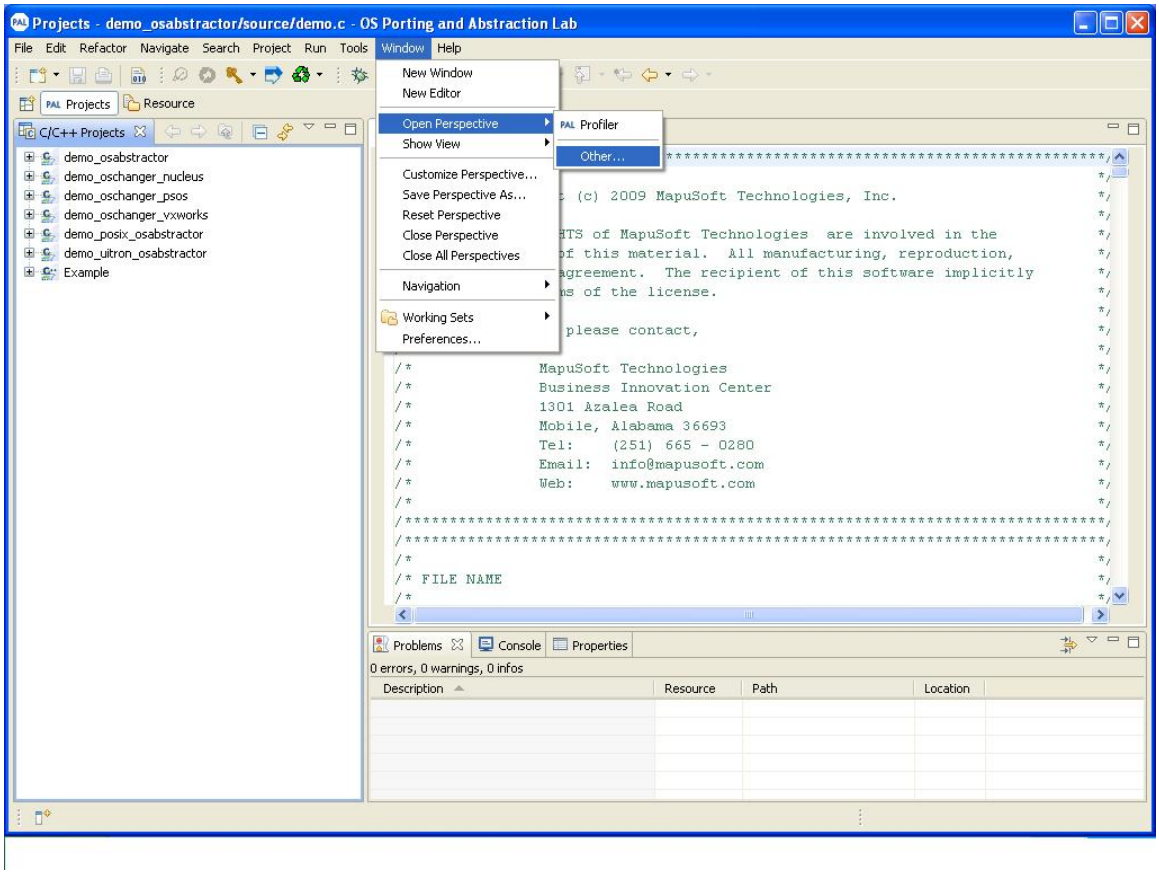
Opening OS PAL Projects Perspective

From OS PAL main menu, click **PAL Projects** perspective button.

Or

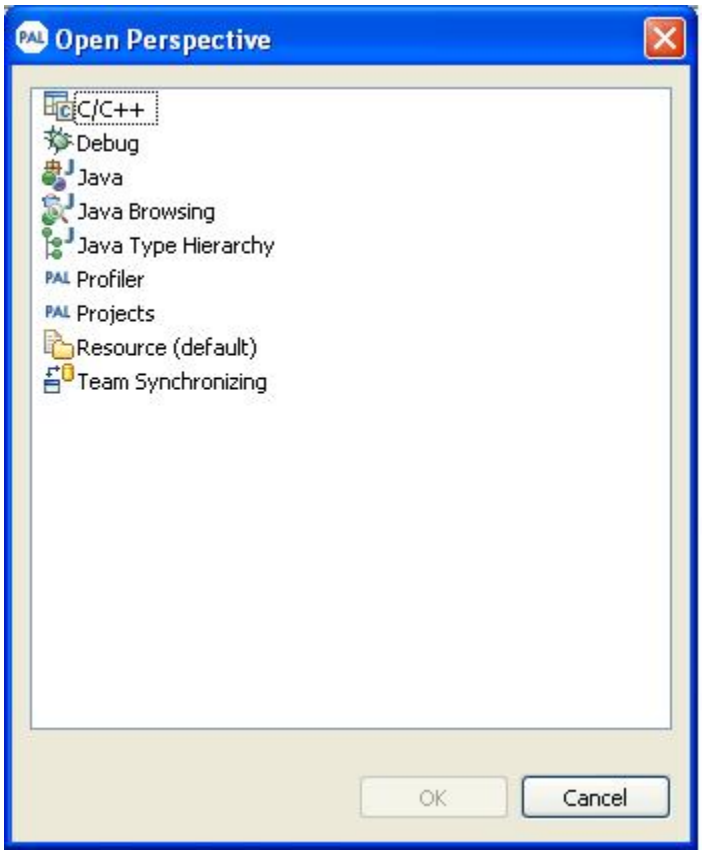
1. From OS PAL main menu, select **Window > Open Perspective > Other** as shown in Figure 5.

Figure 5: Projects Perspective



2. From the Open Perspective window, select **Projects** and click **OK** as shown in Figure 6

Figure 6: Open Perspective Window



Running the Demos Supplied by OS PAL

Example: Running the demo_osabstractor application

NOTE: The example below shows the demo_osabstractor application running in a non-debug mode. By using the run command in non-debug mode, you may experience a problem since the Windows version of standard Eclipse has a problem with flushing stdout in the integrated console. Sometimes, the output will not be displayed until the pipe is flushed since the console uses a windows pipe. For this reason, it is recommended that you use the debug mode to run the application instead, since it uses a separate console window for stdout. Click the following links to find out the details of related eclipse bugzilla entries for more information:

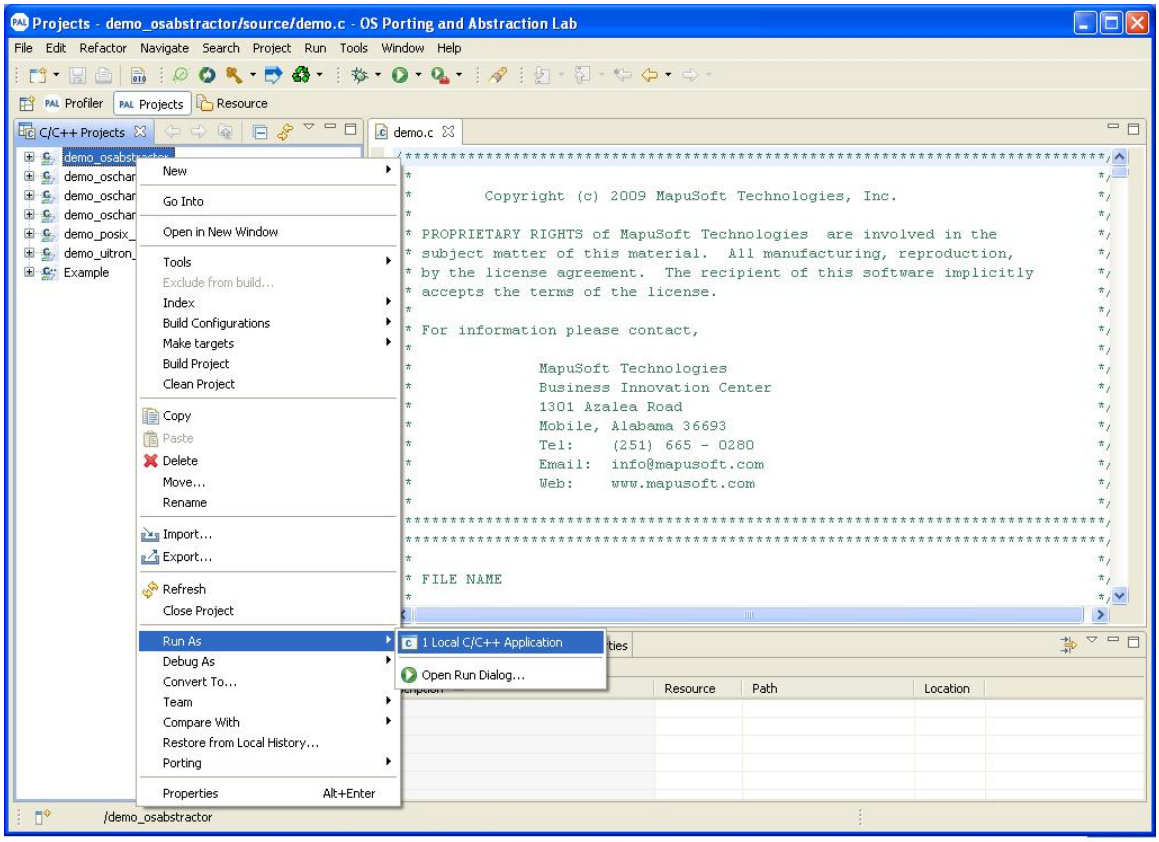
https://bugs.eclipse.org/bugs/show_bug.cgi?id=58081
https://bugs.eclipse.org/bugs/show_bug.cgi?id=27663
https://bugs.eclipse.org/bugs/show_bug.cgi?id=102043

NOTE: If there is an error launching the binaries, right click on the project and click **Refresh**.

To run the demos supplied by OS PAL:

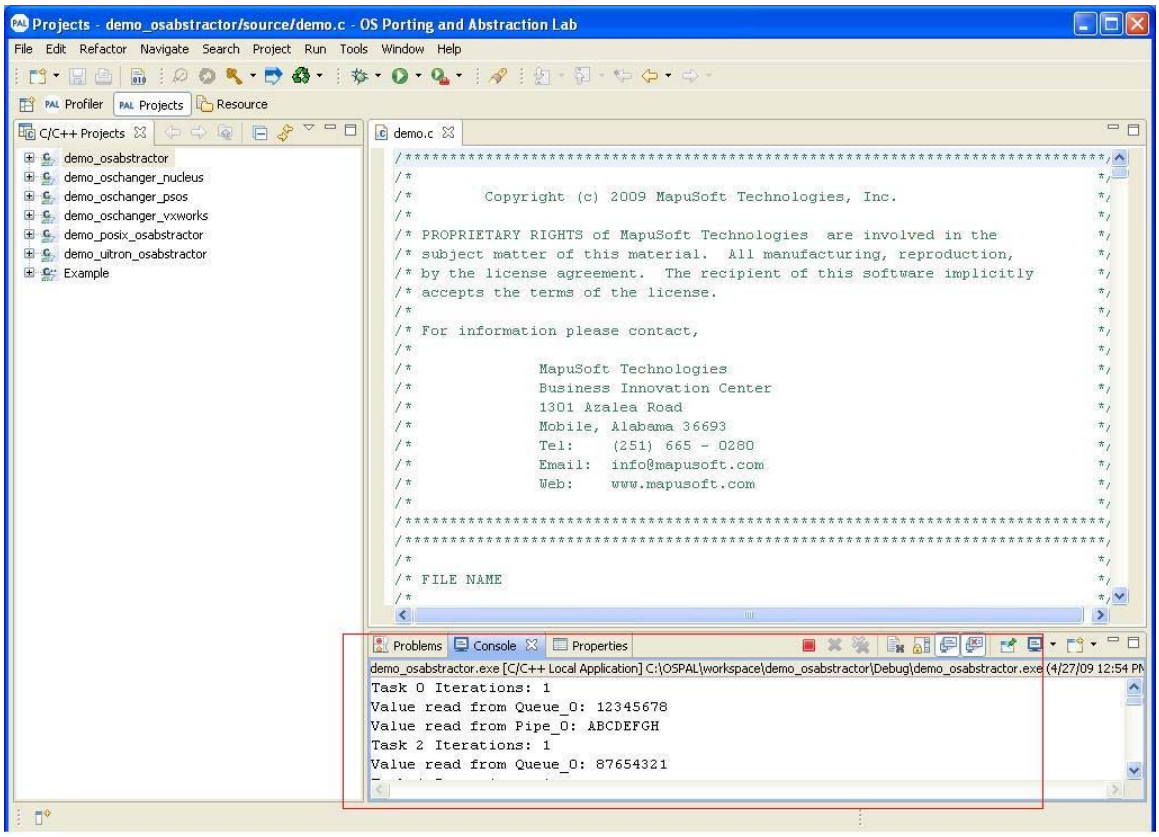
1. From OS PAL main window, select **demo_osabstractor project**.
2. Right click on the project and select **Run As > Local OS PAL C/C++ Application** as shown in the Figure 7.

Figure 7: Running a Demo Application



3. The demo application console is seen as shown in Figure 8. The highlighted portion is the output for the demo application.

Figure 8: Output for Demo Application

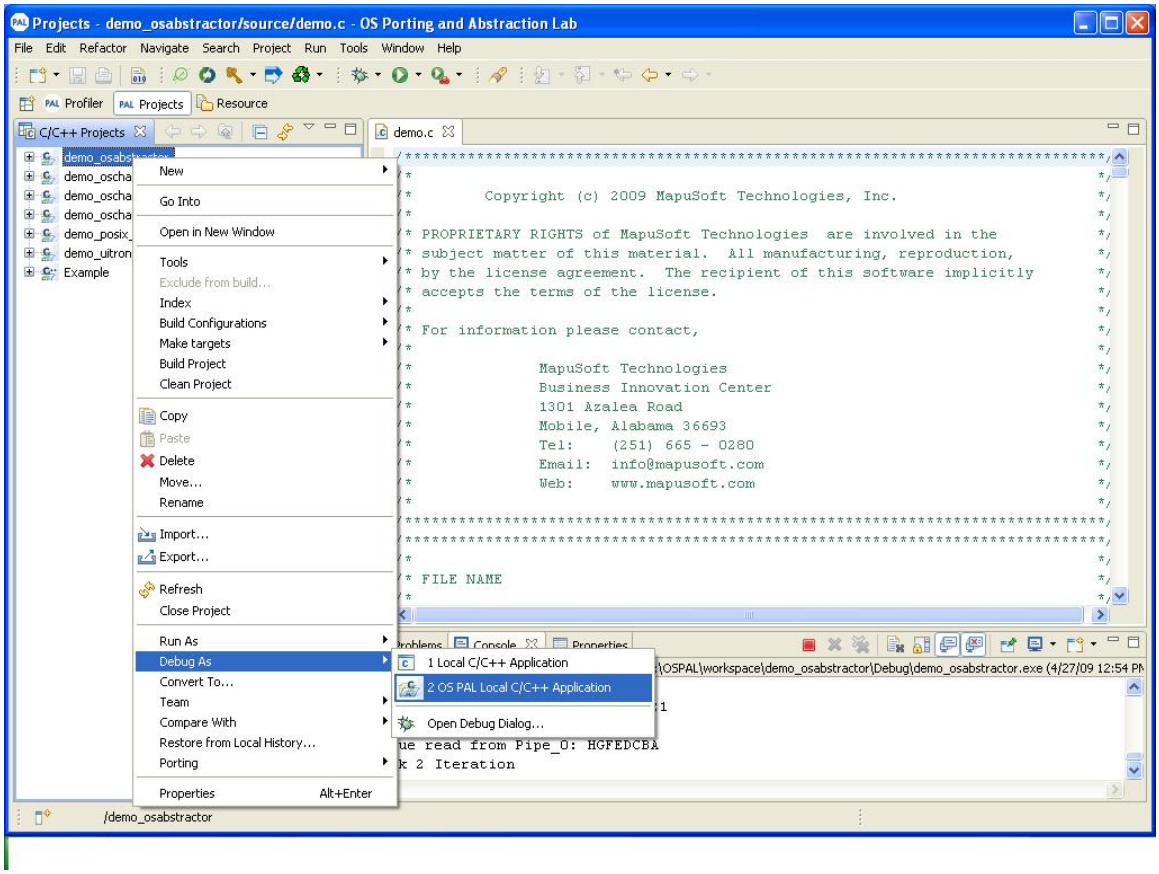


Debugging the Demos Supplied by OS PAL

Example: Debugging the demo_osabstractor application

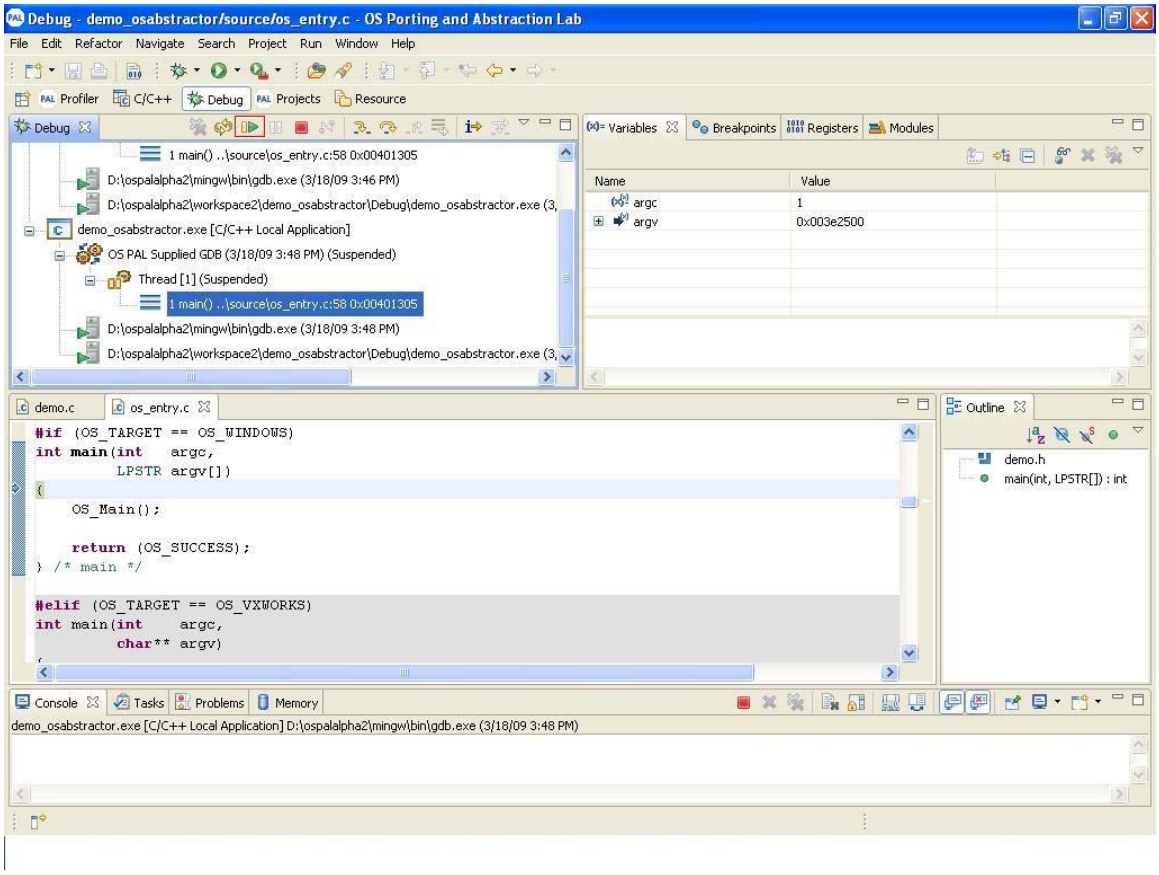
1. From the OS PAL main window, select the osabstractor_demo project.
2. Right click on the project and select **Debug as > Local OS PAL C\C ++ Application** or click on the debugging icon as highlighted in Figure 9.

Figure 9: Debugging the Demo Application



3. Click **Resume** icon to resume the debugging process as shown in Figure 10.

Figure 10: Resume Debugging process



4. The debugging resumes as shown in Figure 11.

Figure 11: Debug Demo Application Perspective

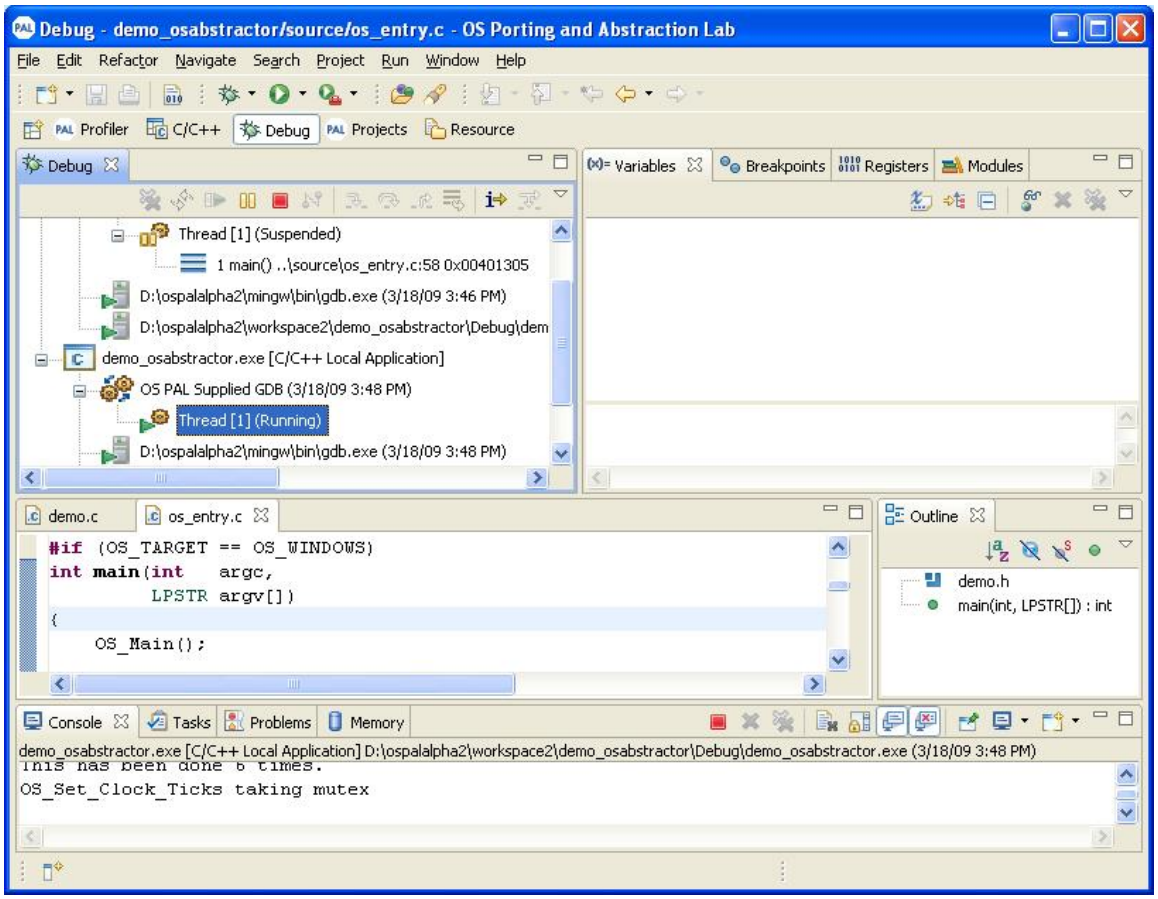
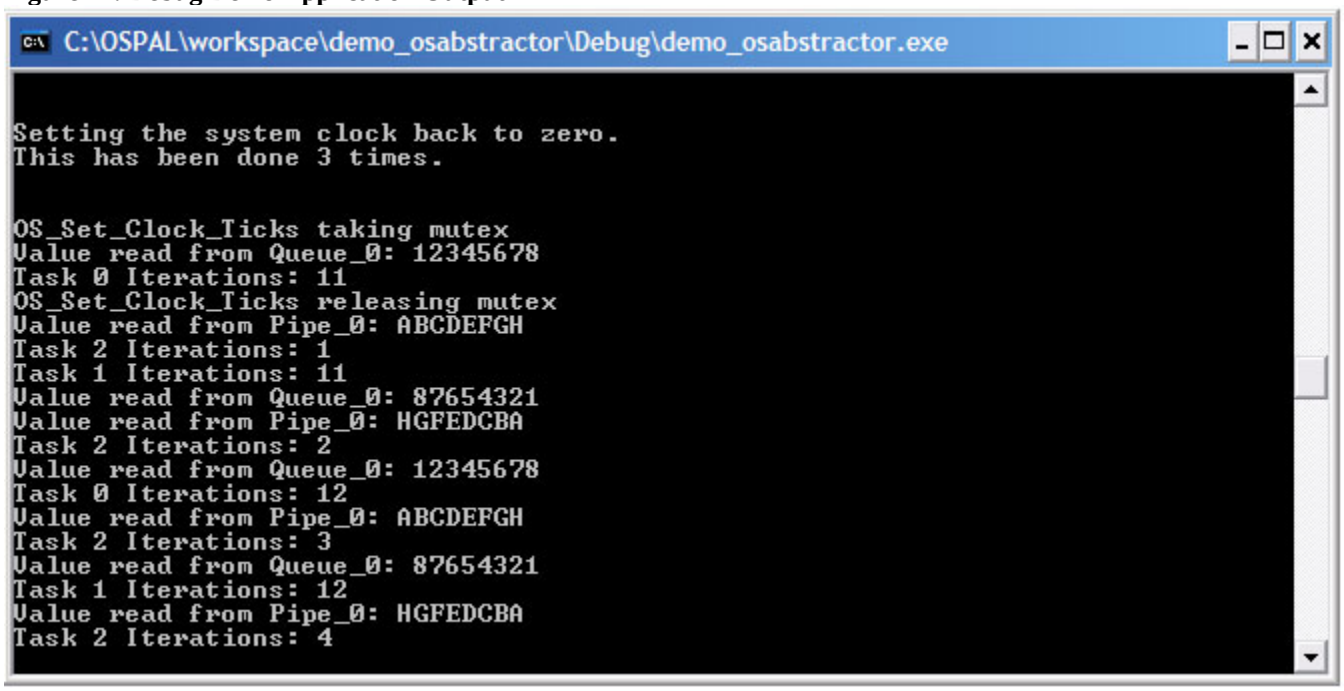


Figure 12: Debug Demo Application Output

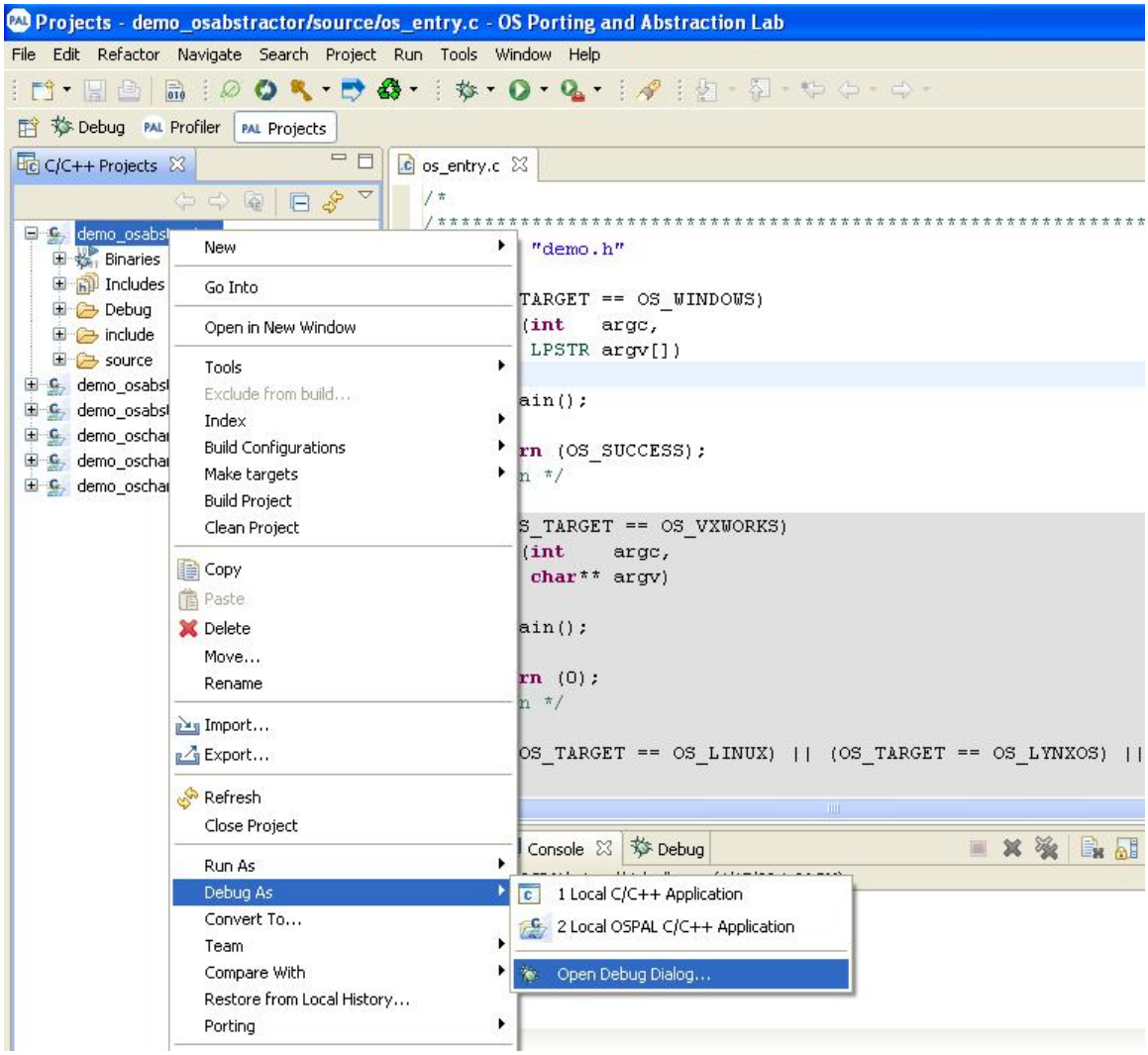


Debugging Using External Console/Terminal

Debugging can be done using an external console or terminal in the following way:

1. From the OS PAL main window, select the osabstractor_demo project.
2. Right click on the project and select **Debug as > Open Debug Dialog** as shown in Figure 13.

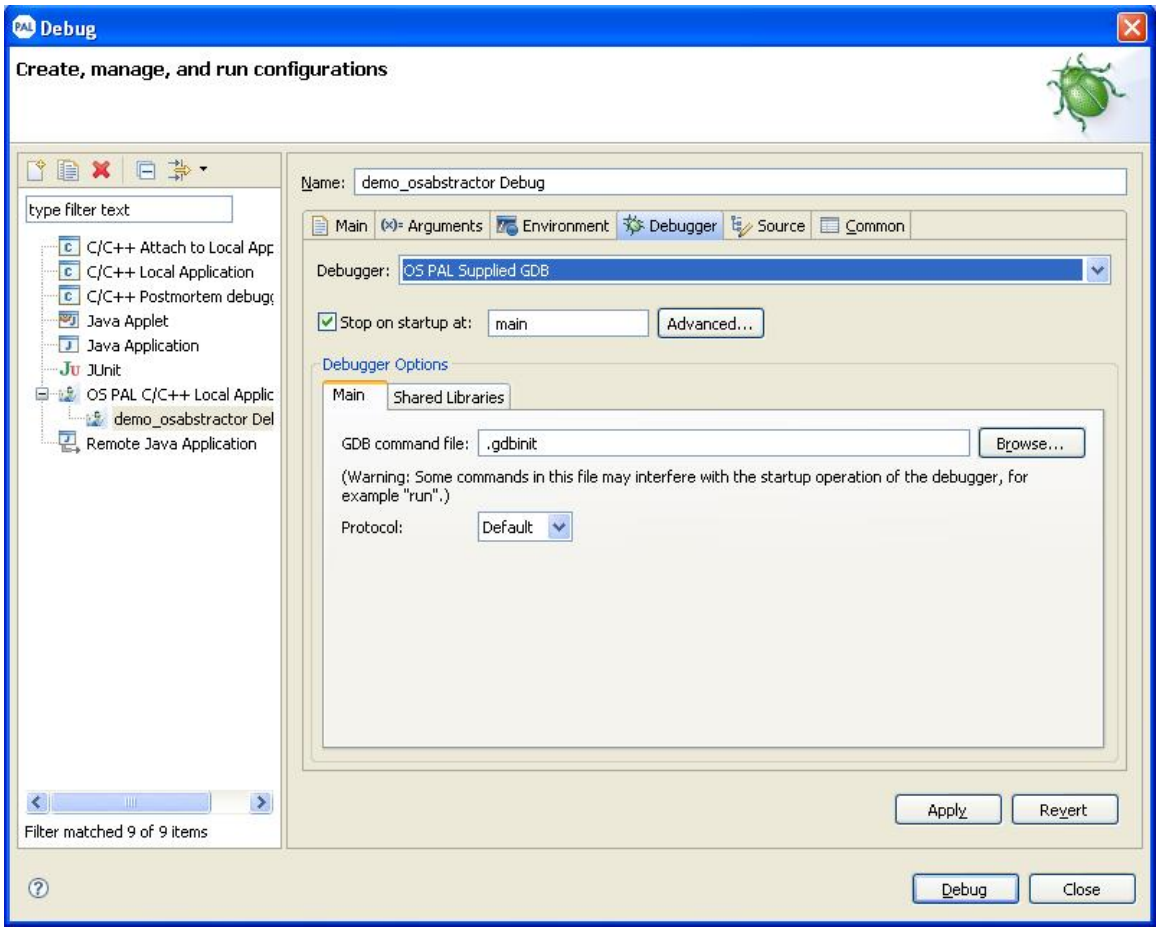
Figure 13: Open Debug Dialog



3. On the Debug Configuration window, you can set your options for debugging as shown in the Figure 14. You can use the OS Pal Supplied GDB to execute debugging.

NOTE: OS PAL does not support Cygwin tools and its use is not recommended.

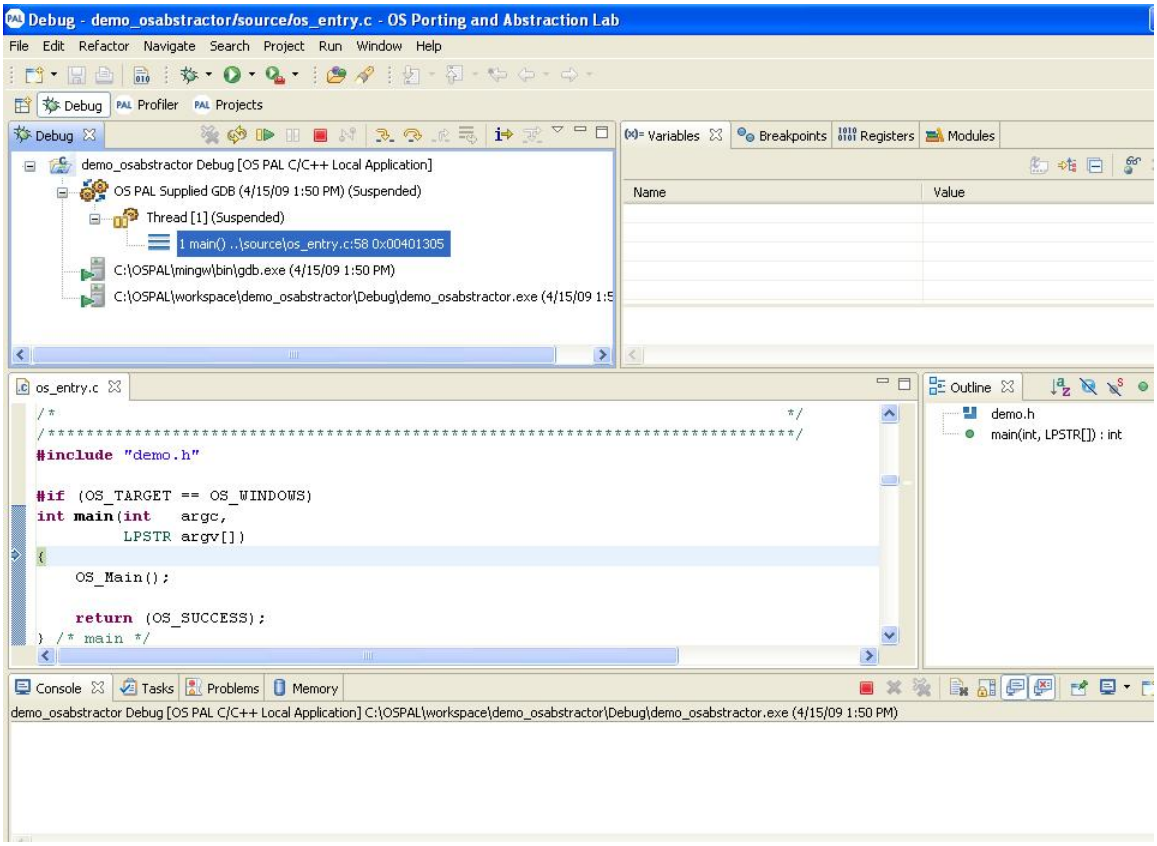
Figure 14: Debug Configuration Window



4. You change any of the options here and click **Apply.**

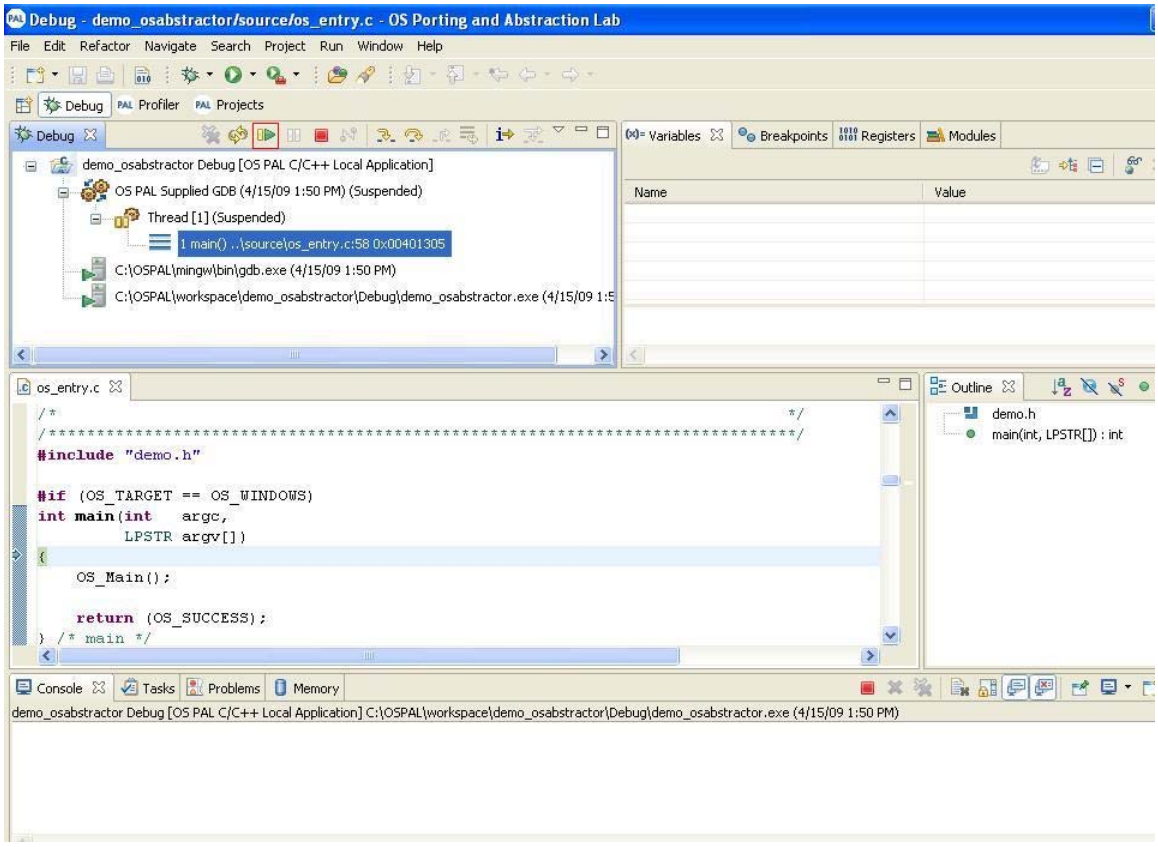
5. Click **Debug** to execute debugging using the external console or Terminal**.** You can view the debugging process in your console as shown in Figure 15.

Figure 15: Debugging Output Using External Console/Terminal



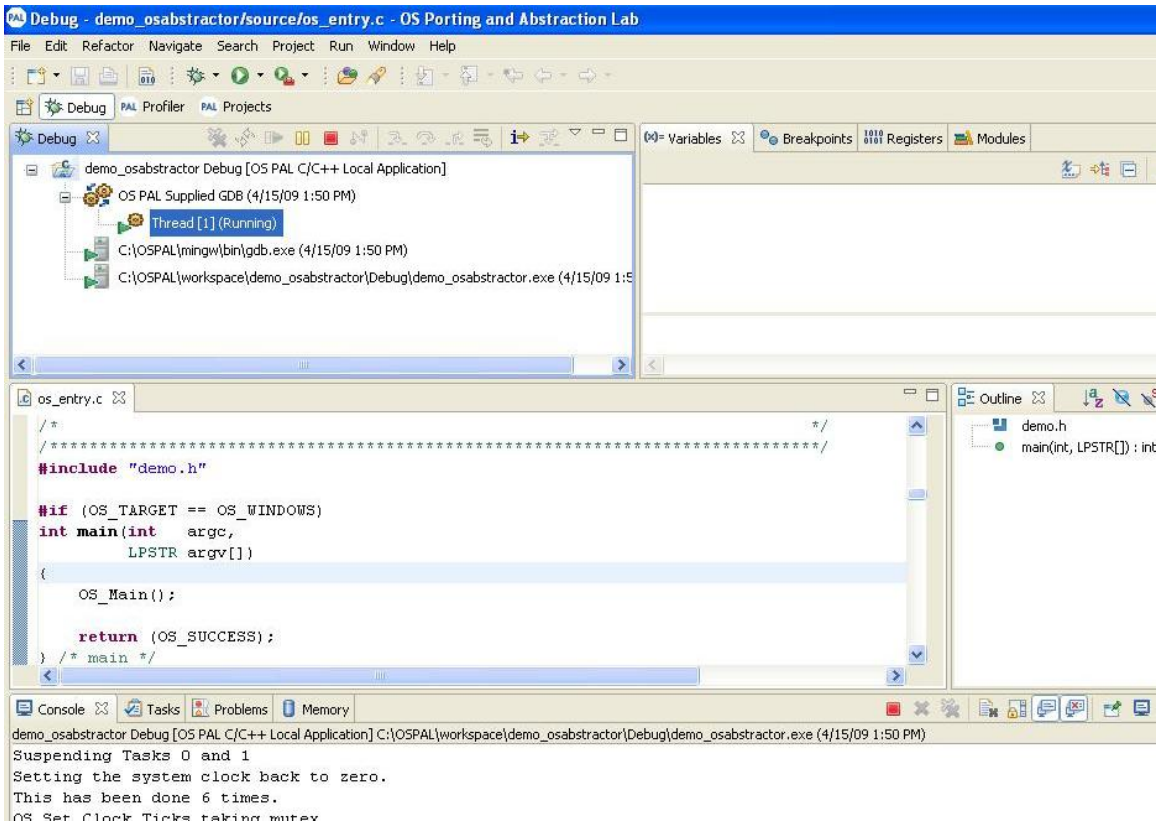
6. To resume debugging, click the resume icon  on the debugging window as shown in the Figure 16.

Figure 16: Resume Debugging Using External Console/Terminal



7. You have now successfully finished debugging by using external console or terminal as shown in Figure 17.

Figure 17: Debugging in Progress



DEVELOPING AN APPLICATION IN OS PAL

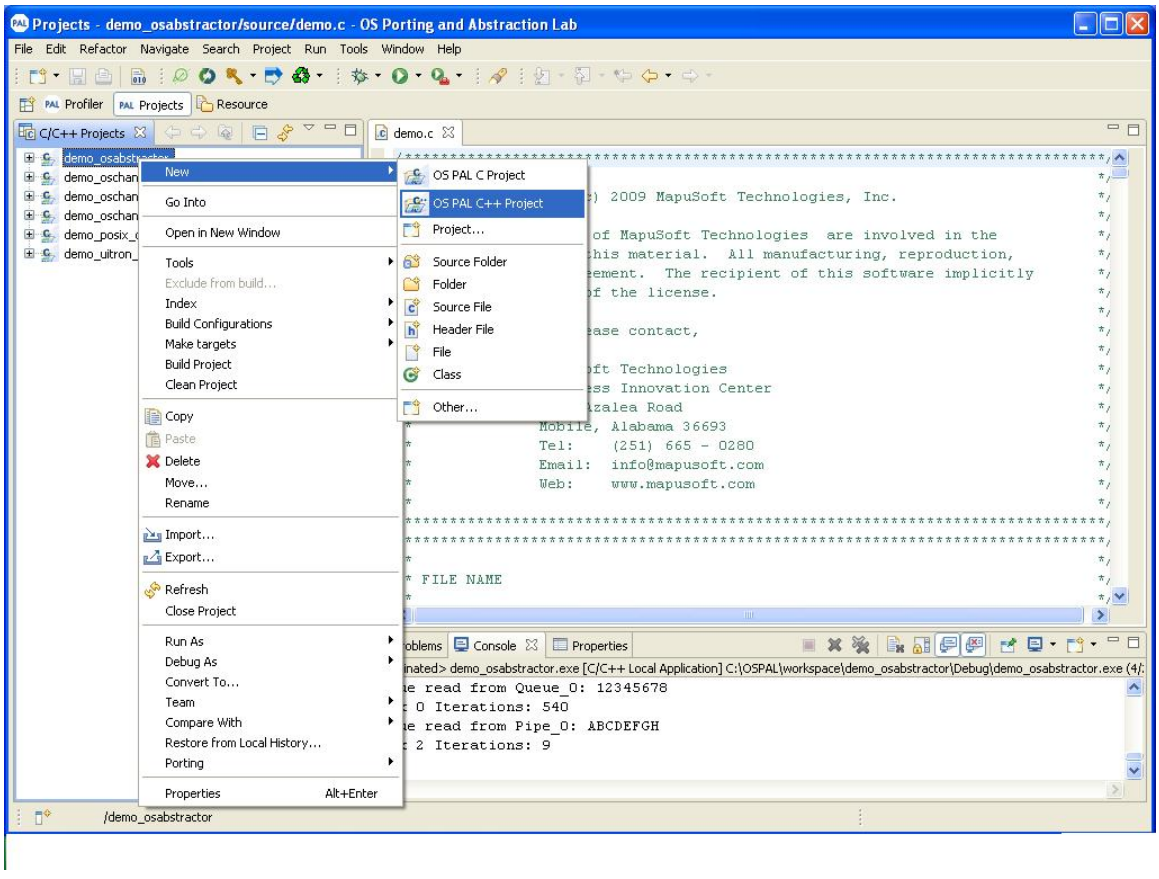
Creating an OS PAL C/C++ Project

NOTE: This feature requires a license. Click http://mapusoft.com/downloads/ospal-evaluation/ to request an evaluation license.

To create an OS PAL C/C++ project:

1. From OS PAL main window, select any project under C/C++ **Projects** tab on the left pane.
2. Select **New > OS PAL C/C++ Project** as shown in Figure 18.

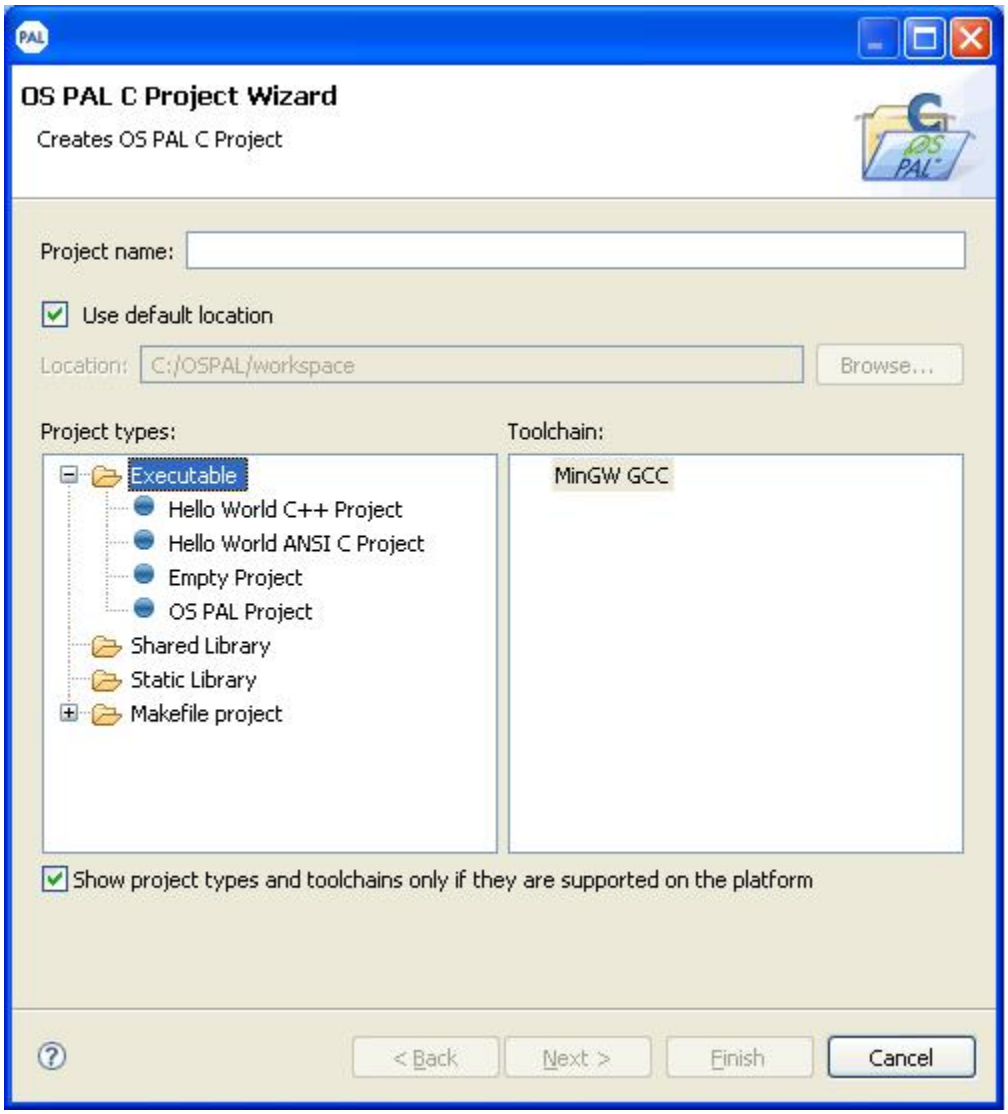
Figure 18: Creating an OS PAL C/C++ Project



3. On OS PAL C Project Wizard window, type a project name and give a location next to **Project Name** text box.

4. Under Project Types, expand the **Executable** menu. Select **OS PAL Project** and click **Next** as shown in Figure 19.

Figure 19: OS PAL C Project Wizard Window



The following are the project types:

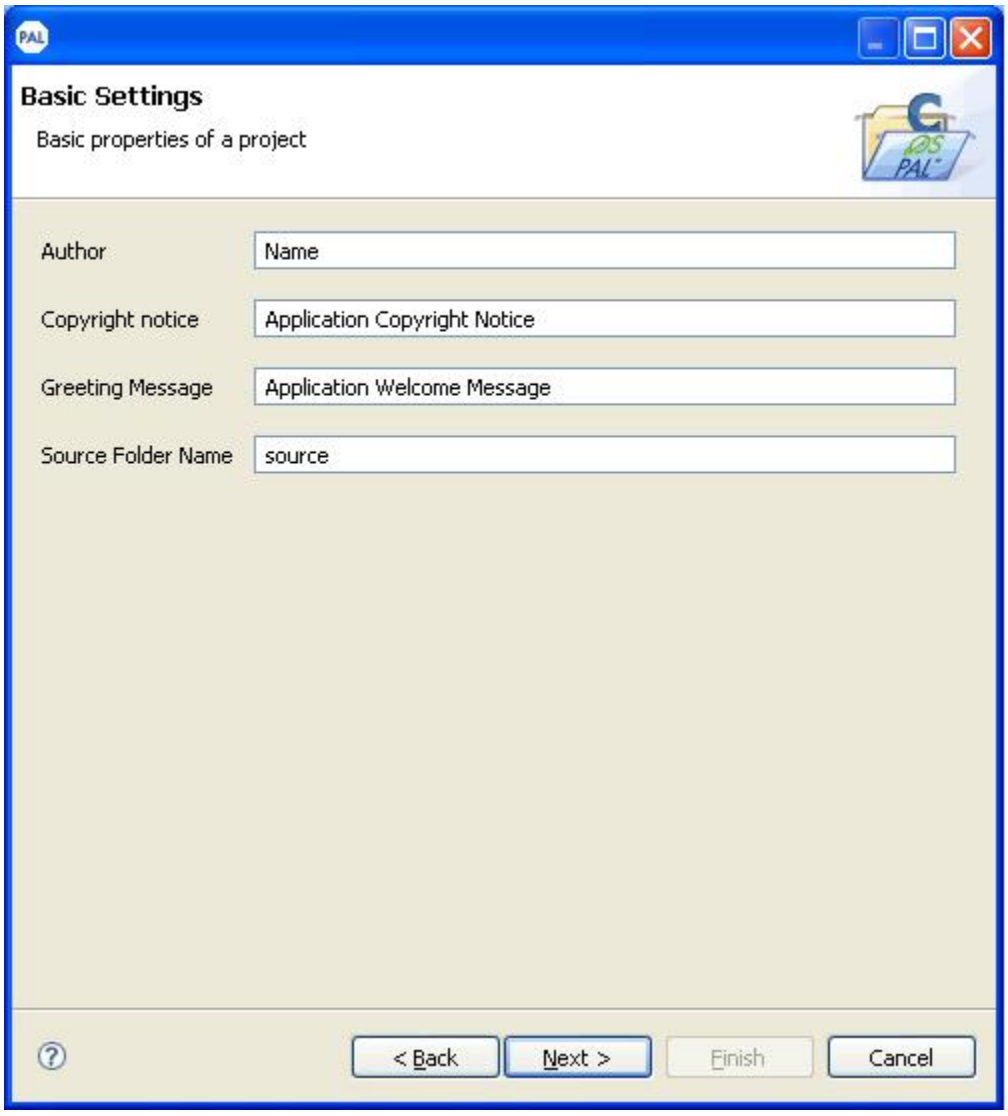
Executable - Provides an executable application. This project type folder contains three templates. Makefile for the Executable project type is automatically created by the CDT.

- **Hello World C++ Project** provides a simple C++ Hello World application with main().
- **Hello World ANSI C Project** provides a simple C Hello World application with main().
- **Empty Project** provides a single source project folder that contains no files.

Shared Library - An executable module that is compiled and linked separately. When you create a project that uses a shared library (libxx.so), you define your shared library's project as a Project Reference for your application. The makefile for this project type is automatically created by the CDT.
Static Library - A collection of object files that you can link into another application (libxx.a). The CDT combines object files (i.e. *.o) into an archive (*.a) that is directly linked into an executable. The makefile for this project type is automatically created by the CDT.
Makefile Project – This creates an empty project without the meta-data files. This selection is useful for importing and modifying existing makefile-based projects; a new makefile is not created for this project type.

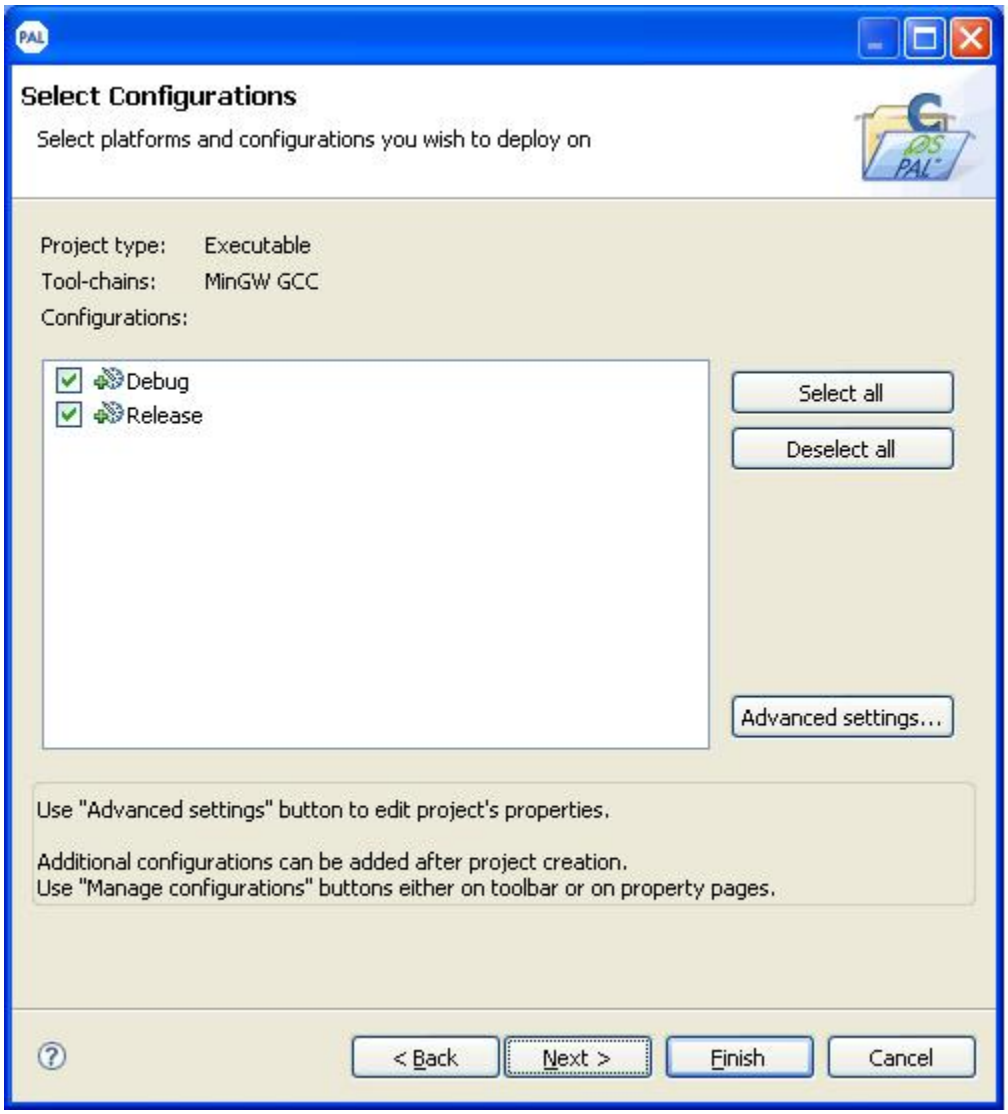
5. On the Basic Settings window, define the basic properties of your project and click **Next** as shown in Figure 20Figure 20:.

Figure 20: Basic Settings Window



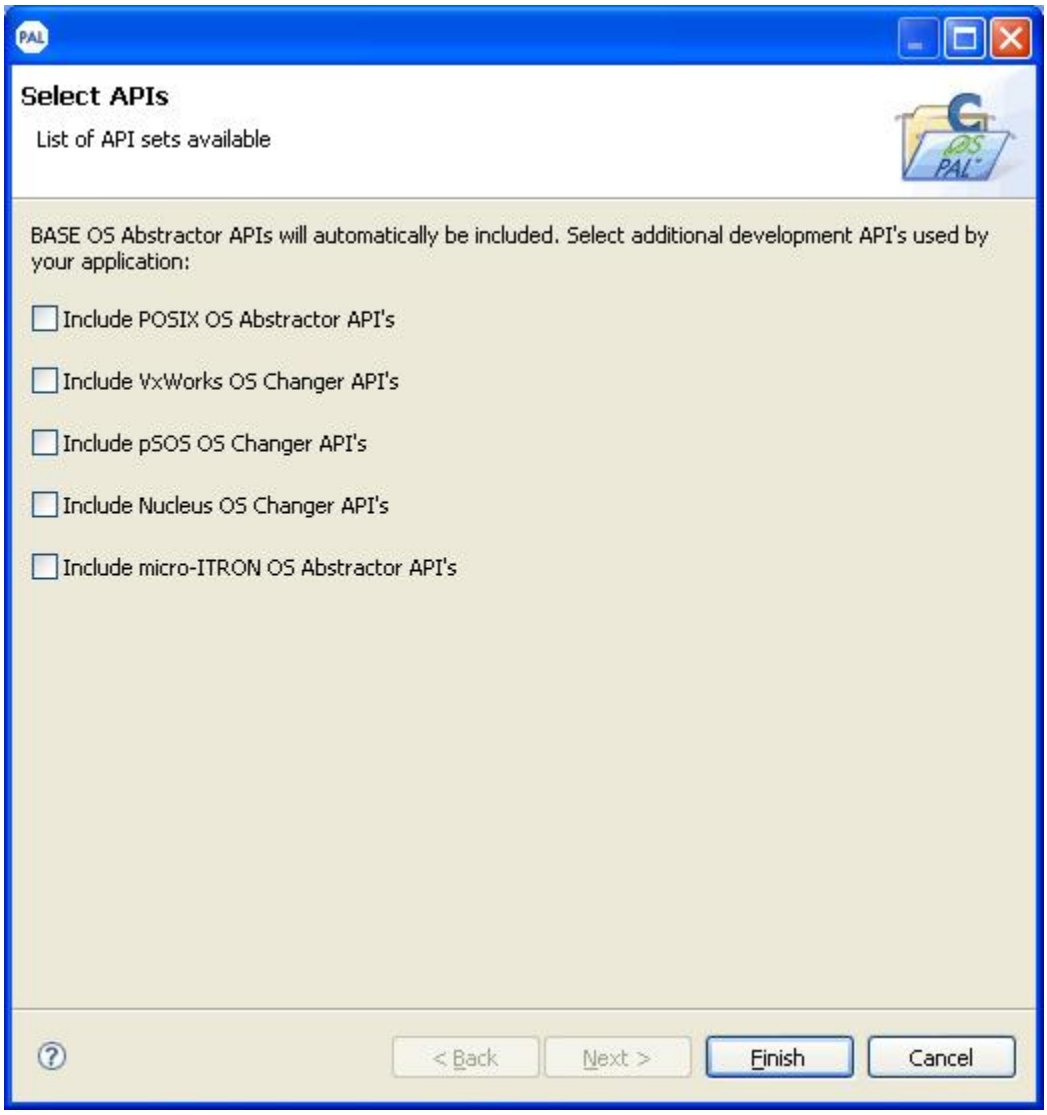
6. On the Configurations window, select the platforms and configurations for deployment and click **Next** as shown in Figure 21.

Figure 21: Configurations Window



7. On the Select APIs window, select the required OS PAL development APIs and click **Finish** as shown in Figure 22.

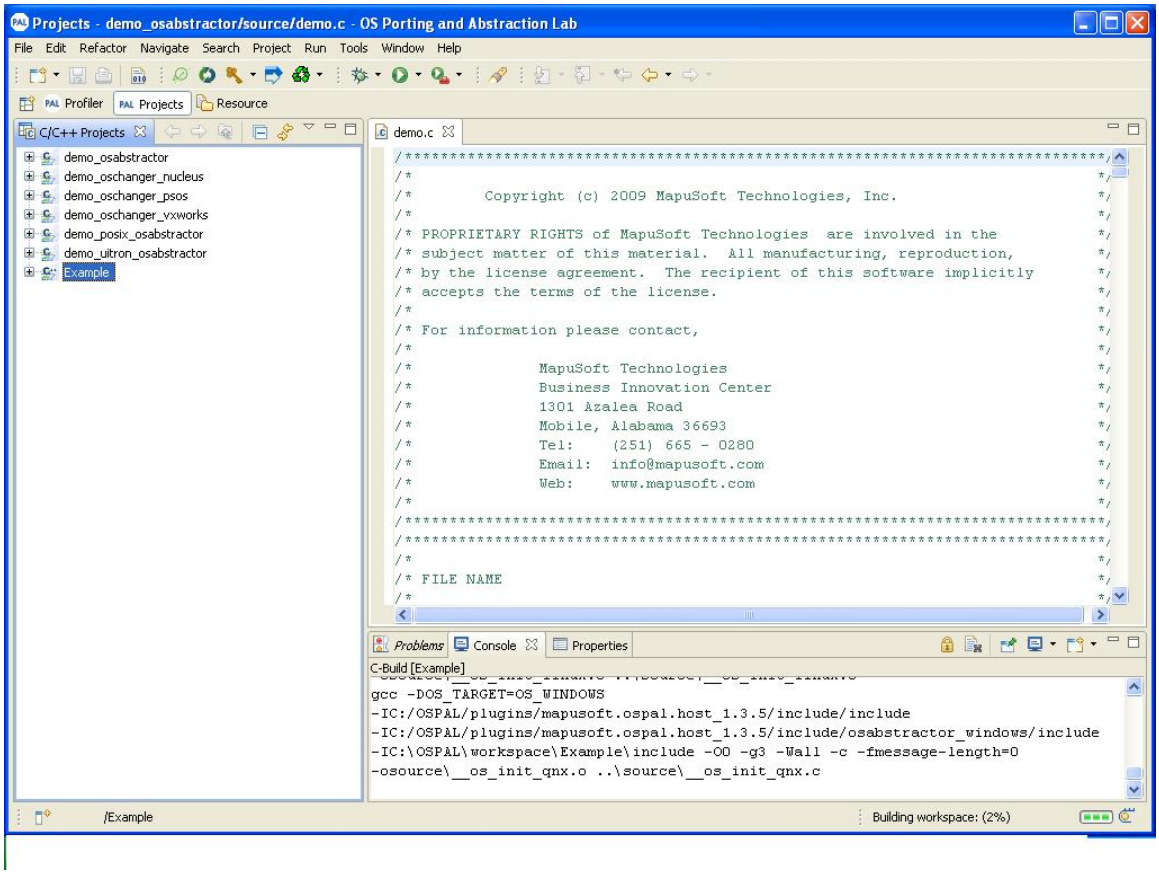
Figure 22: Select APIs Window



For more information about using the development APIs, click here http://mapusoft.com/contact/ to send a request for the reference manual.

8. You will see the output as shown in Figure 23.

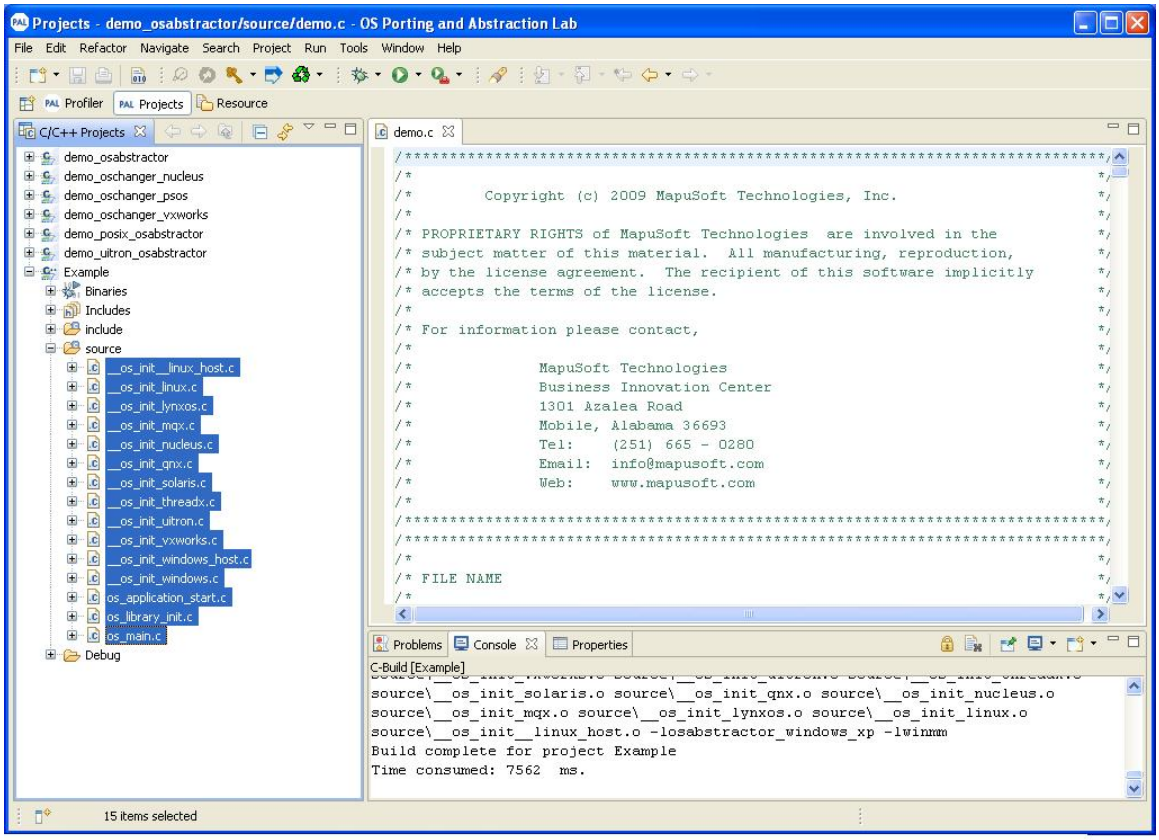
Figure 23: Creating OS PAL C/C++ Project Output



OS PAL C Project Template Files

To view the C project template files, expand the project folder you have just created by clicking on the + sign beside the Project name as shown in Figure 24**.**

Figure 24: C Project Template Files



You can view the following template files for your project on the left pane of the window:

- **__os_init__linux_host.c**–This function is the entry function for the native operating system.This is where you should put any of your operating system specific code. For instance, if you wanted to add a signal handler on Linux, you could do it here before calling OS_Main().
- **__os_init_linux.c–**This function is the entry function for the native operating system. This is where you should put any of your operating system specific code. For instance, if you wanted to add a signal handler on Linux, you could do it here before calling OS_Main().

- **__os_init_lynxos.c–** This function is the entry function for the native operating system. This is where you should put any of your operating system specific code. For instance, if you wanted to add a signal handler on lynxos, you could do it here before calling OS_Main().
- **__os_init_mqx.c–** This function is the entry function for the native operating system. This is where you should put any of your operating system specific code. For instance, if you wanted to add a signal handler on MQX, you could do it here before calling OS_Main().
- **__os_init_nucleus.c–** This function is the entry function for the native operating system. This is where you should put any of your operating system specific code. For instance, if you wanted to add a signal handler on Nucleus, you could do it here before calling OS_Main().
- **__os_init_qnx.c–** This function is the entry function for the native operating system. This is where you should put any of your operating system specific code. For instance, if you wanted to add a signal handler on QNX, you could do it here before calling OS_Main().
- **__os_init_solaris.c–** This function is the entry function for the native operating system. This is where you should put any of your operating system specific code. For instance, if you wanted to add a signal handler on Solaris, you could do it here before calling OS_Main().
- **__os_init_threadx.c–** This function is the entry function for the native operating system. This is where you should put any of your operating system specific code. For instance, if you wanted to add a signal handler on ThreadX, you could do it here before calling OS_Main().
- **__os_init_uitron.c–** This function is the entry function for the native operating system. This is where you should put any of your operating system specific code. For instance, if you wanted to add a signal handler on micro-ITRON, you could do it here before calling OS_Main().
- **__os_init_vxworks.c–** This function is the entry function for the native operating system. This is where you should put any of your operating system specific code. For instance, if you wanted to add a signal handler on VxWorks, you could do it here before calling OS_Main().
- **__os_init_windows_host.c–**These functions are the various entry functions for the different operating systems. This is where you should put any of your operating system specific code. For instance, if you wanted to add a signal handler on Linux, you could do it here before calling OS_Main().When optimizing, you will need to write an equivalent function for your target operating system.
- **__os_init_windows.c–** These functions are the various entry functions for the different operating systems. This is where you should put any of your operating system specific code. For instance, if you wanted to add a signal handler on Linux, you could do it here before calling OS_Main(). When optimizing, you will need to write an equivalent function for your target operating system**.**

- **os_application_start.c–** This function is the first OS agnostic function and should be the start point for the application development**.**
- **os_library_init.c–** This function initializes the required OS Abstractor and OS Changer products and creates the entry threads for each product.
- **Os_main.c–** This function initializes the OS Abstractor layer and calls OS_Application_Wait_For_End which will suspend until OS_Application_Free or OS_Delete_Process is called. It also spawns the first OS independants thread which is the true entry point for OS Abstractor.

The application code starts in the os_library_init.c file, the user defined entry function and name of the application for base osabstractor can be specified in:
#define OS_ABSTRACTOR_BASE_ENTRY_FUNCTION
#define OS_APPLICATION_START_TASK_NAME

For Vxworks OS Changer, the user defined entry function and stack size can be specified in:
#define VXWORKS_ENTRY_FUNCTION
#define VXWORKS_ENTRY_FUNCTION_STACK_SIZE

Similarly, this is how it works for all the remaining changers/abstractors.

You can insert code that is only included when they use OS PAL host in the following way:

- For windows host you can insert code in __os_init_windows_host.c (inside main function before calling OS_MAIN), that is only included in OS PAL windows host.
- For Linux host, you can insert code in __os_init_linux_host.c (inside main function before calling OS_MAIN), that is only included in OS PAL Linux host.

You can insert code that is specific to a target OS (inside main function before calling OS_MAIN) in the following way:

- For LynxOS target, insert in __os_init_lynxos.c
- For MQX target, insert in __os_init_mqx.c
- For Linux target, insert in __os_init_linux.c
- For Nucleus target, insert in __os_init_nucleus.c
- For QNX target, insert in __os_init_qnx.c
- For Solaris target, insert in __os_init_solaris.c
- For Threadx target, insert in __os_init_threadx.c
- For uITRON target, insert in __os_init_uitron.c
- For VxWorks target, insert in __os_init_vxworks.c

HOST Defines: The below defines are the system settings used by the OS_Application_Init() function. Use these to modify the settings when running on the host. A value of -1 for any of these will use the default values located in osabstractor_usr.h. When you optimize for the target side code, the wizard will create a custom osabstractor_usr.h using the settings you specify at that time so these defines will no longer be necessary.

```
#define HOST_DEBUG_INFO                     -1
#define HOST_TASK_POOL_TIMESLICE            -1
#define HOST_TASK_POOL_TIMEOUT              -1
#define HOST_ROOT_PROCESS_PREEMPT           -1
#define HOST_ROOT_PROCESS_PRIORITY          -1
#define HOST_ROOT_PROCESS_STACK_SIZE        -1
#define HOST_ROOT_PROCESS_HEAP_SIZE         -1
#define HOST_DEFAULT_TIMESLICE              -1
#define HOST_MAX_TASKS                      -1
#define HOST_MAX_TIMERS                     -1
#define HOST_MAX_MUTEXES                    -1
#define HOST_MAX_PIPES                      -1
#define HOST_MAX_PROCESSES                  -1
#define HOST_MAX_QUEUES                     -1
#define HOST_MAX_PARTITION_MEM_POOLS        -1
#define HOST_MAX_DYNAMIC_MEM_POOLS          -1
#define HOST_MAX_EVENT_GROUPS               -1
#define HOST_MAX_SEMAPHORES                 -1
#define HOST_USER_SHARED_REGION1_SIZE       -1
```

OS_HOST: This flag is used only in OS PAL environment. It is not used in the target environment.
Host mode defines can be modified in OS_MAIN.C file. For example, modify maximum tasks under host environment in HOST_MAX_TASKS.

NOTE: You can manually change the values in the Code Optimization Wizard. Refer to DEVELOPING TARGET CODE WITH OS PAL section in the manual.

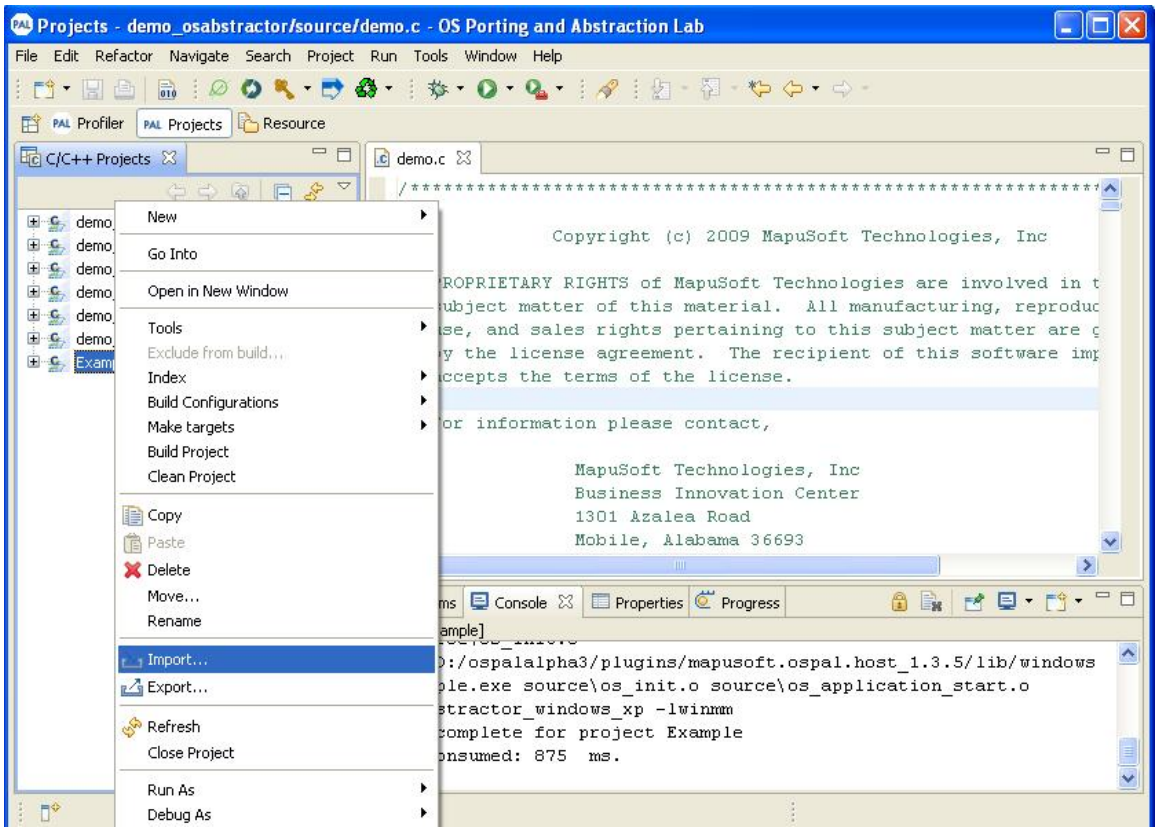
Adding Source Code Files to OS PAL project

NOTE: This feature requires a license. Click http://mapusoft.com/downloads/ospal-evaluation/ to request an evaluation license.

To add source code files:

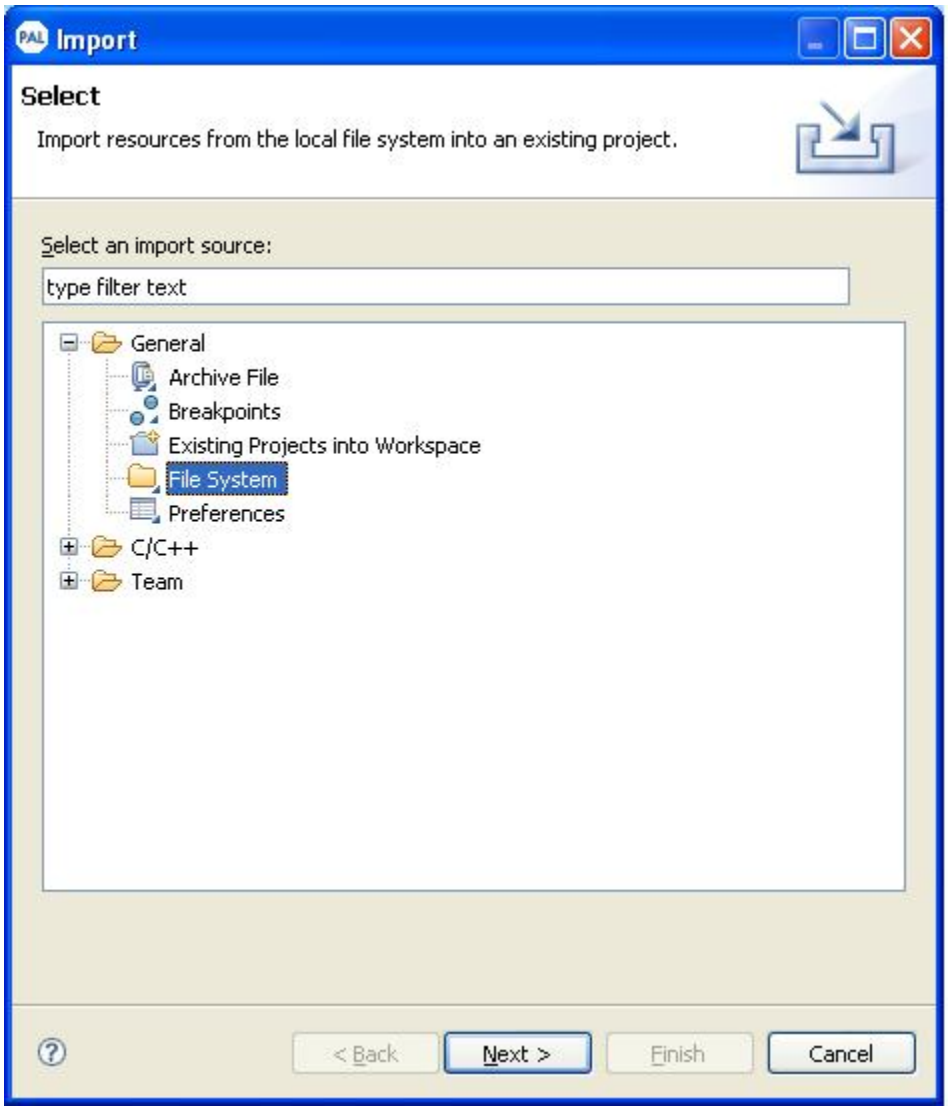
1. Select a project under C/C++ Projects pane.
2. Right click on it and select **Import** as shown in Figure 25.

Figure 25: Adding Source Code Files



3. Select your import source and click **Next** as shown in Figure 26.

Figure 26: Importing Files from Local File System

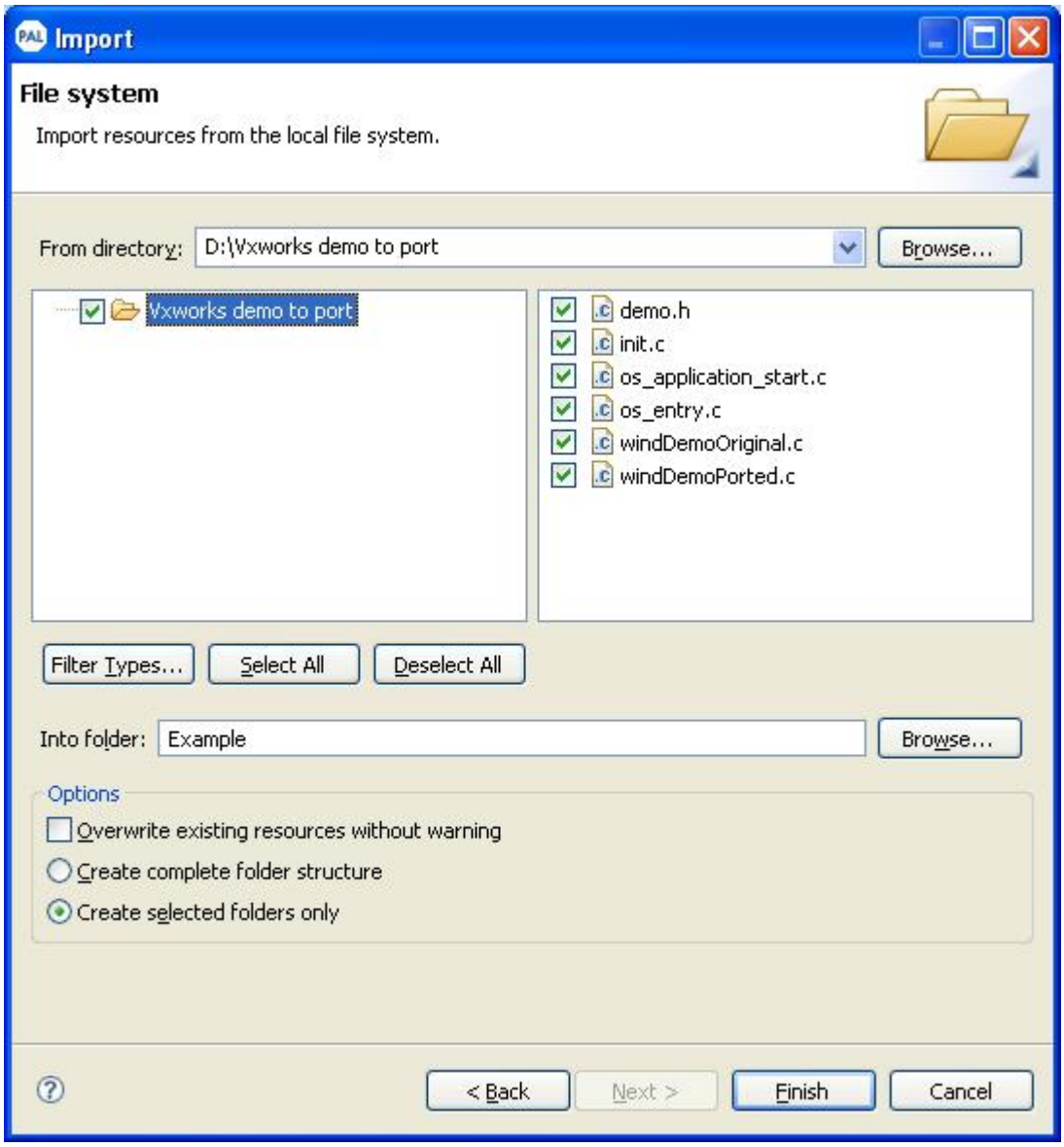


4. Select the directory on your local file system which contains the source code files and click **OK** as shown in Figure 27.

Figure 27: Importing Source Code Files from Directory

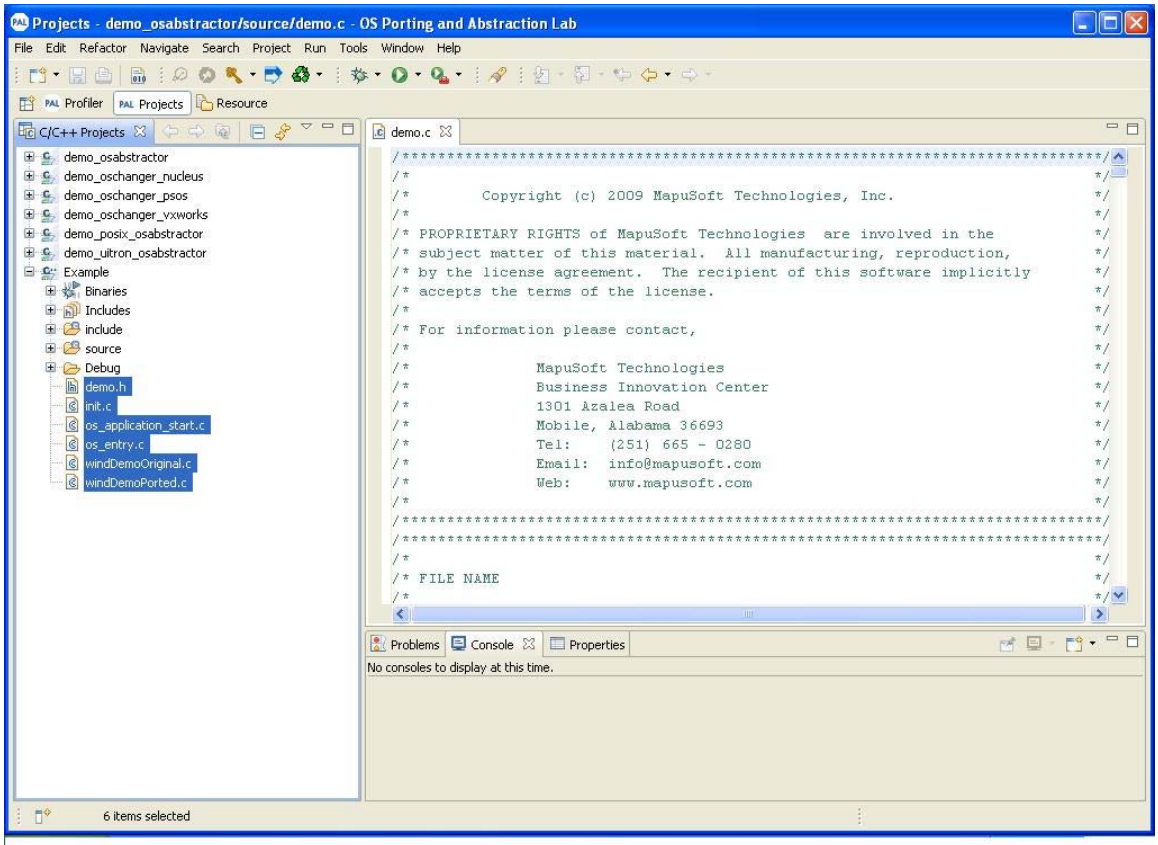
5. Select the check boxes corresponding to the source code files you want to import and click **Finish** as shown in Figure 28.

Figure 28: Selecting Source Code Files for Importing



6. You can view the source code files added to your OS PAL project as shown in Figure 29.

Figure 29: Importing Source Code Files Output

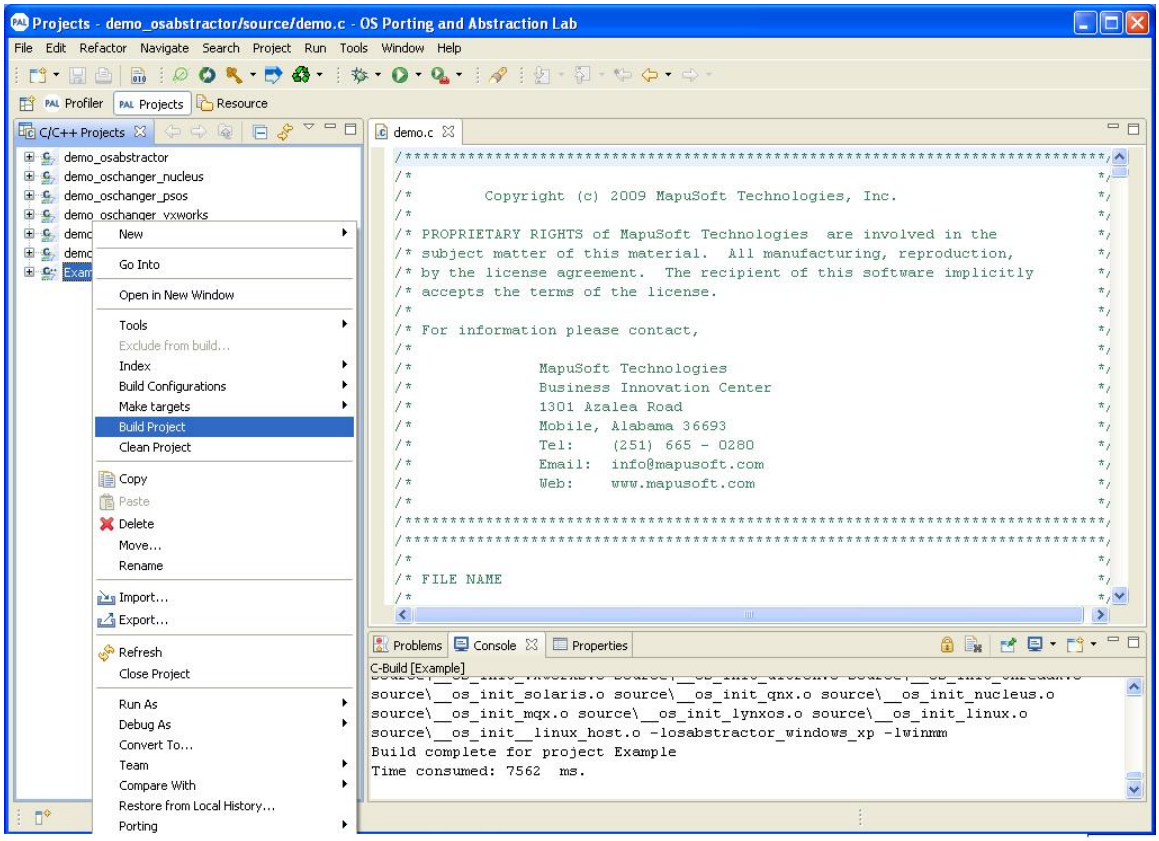


Building Binary Files for a Project

NOTE: This feature requires a license. Click http://mapusoft.com/downloads/ospal-evaluation/ to request an evaluation license.

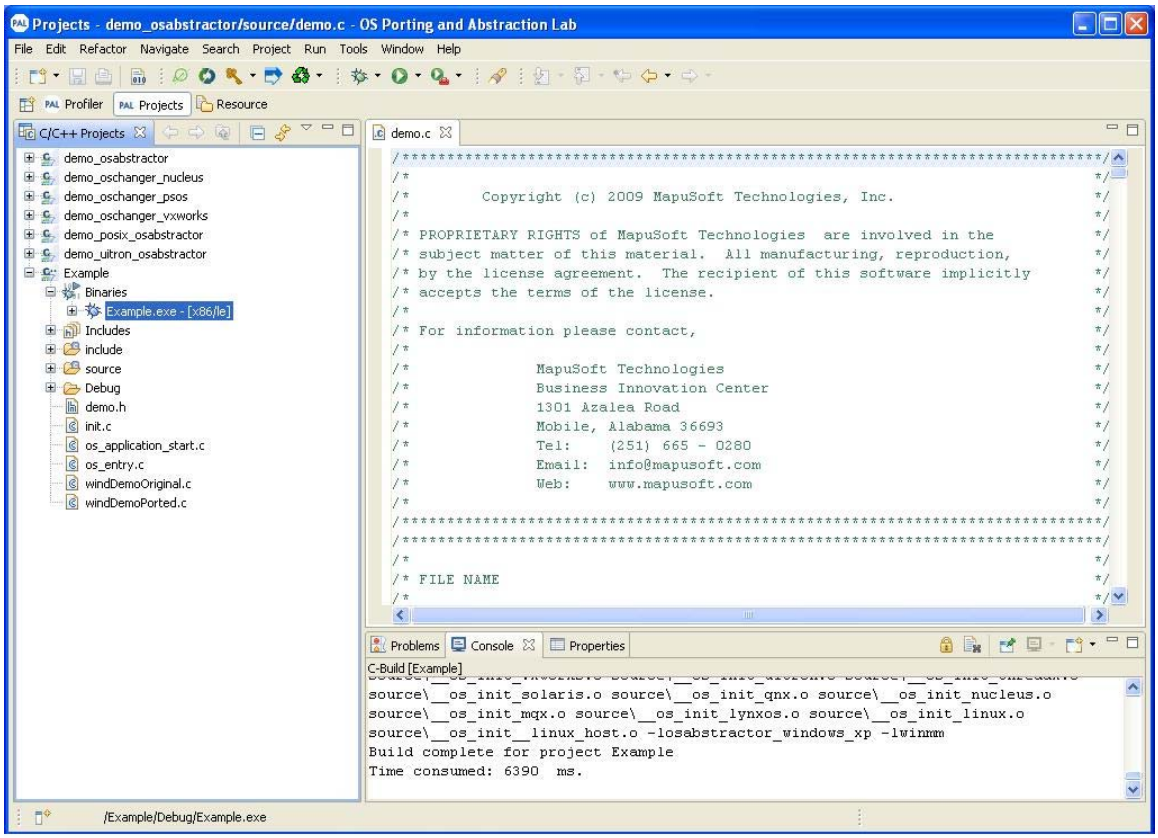
1. Create a project.
2. Import source code into OS PAL.
3. Select a project under C/C++ Projects pane, right click and select **Build Project** as shown in Figure 30.

Figure 30: Building Binary Files



4. You can view how the binary files are built in Figure 31.

Figure 31: Output for Building Binary Files for a Project

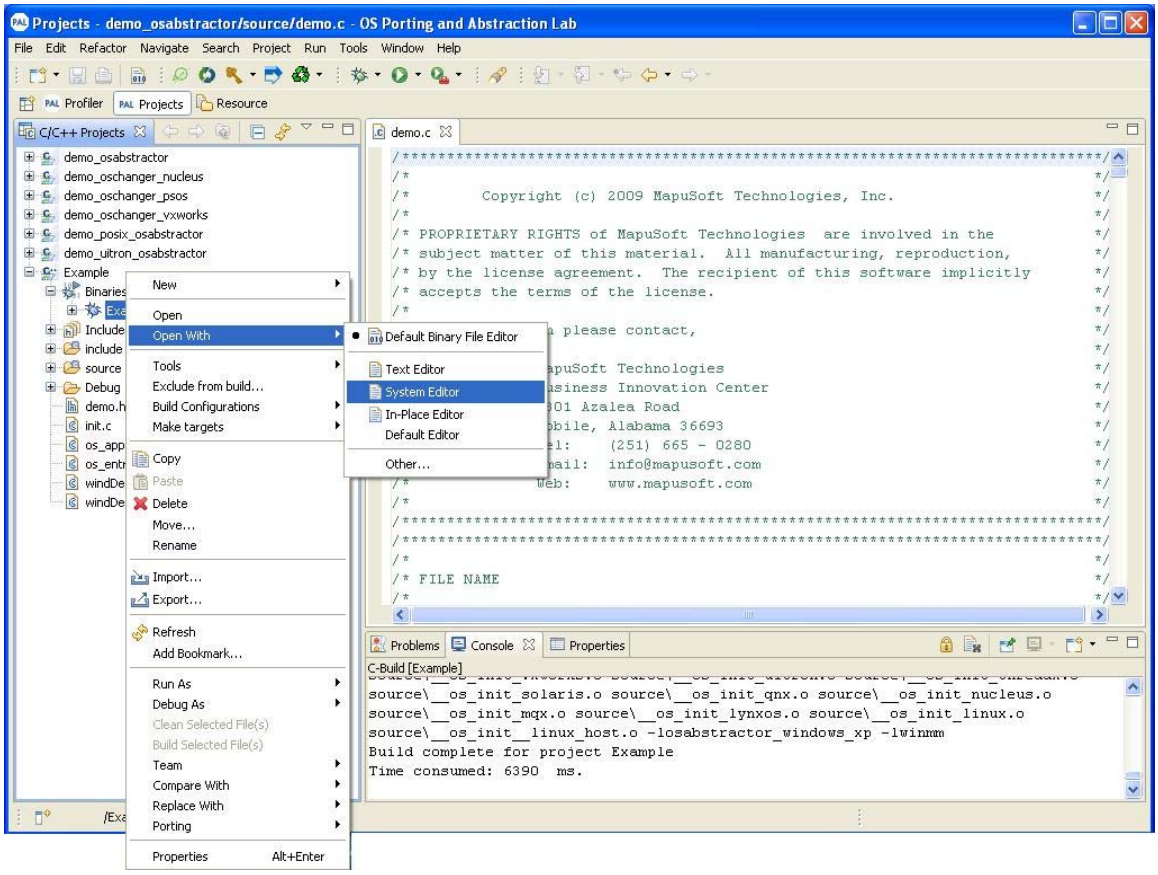


Executing Binary Files

To execute the binary in Windows host:

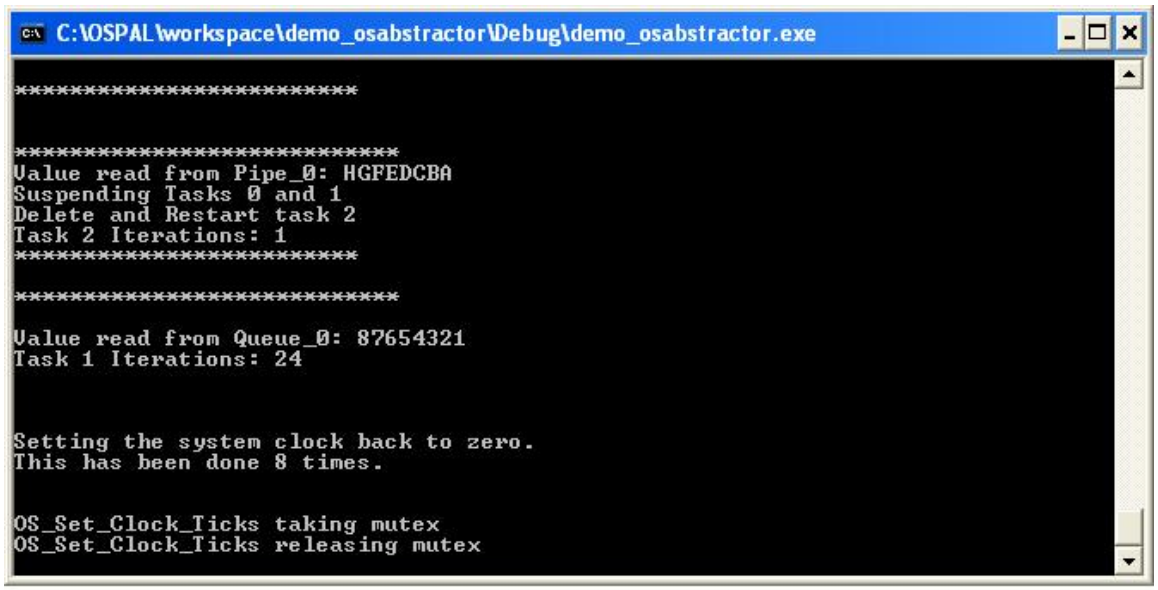
1. Select Project that you have created.
2. Select the Binary file, right click and select **Open With > System Editor** as shown in Figure 32.

Figure 32: Executing the Binary File



3. This approach always will fork a terminal to view the output as shown in Figure 33Figure 33: Binary Output

Figure 33: Binary Output



Updating Project Settings

OS PAL provides exclusive way to update the Projects Settings by just a click of a button. This is very useful in any one of the following cases:

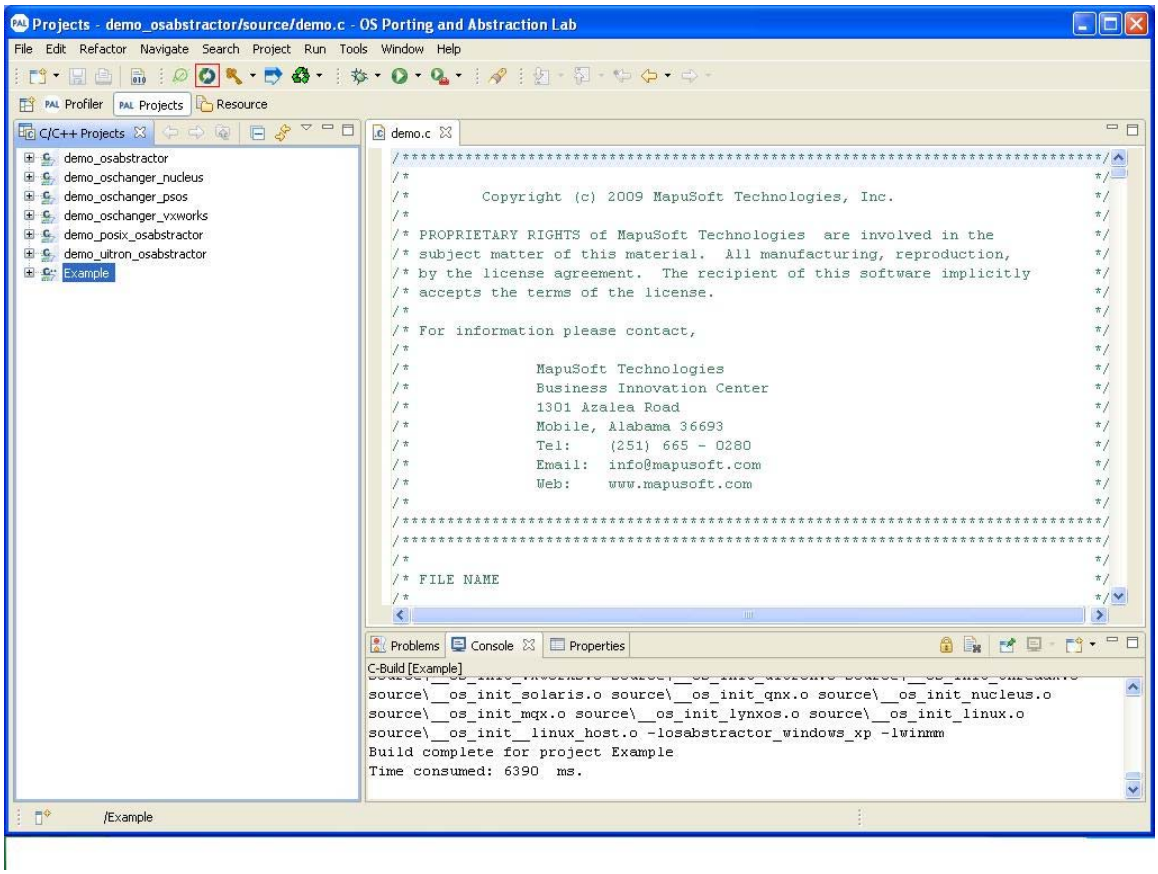
1. If the user has moved his workspace to a different location
2. If the project requires new toolchain that is installed recently

The **Update** button performs an auto update on all the projects updates which include files, new directory structures, libraries, and toolchains to the Project Settings.

To update project settings:

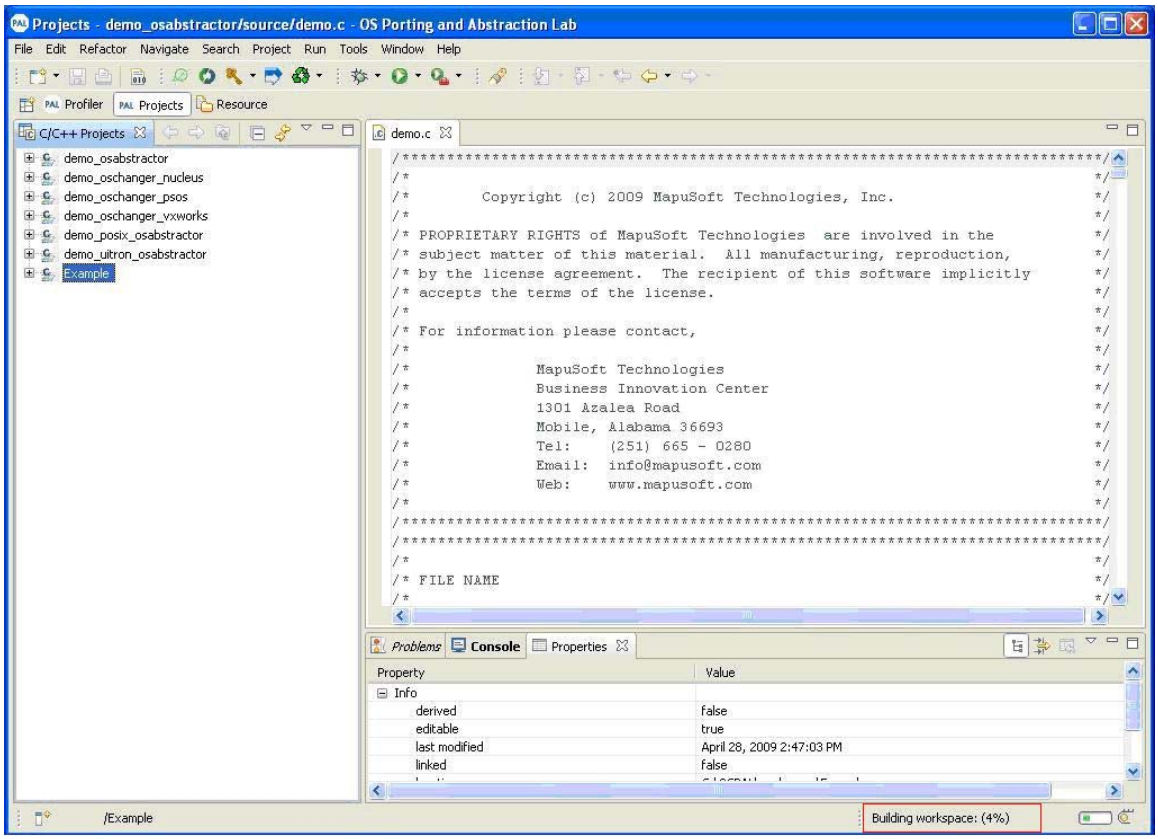
1. From OS PAL main menu select **Tools** > **Update Settings** or click **Update** Project Settings button on OS PAL main menu or, as shown in the Figure 34.

Figure 34: Updating Project Settings



2. OS PAL does an auto search for the project updates and updates the settings as shown in the highlighted area on the status bar in Figure 35.

Figure 35: OS PAL Project Updates



USING THE OS PAL PROFILER

About OS PAL Profiler

NOTE: This feature requires a license. Click http://mapusoft.com/downloads/ospal-evaluation/ to request an evaluation license.

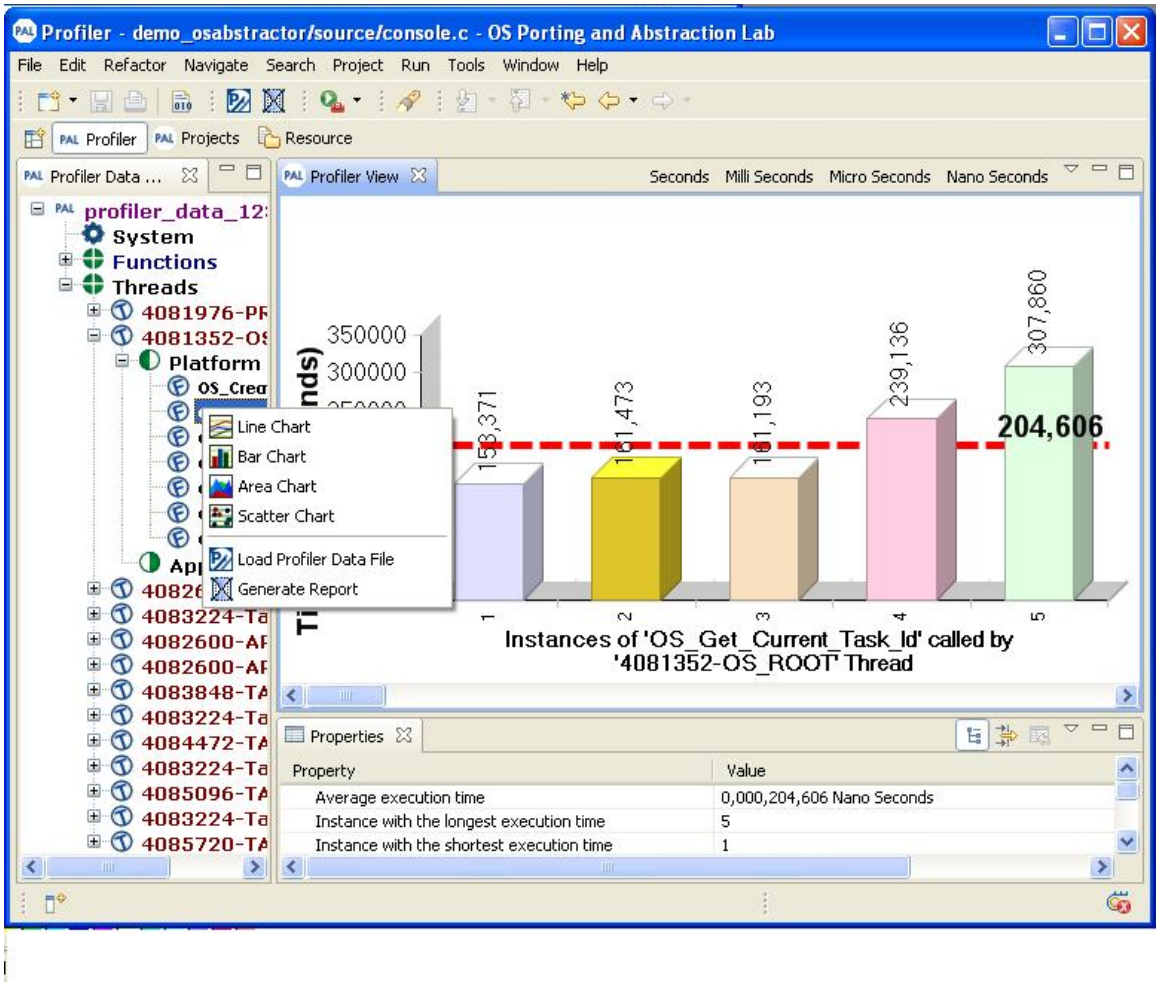
The OS PAL Profiler is an add-on to the established OS PAL Eclipse based code migration and API optimization technology and is designed to enable data collection.

OS PAL Profiler offers the following:

- The data collected by the Profiler provides feedback concerning the utilization of MapuSoft's APIs in the project.
- The reports allow for performance impact analysis by detailing specific API execution time during a particular time period as well as the average and total API execution times.
- OS PAL enables user to collect data pertaining to the Mapusoft API's (Platform API profiling) and profiling user specific functions (Application Profiling).
- Users can analyze the data with the included OS PAL Profiler graphical viewer which offers area, bar, line, pie, and scatter charts, as shown in Figure 36.

NOTE: In the current release, Profiler feature is not supported in ThreadX and Nucleus targets.

Figure 36: OS PAL Profiler



Components on the Profiler Window

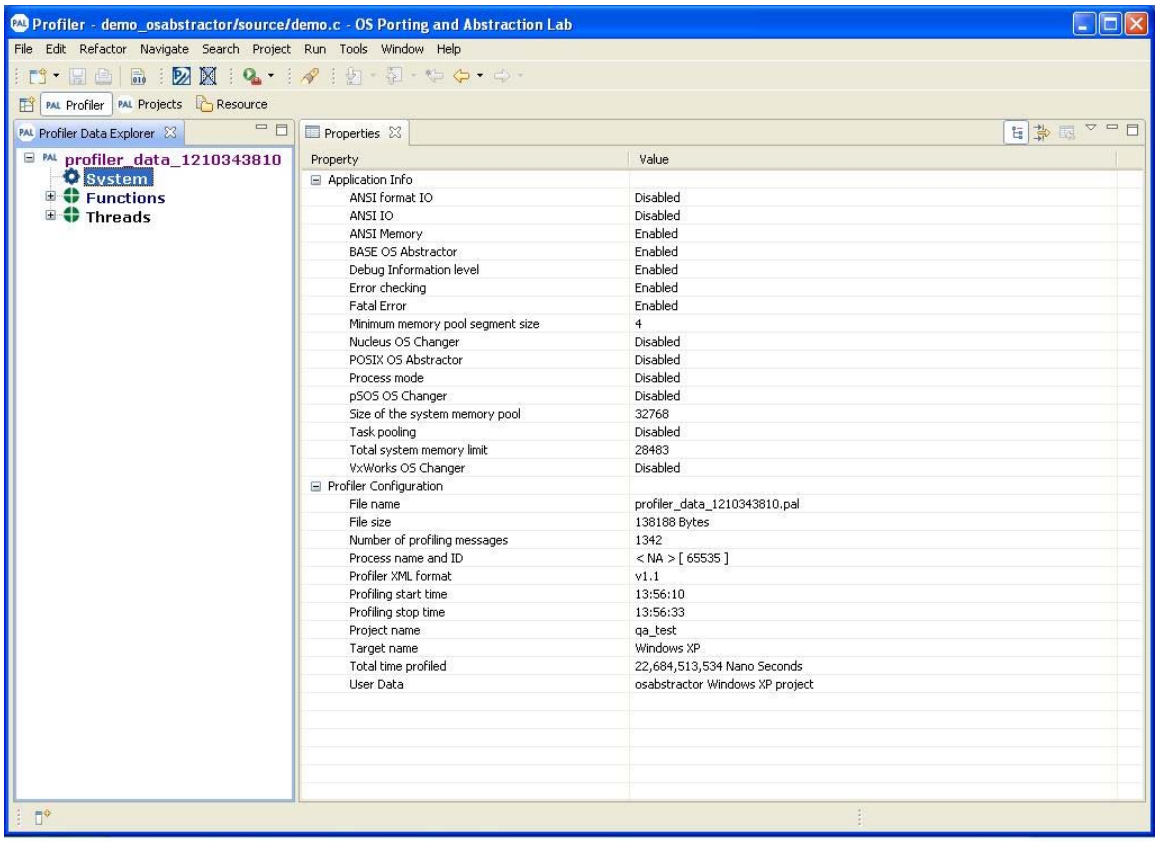
OS PAL Profiler window contains two panes. The left pane has three Profiler components listed and on the right pane, you can view the respective details and information in a graphical view.

The three main components of OS PAL Profiler are:

Profiler_data file–This is the generated profiler data file. For more information, refer to Generating Timing Report section.

1. **System**–This displays the system details of your application as shown in Figure 37.

Figure 37: OS PAL Profiler- System Details

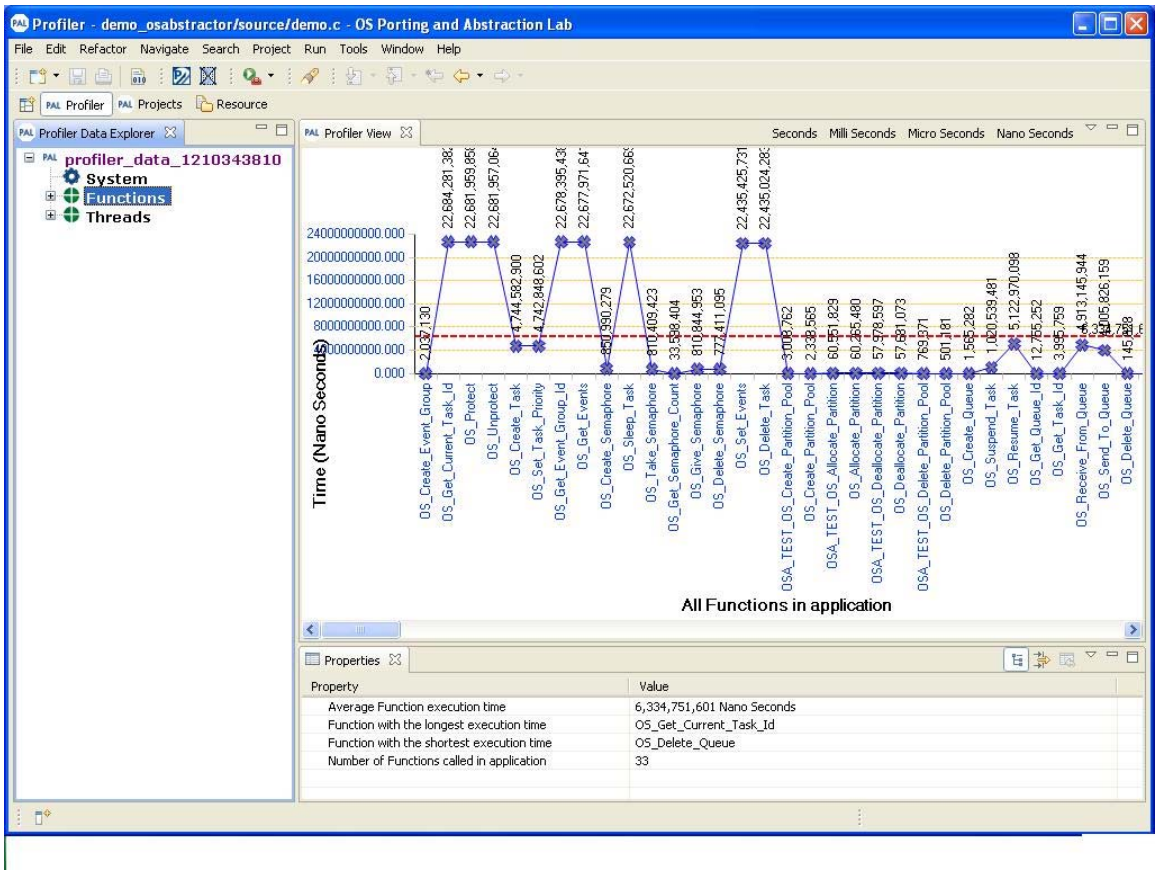


If you expand **System** tab you have the following details which are displayed on the right pane as shown in the image:

- Application info–Application property values
- Profiler Configuration–Profiling Application values

2. **Functions**–This displays all the functions called in the application and the time taken to execute these functions as shown in Figure 38.

Figure 38: OS PAL Profiler - Functions

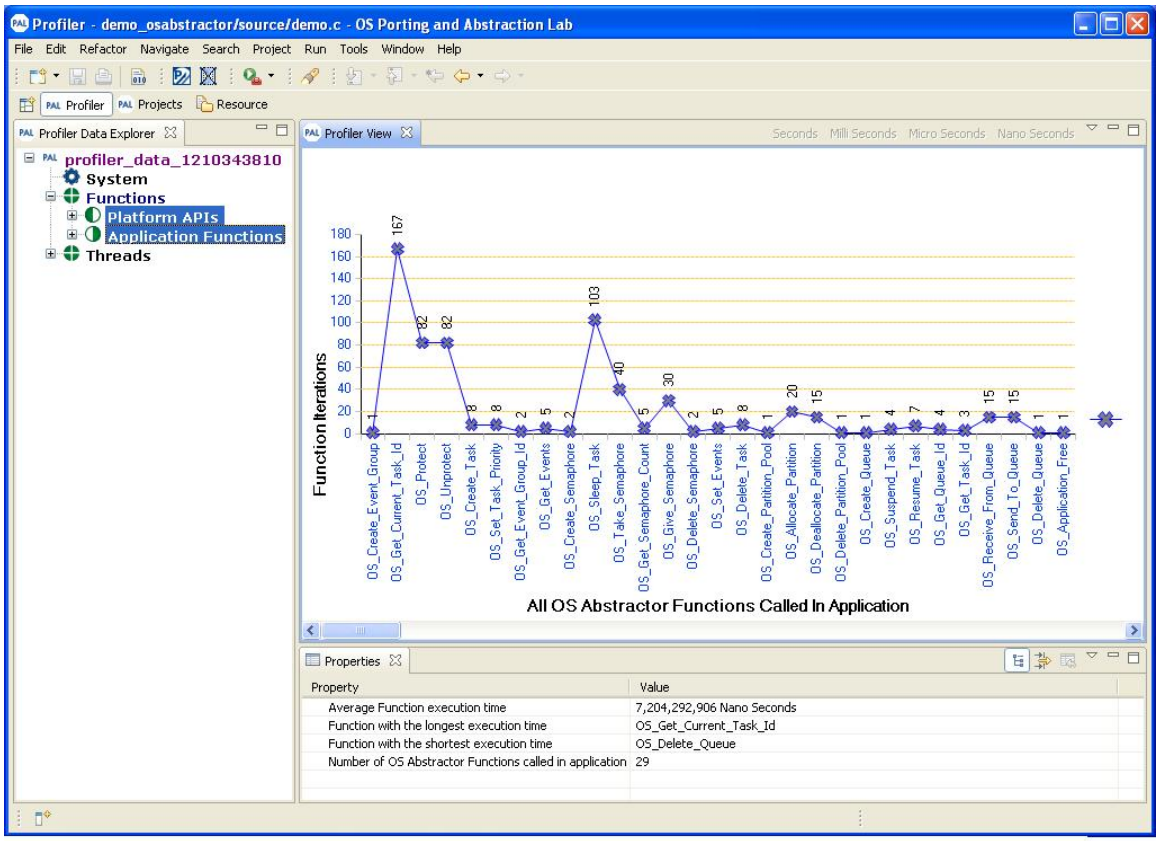


On the bottom of the window the function properties are displayed such as:

- Average Function execution time
- Function with the longest execution time
- Function with the shortest execution time
- Number of Functions called in application

From the left pane, expand the **Functions** tab. It displays the following information as shown in Figure 41:

Figure 39: Platform APIs and Application Functions



Platform APIs–These are all the OS Abstractor functions called in the application. On the x-axis, all the functions are displayed. On the y-axis, all functions iterations are displayed. On the bottom of the window the function properties are displayed such as:

- Average Function execution time
- Function with the longest execution time
- Function with the shortest execution time
- Number of OS Abstractor Functions called in application

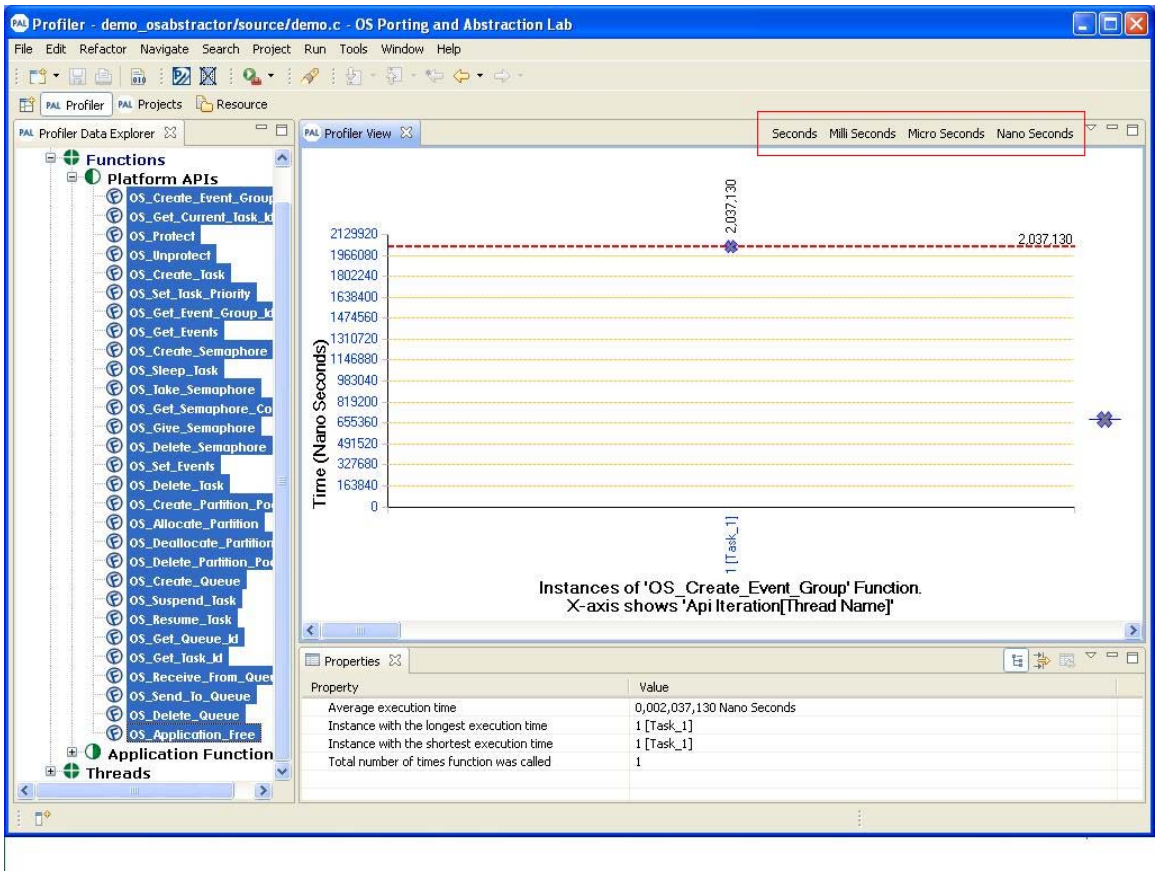
If you expand the Platform APIs, you can view all the platform APIs called in the application as shown in Figure 40. On the bottom of the window, the function properties are displayed such as:

- Average execution time
- Instance with the longest execution time The *Task _1* in square brackets denote that these function properties belong to the Task 1 Thread.
- Instance with the shortest execution time
- Total number of times function was called

If you click on a Platform API, you can view the number of instances of the specific function on the x-axis and the time taken for each API on the y-axis.

NOTE: On top of the profiler view, you can view different measures of time such as seconds, milli seconds, micro seconds, and nano seconds as highlighted in the Figure 40. This is used to capture the time taken for each instance of the function in different time measures. If you click on nano seconds, the time graph will be shown as Time (nano seconds) as shown in the figure.

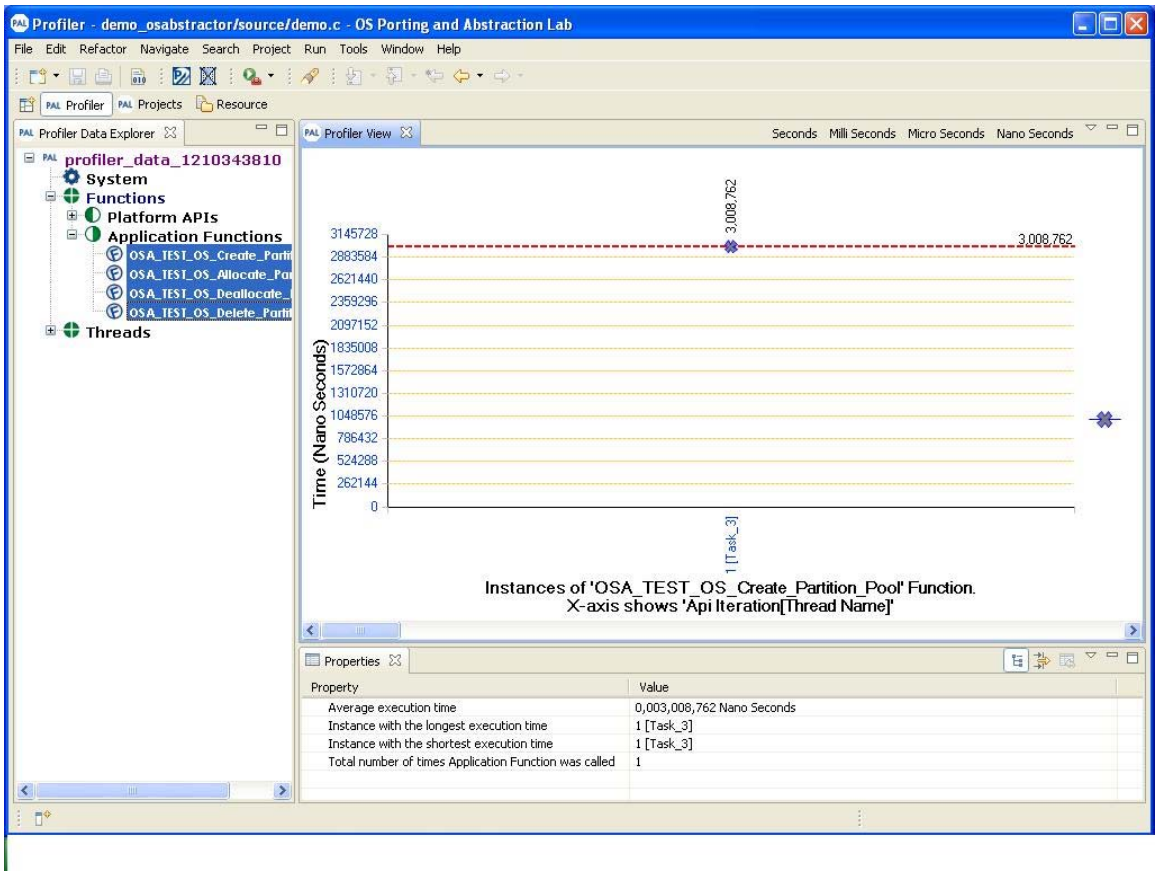
Figure 40: Platform APIs



Application Functions–These are all the user specific functions called in the application. On the x-axis, all the user specific functions are displayed. On the y-axis, all functions iterations are displayed. On the bottom of the window the function properties are displayed as shown in Figure 41 such as:

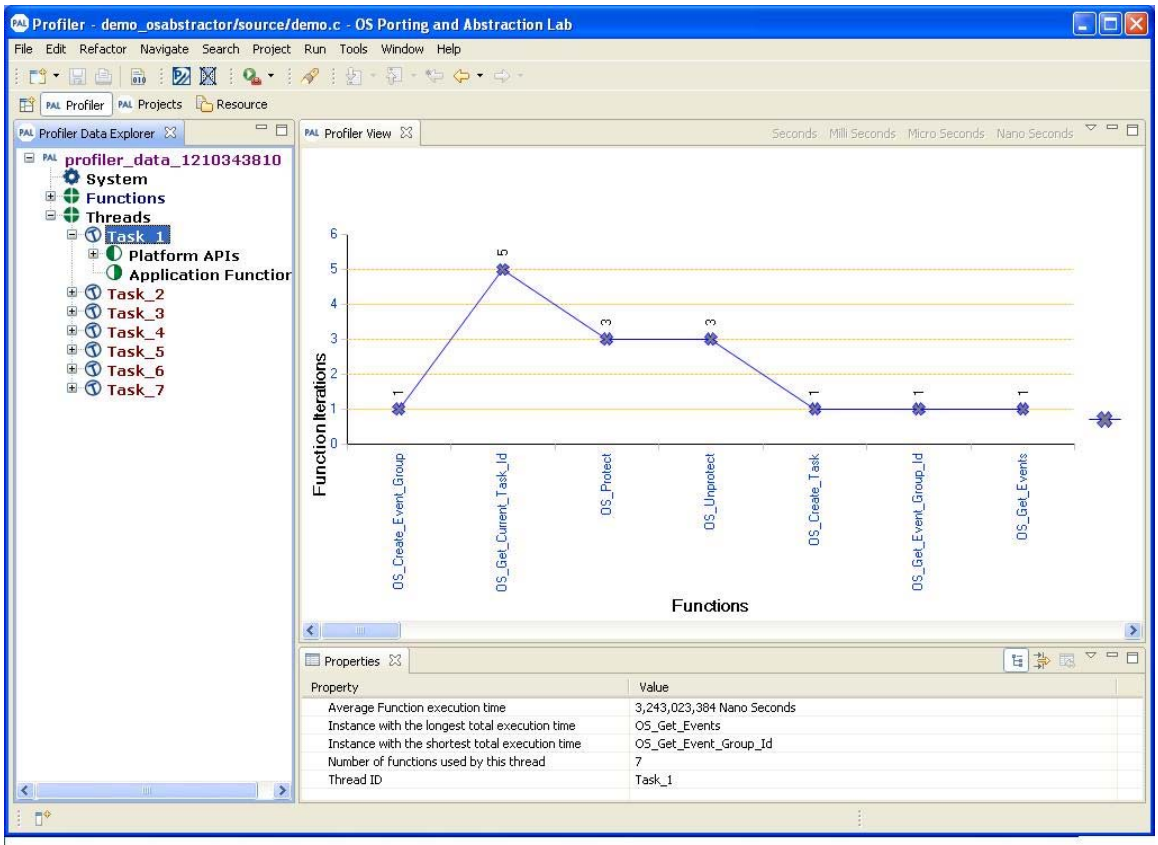
- Average Function execution time
- Function with the longest execution time
- Function with the shortest execution time
- Number of times Application Functions are called in application

Figure 41: Application Functions



3. **Threads**–Threads are created to execute any function in an application. IN OS PAL Profiler you can view the Thread properties by expanding the Thread tab as shown in Figure 42. On the bottom of the window the thread properties are displayed as shown in Figure 42 such as:
    - Average Function execution time
    - Instance with the longest total execution time
    - Instance with the shortest total execution time
    - Number of Functions used by this thread
    - Thread ID

Figure 42: OS PAL Profiler - Threads



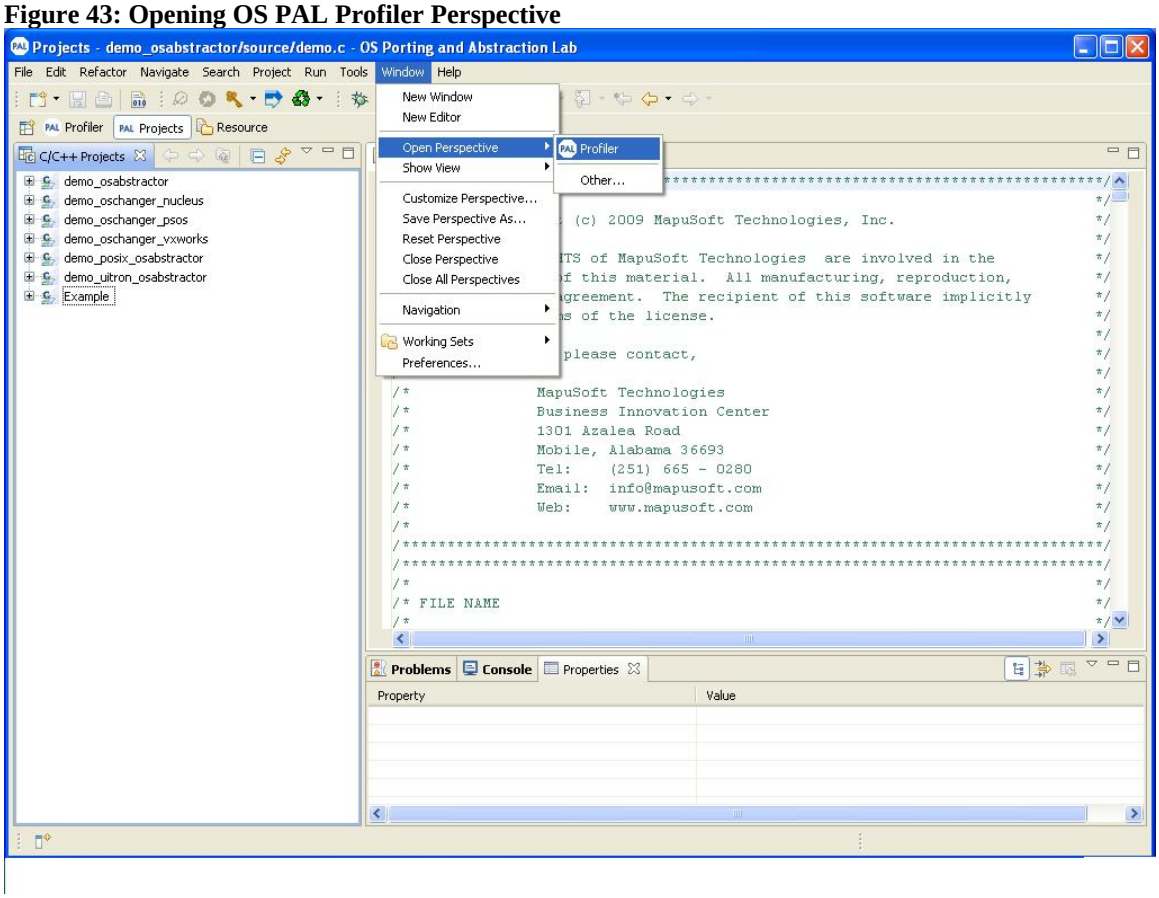
- Tasks–These are functions called for each task. If you expand the Task tab, you have the following as already discussed under the Functions tab:
    - Platform APIs
    - Application Functions

Opening OS PAL Profiler Perspective

From OS PAL: main menu, click **OS PAL Profiler** perspective button
Or,

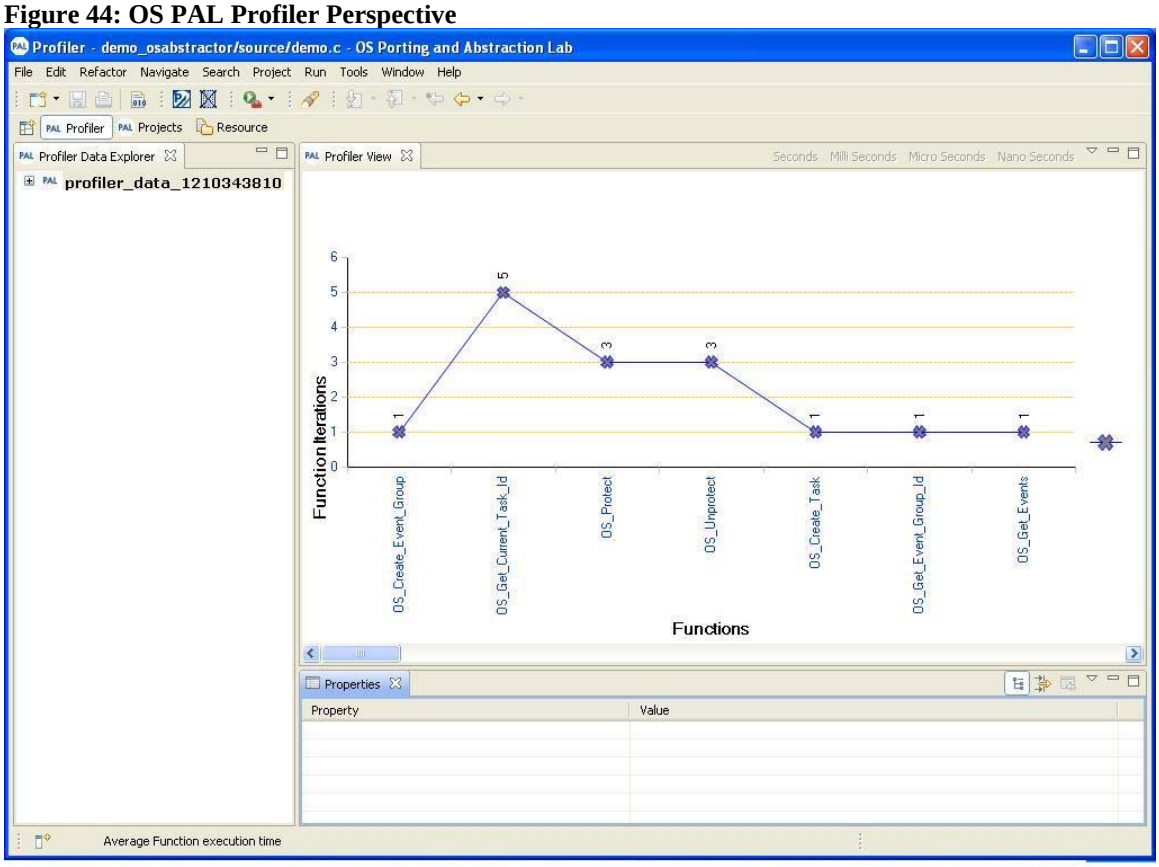
1. On OS PAL main menu, select **Window > Open Perspective > Profiler** as shown in the Figure 43.

Figure 43: Opening OS PAL Profiler Perspective



2. You can view the OS PAL Profiler Perspective as shown in Figure 44.

Figure 44: OS PAL Profiler Perspective

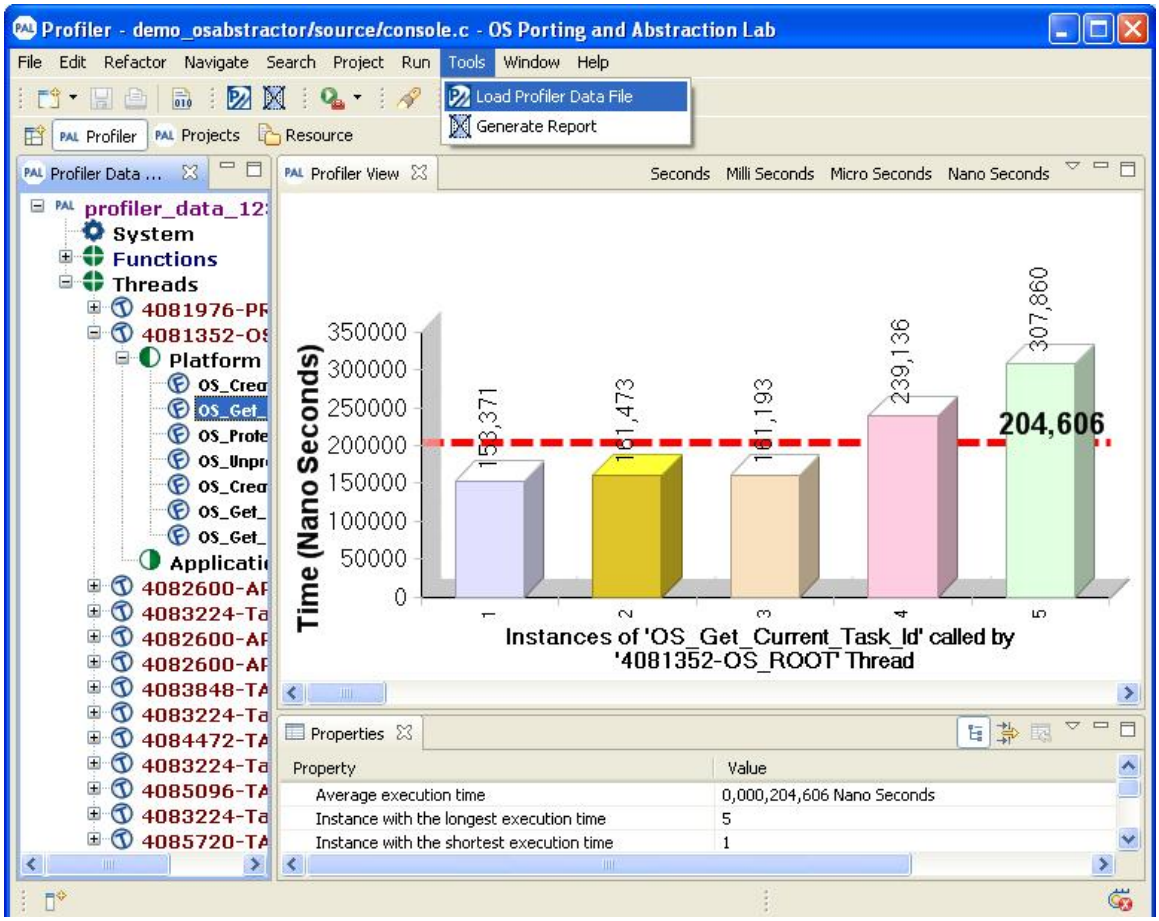


Viewing OS PAL Profiler Data

NOTE: This feature requires a license. Click http://mapusoft.com/downloads/ospal-evaluation/ to request an evaluation license.

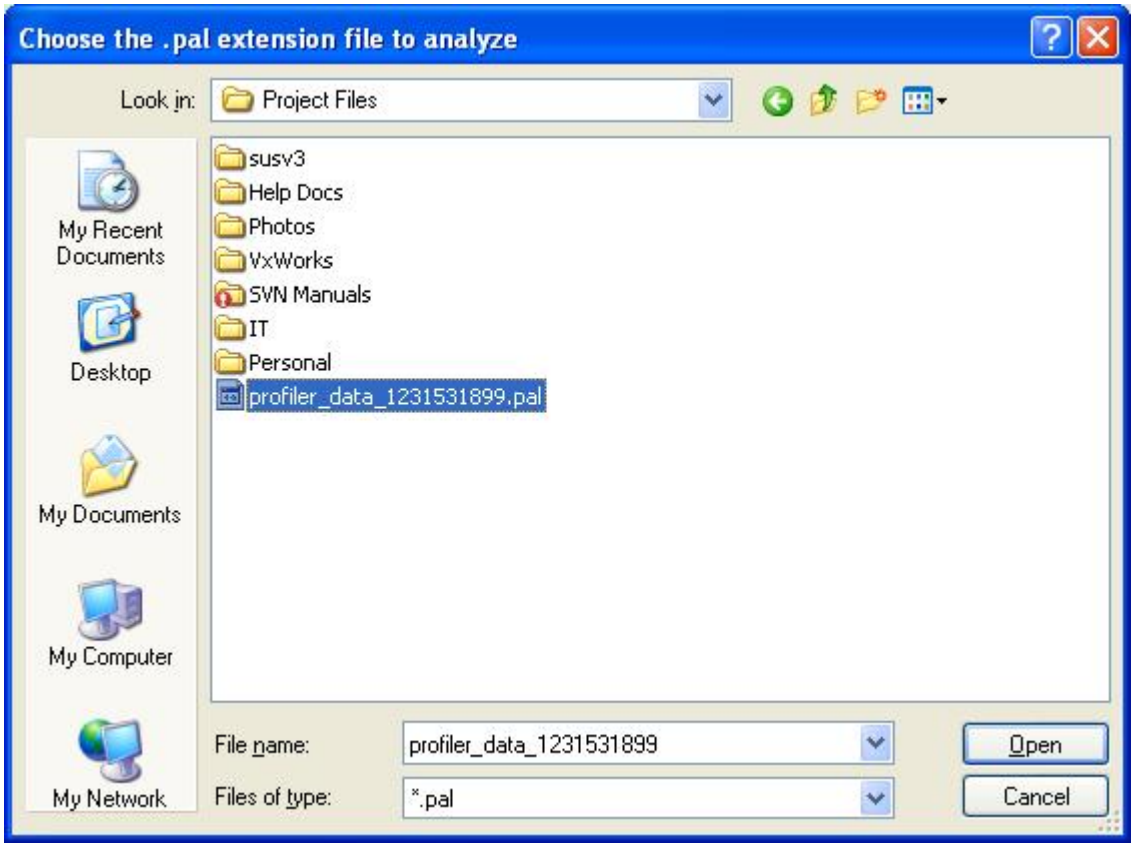
1. Open the OS PAL Profiler perspective.
2. From the OS PAL main menu, select **Tools > Load Profiler Data File** as shown in Figure 45.

Figure 45: Viewing OS PAL Profiler Data



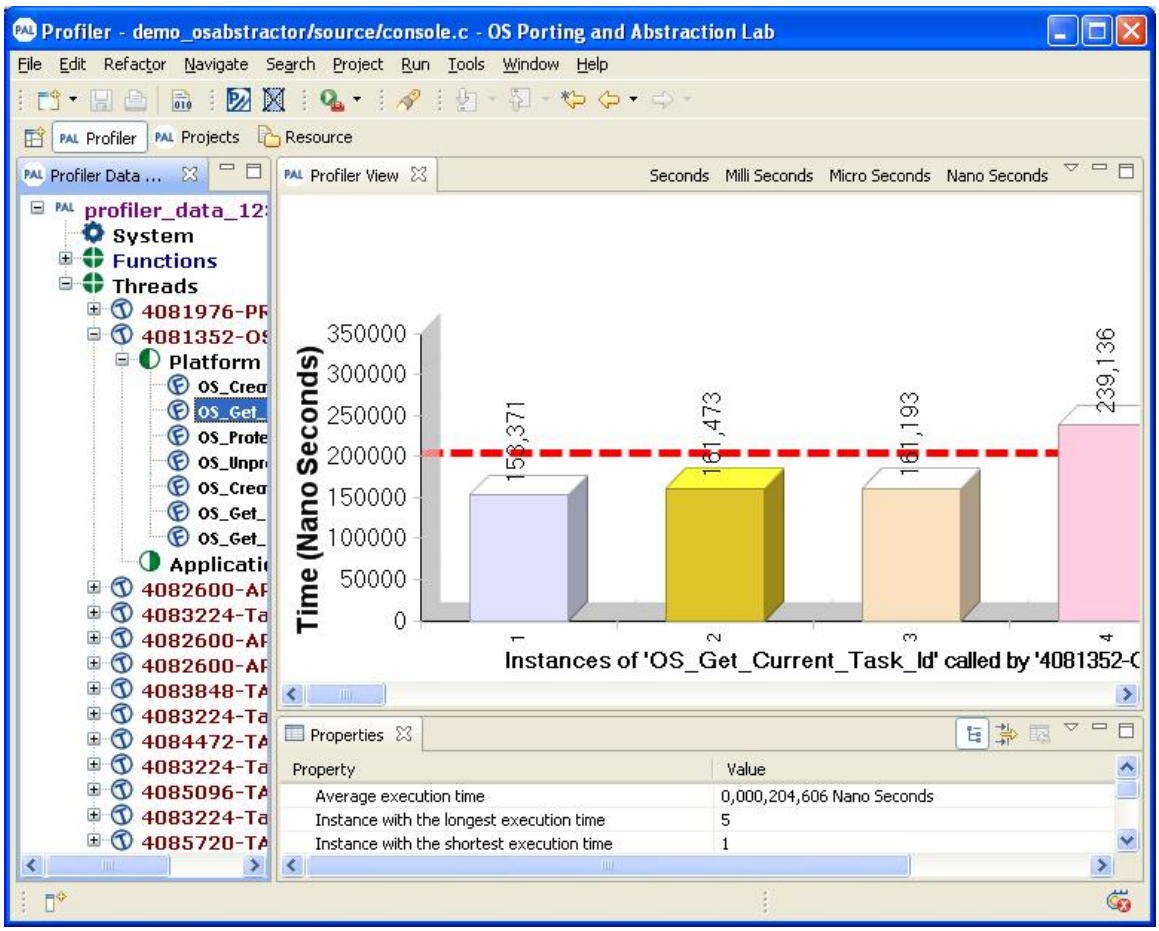
3. Browse to your saved profiler data file, and click **Open** as shown in Figure 46.

Figure 46: Selecting the .pal Extension File to Analyze



4. Select an API to view the data and right click on **Profiler Data Explorer** tab to view the different graph options as shown in Figure 47.
   NOTE: You can select an appropriate graphical viewer to view your profiler data. You can view the profiler data in a line chart, bar chart, area chart, or a scatter chart.

Figure 47: Selecting the API to view the Profiler Data



Generating OS PAL Profiler Data for your Target

NOTE: This feature requires a license. Click http://mapusoft.com/downloads/ospal-evaluation/ to request an evaluation license.

OS PAL now provides two kinds of profiling features.

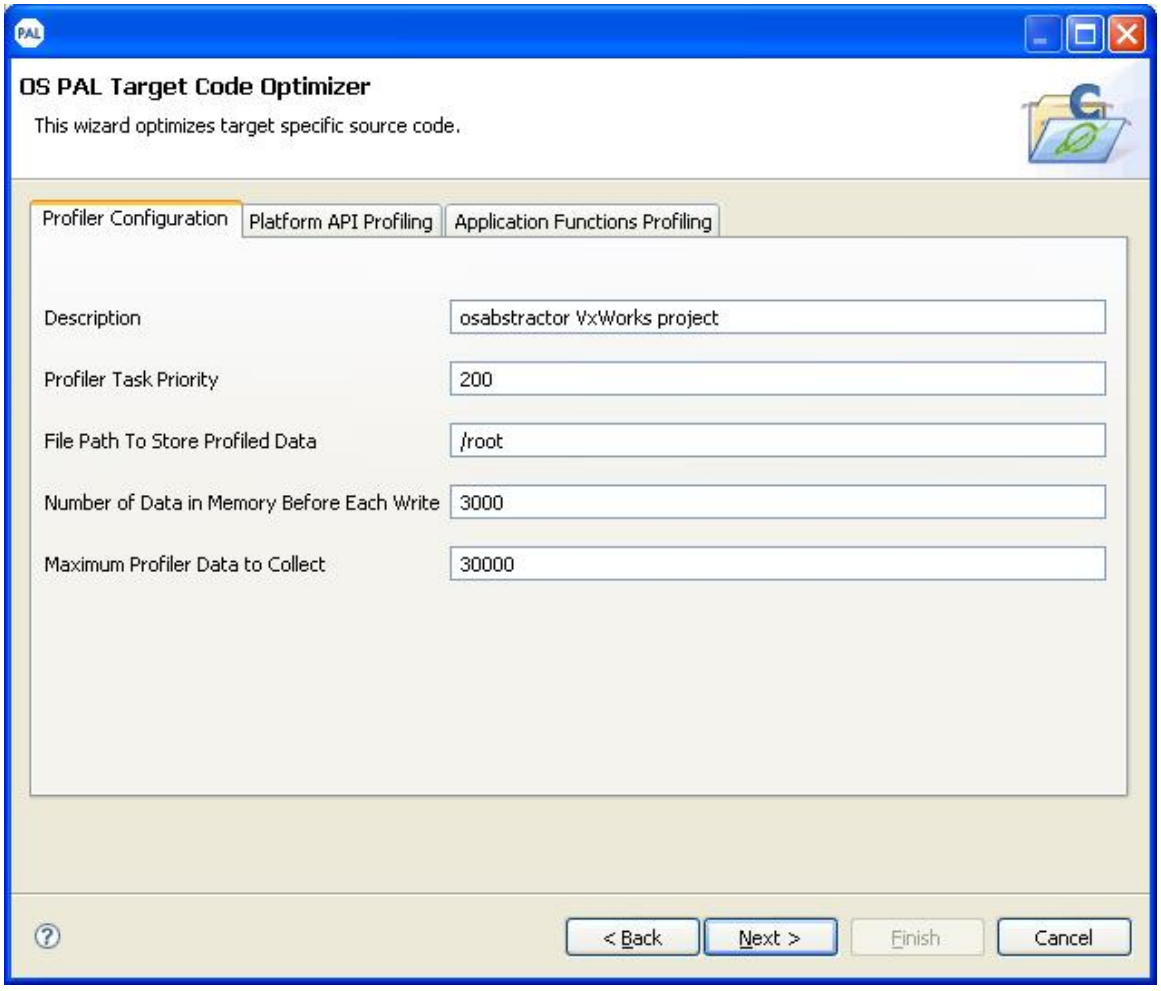
- Platform API Profiling–System specific API profiling
- Application Profiling–User specific API profiling

NOTE: For more information on Profiler Configuration, refer to Generating Target Code in this manual.

To generate OS PAL Profiler data by using Platform API Profiling:

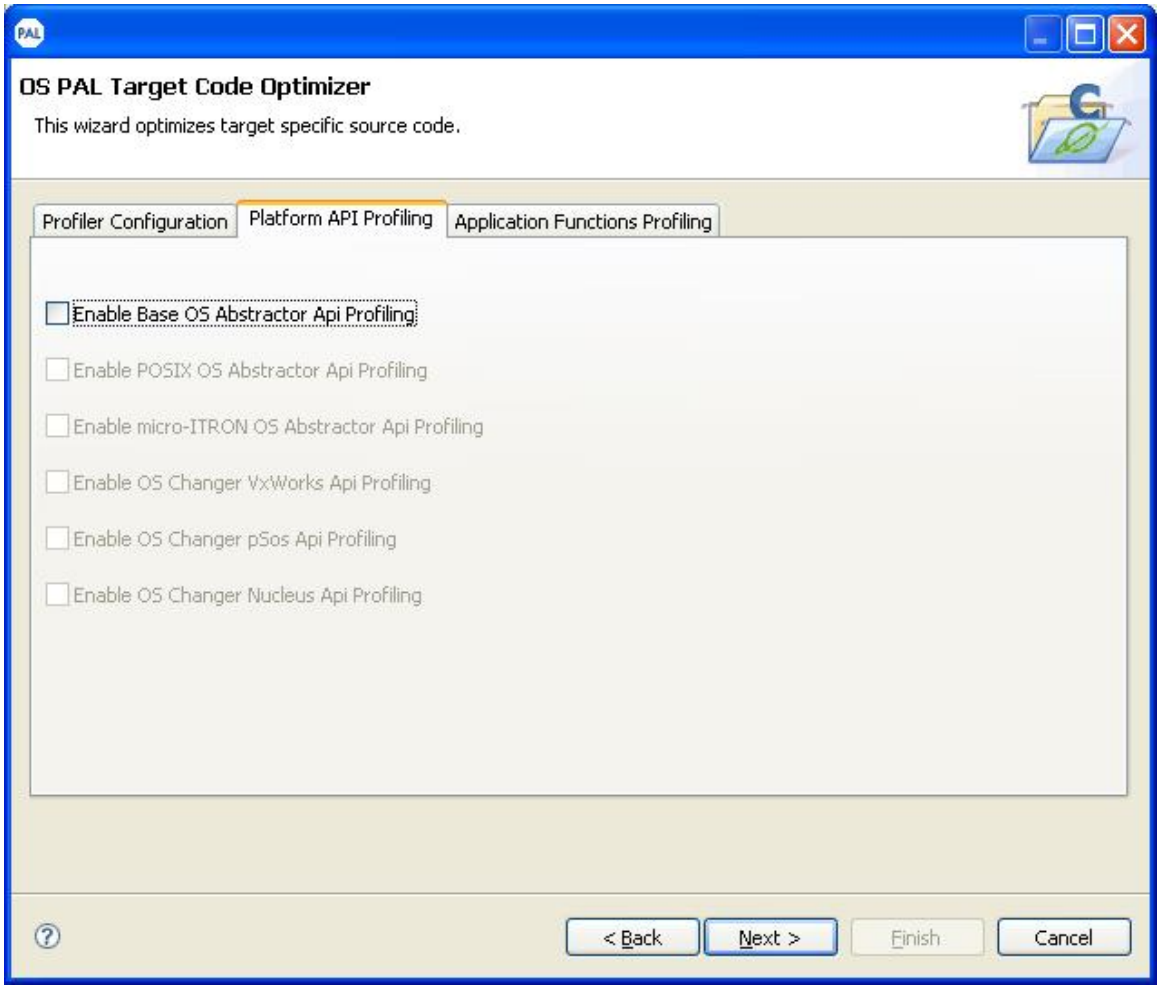
1. On OS PAL Target Code Optimizer window, select the **Profiler Configurations** tab and define the profiler data specifications as shown in Figure 48.

Figure 48: Profiler Configuration



2. Click **Platform API Profiling** tab and select the check box next to the appropriate Changer product to enable Platform API Profiling to generate system specific APIs profiler data as shown in Figure 49.

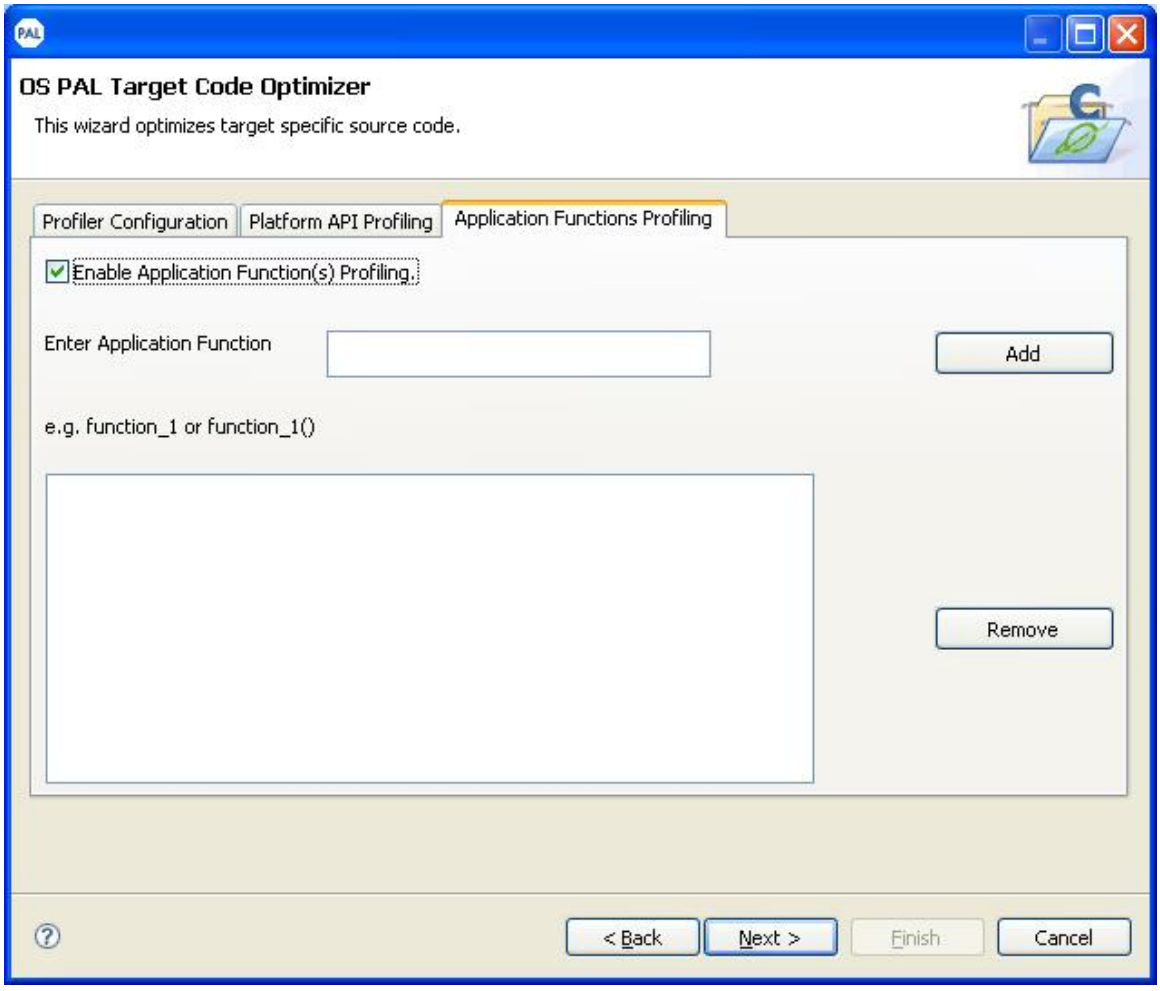
Figure 49: Platform API Profiling



To generate OS PAL Profiler data by using Application Function Profiling:

1. ON OS PAL Target Code Optimizer window, select the **Profiler Configurations** tab and define the profiler data specifications as shown in Figure 48.
2. Click **Application Profiling** tab and select the check box next to Enable Application Profiling to generate user specific APIs profiler data as shown in Figure 50.

Figure 50: Application Profiling



NOTE: C++ overloaded functions are not supported in Application Functions profiling.

Generating OS PAL Timing Report

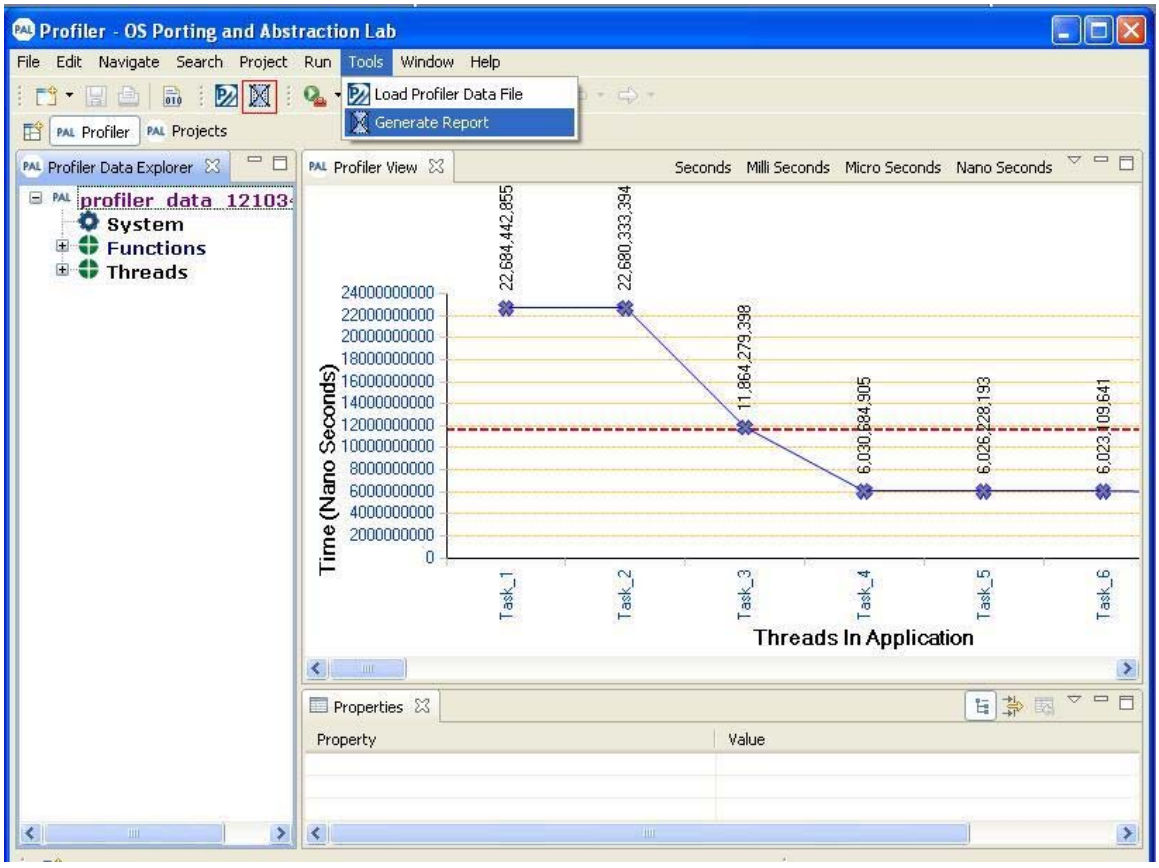
NOTE: This feature requires a license. Click http://mapusoft.com/downloads/ospal-evaluation/ to request an evaluation license.

OS Pal now provides you a new feature to view the performance report for each APIs.

To generate Timing Report:

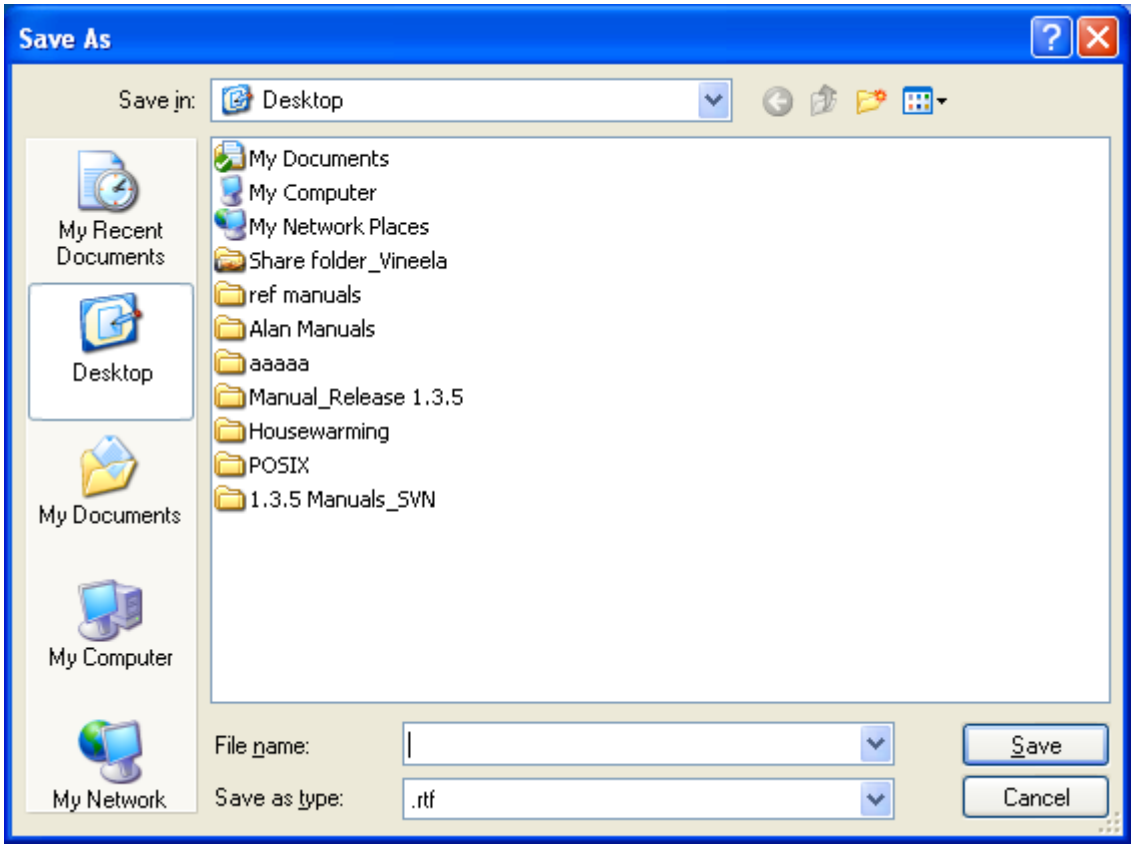
1. From the OS PAL main menu, go to Profiler perspective and select any Profiler Data on your left pane to generate the report.
2. Select **Tools > Generate Report**. You can also click on Generate Report button on the OS PAL window as shown in Figure 51.

Figure 51: Generate Timing Report



3. A Save As window is displayed. Select the directory where you want to save the report and enter a file name for the report and click **Save** as shown in Figure 52.

Figure 52: Saving the Timing Report



4. Your Timing Report is successfully generated as an .rtf file.

The Timing Report displays the following information:

1. **Timing Information—**The timing information gives a detailed description of the following:

   - Best Time Value – Specifies the minimum time taken to perform the action on each platform API
   - Worst Time Value – Specifies the maximum time taken to perform the action on each platform API
   - Average Time Value – Specifies the average time taken to perform the action on each platform API

2. **Application information—**When you perform the application profiling on OS PAL, the report displays the following application property values:
    - Total system memory limit– Specifies the total system memory pool limit of the application.
    - Size of the system memory pool– Specifies the size of the system memory pool of the application.
    - Minimum memory pool segment size– Specifies the minimum size of the memory pool segment of the application.
    - VxWorks OS Changer–Specifies if you have enabled VxWorks OS Changer.
    - pSOS OS Changer– Specifies if you have enabled pSOS OS Changer.
    - POSIX OS Abstractor– Specifies if you have enabled POSIX OS Changer.
    - BASE OS Abstractor– Specifies if you have enabled the BASE OS Abstractor.
    - Process mode– Specifies if the OS Abstractor process feature is enabled or disabled.
    - Task pooling– Specifies if the Task pooling feature is enabled for this application.
    - ANSI Memory– Specifies if you want to map ANSI malloc() and free() to OS Abstractor equivalent functions.
    - ANSI format IO– Specifies if you want to map ANSI printf() and sprintf() to OS Abstractor equivalent functions.
    - Debug Information level– Specifies if you want to enable the debug output.
    - Error checking– Specifies if you want to enable the error checking.
    - Fatal Error– Specifies if you want to enable the feature to ignore fatal errors.

3. **Profiler Configuration—**When you perform profiling on OS PAL APIs, the report displays the following profiling application values:

    - File name–Specifies the name of the .pal file generated by OS Abstractor.
    - Project name–Specifies the name of your project.
    - Target name–Specifies the target OS you have selected for profiling.
    - File size–Specifies the size of the file to be profiled.
    - Profiler XML format–Specifies the version of the XML used for profiling.
    - Process name and ID–Specifies the process name and the ID.
    - User Data–Specifies the information provided by the user.
    - Profiling start time–Specifies the starting time of profiling.
    - Profiling stop time– Specifies the end time of profiling.
    - Total time profiled– Specifies the total time taken for profiling.
    - Number of profiling messages– Specifies the number of profiler messages.

DEVELOPING TARGET CODE WITH OS PAL

Generating Target Code

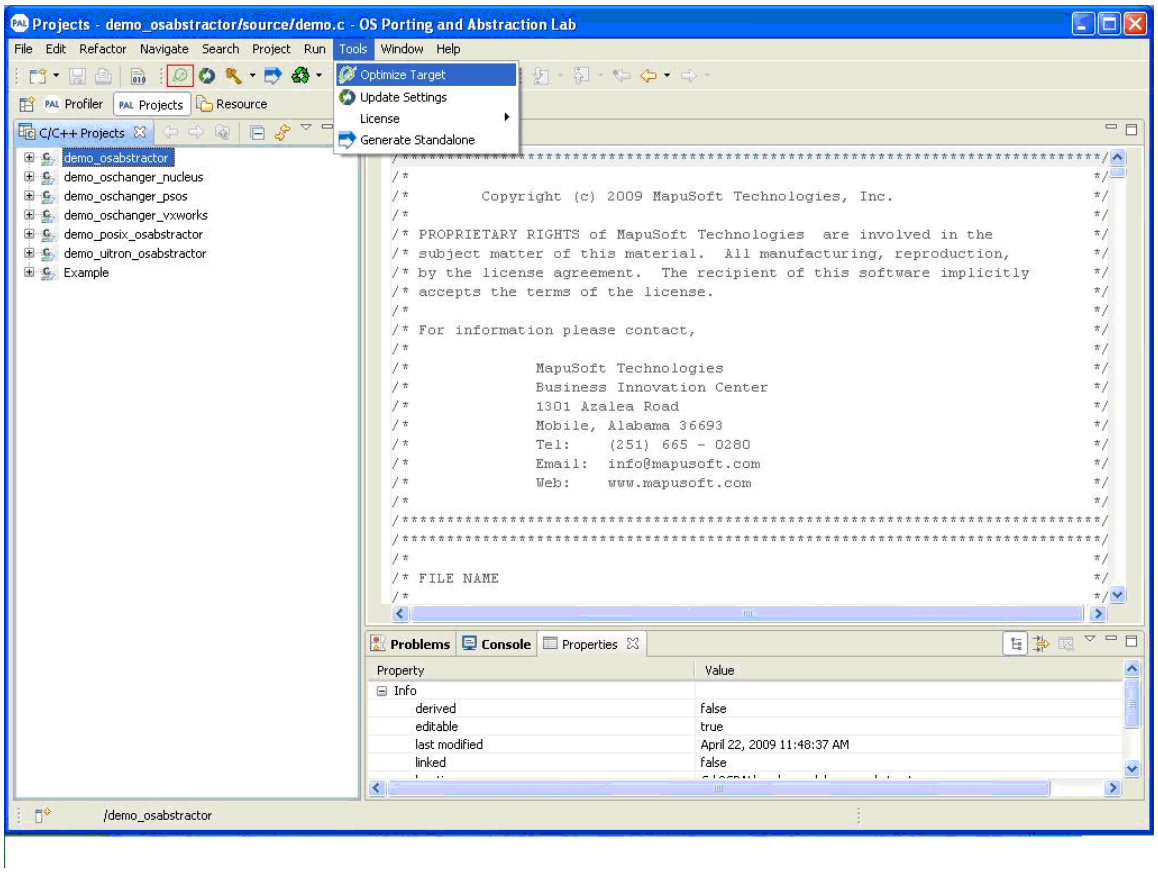
NOTE: This feature requires a target license. Click http://mapusoft.com/contact/ to send a request to receive licenses and documentation.

Most of the configurations described below can also be changed at run time using the *OS_Application_Init function.*

To generate Target Code:

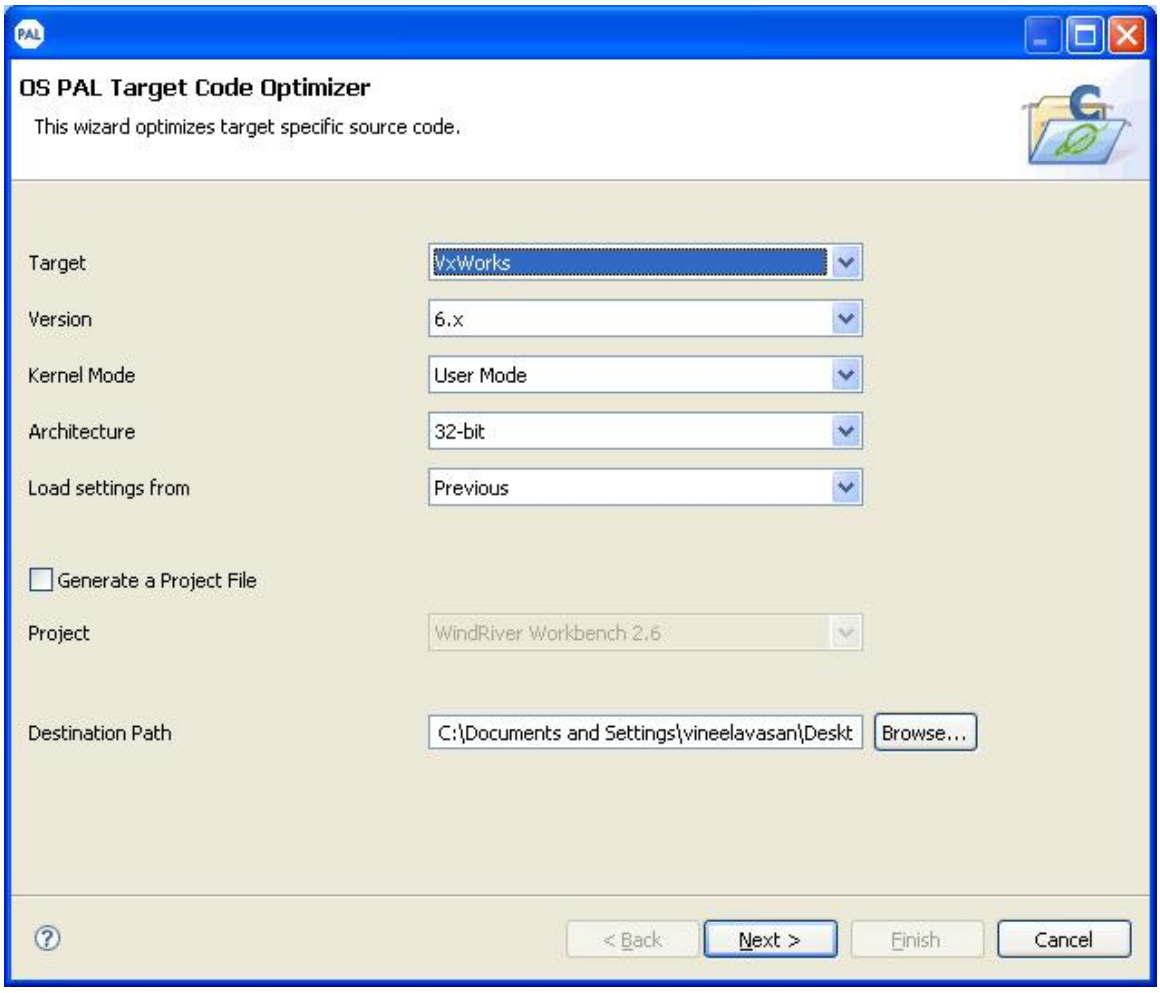
1. From OS PAL projects, select a project.
2. From OS PAL main menu, click **Tools > Optimize Target** or click on the OS PAL Target Code Optimizer button as shown in Figure 53.

Figure 53: OS Pal Target Code Optimizer



3. From OS PAL Code Optimizer window, select your target platform specifications from the drop down list. VxWorks operating system is selected as an example as shown in Figure 54.

Figure 54: Selected VxWorks Target in this Example

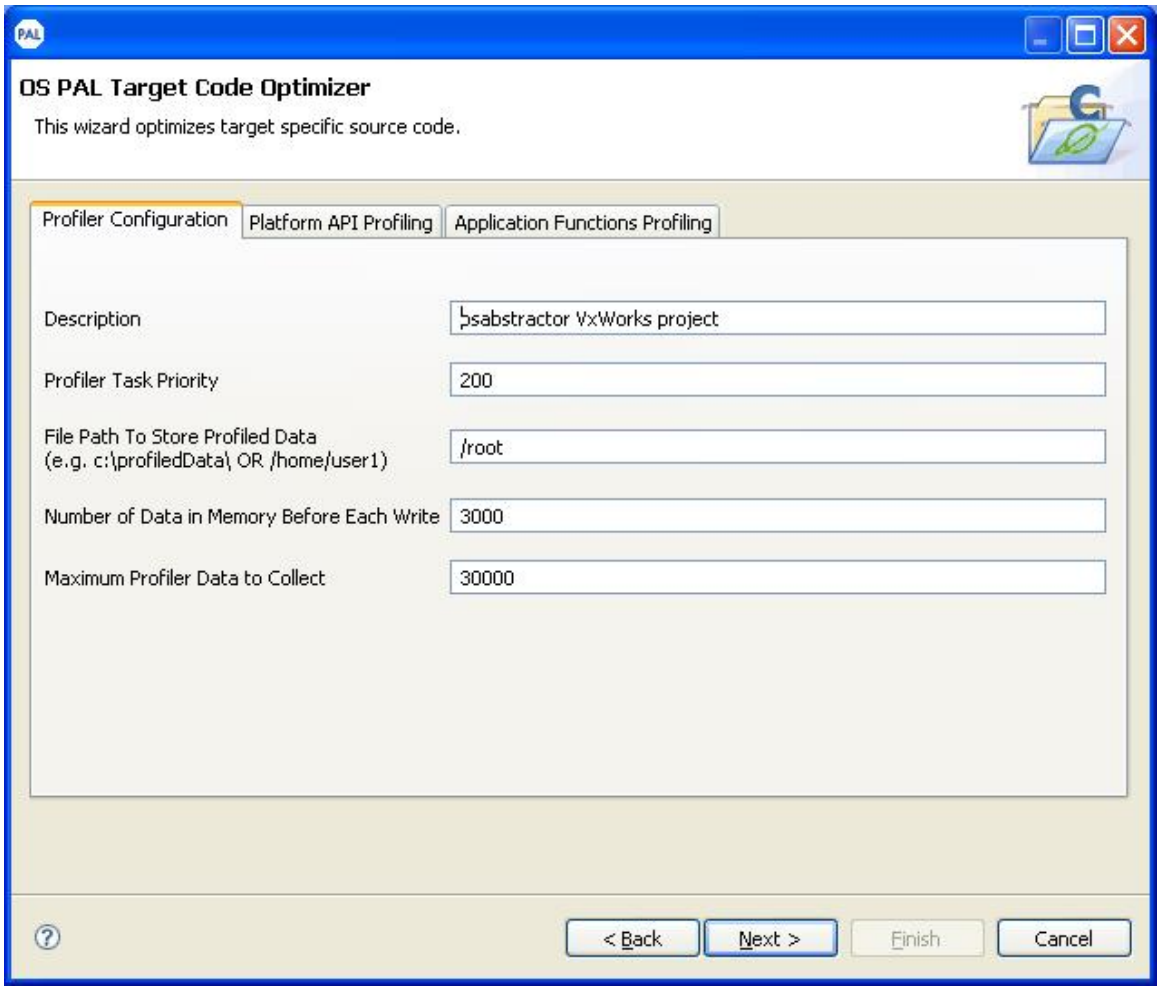


The field descriptions on OS PAL Target Code Optimizer window are as follows:

Field	Description	Your Action
Target	Specifies the target OS name.	Enter the target OS name in the text box.
Version	Based on the Target OS name you selected, this specifies, the available version names listed in the Version drop down list.	Select the appropriate target version.
Kernel Mode	If applicable to the Target OS name and version, this specifies the following modes: • User mode • Kernel mode	Select the Target kernel mode by selecting it from the drop down list.
Architecture	This specifies the architecture of the Target OS. The options are: • 32-bit • 64-bit	Select the architecture you need.
Load Settings	This specifies the following two options to load settings from: • **Previous**: If you select Previous, then initial values for this wizard are loaded from previously saved settings and populated. • **Default:** If you select Default, then the values from default settings are populated.	Select the option to load settings from by selecting from the drop down list.
Generate Project File	Specifies if you want to generate a project file.	Select the check box next to Generate project file.
Project	Specifies the different target project types that you can generate for this project. The generated project files are directly imported into the specified IDE (Eclipse/Visual Studio), and this project becomes a project of that IDE.	Select the project from the drop down list.
Destination Path	Specifies the path to place the generated target code.	Click **Browse** and select the folder to place the generated code.

4. On **Profiler Configuration** tab, define your profiler data specifications as shown in Figure 55.

Figure 55: Profiler Configuration

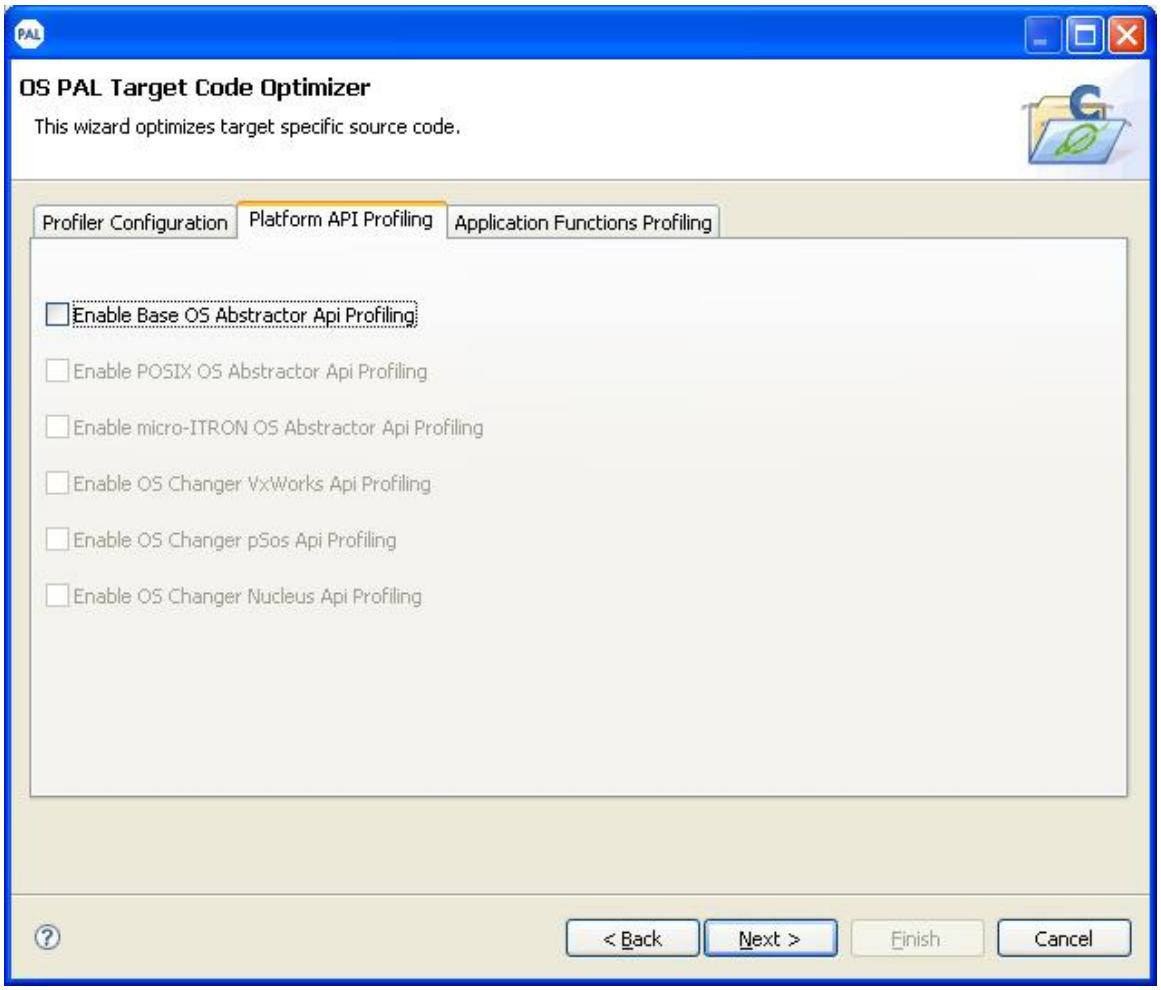


The field descriptions on Profiler Configuration tab are as follows

Field	Description	Your Action
Description	Specifies the description for the OS Abstractor project.	Type description for the OS Abstractor project.
Profiler Task Priority	Specifies the priority level of the profiler thread.	Enter a priority level for the profiler thread. The value can be between 0 through 225. The default value is set to 200.
File Path to Store Profiled Data	Specifies the directory location where the profiler file will be created.	Enter a data file path. The default location set is */root* on Unix based machines and *c:/* on MS Windows machine.
Number of Data in Memory Before Each Write	Specifies the depth of the profiler queue.	Enter the number of data in memory before each write.
Maximum Profiler Data to Collect	Specifies the maximum records collected in the XML file.	Enter the number of profiler messages.

1. On **Platform API Profiling** tab, select the check box to enable the Profiler as shown in Figure 56.

Figure 56: Platform API Profiling

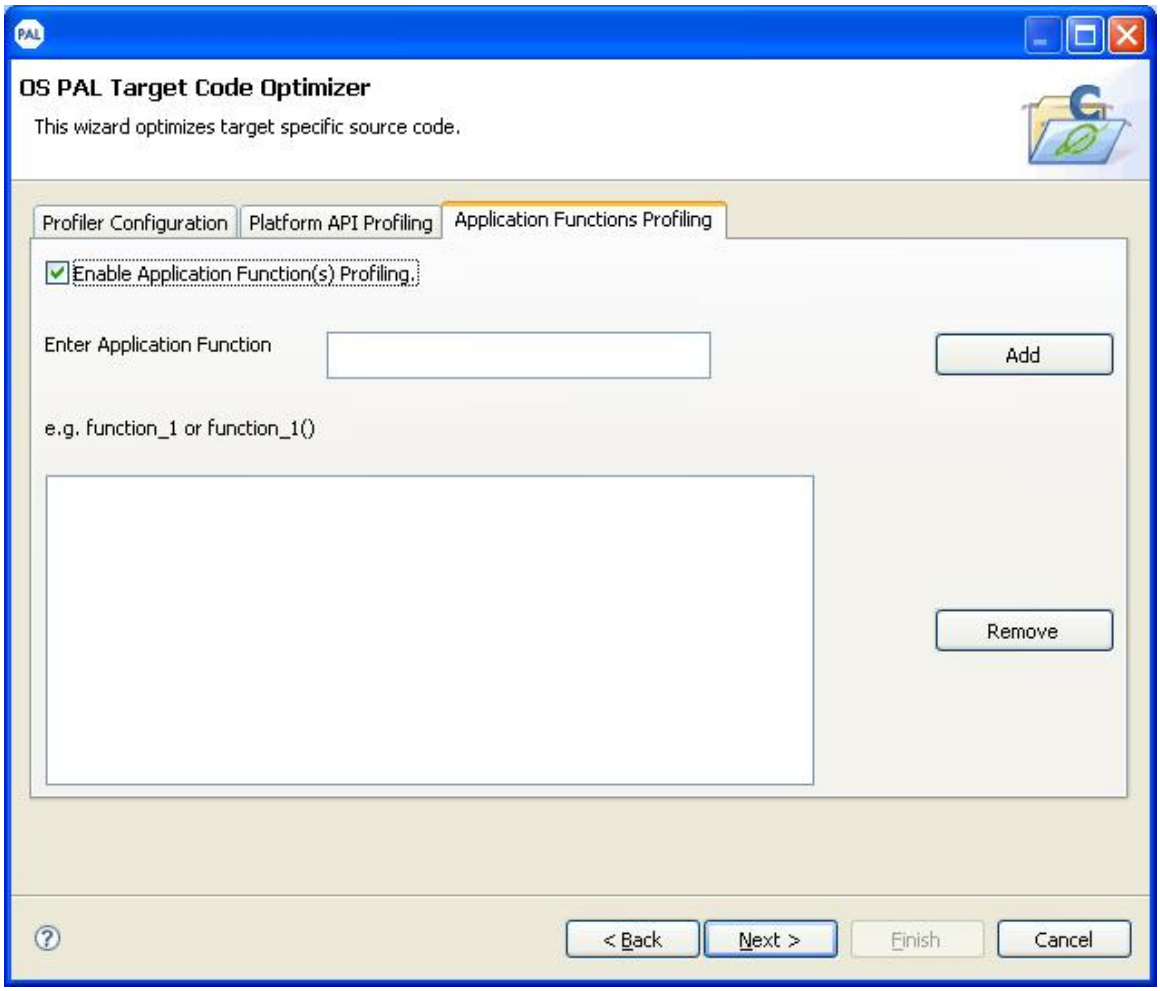


The field descriptions on Platform API Profiling tab are as follows:

Field	Description	Your Action
Enable Base OS Abstractor API Profiling	Specifies if the Base OS Abstractor API Profiling feature is enabled or disabled.	Select the check box to enable platform profiling. Platform Profiling means OS Abstractor APIs profiling. **NOTE**: By default, Base OS Abstractor API profiling is enabled for all projects.
Enable POSIX OS Abstractor API Profiling	Specifies if POSIX OS Abstractor API Profiling feature is enabled for your project.	Select the check box, if you need profiling for your POSIX APIs.
Enable micro-ITRON OS Abstractor API Profiling	Specifies if micro-ITRON OS Abstractor API Profiling feature is enabled for your project.	Select the check box, if you need profiling for your micro-ITRON APIs.
Enable OS Changer VxWorks API Profiling	Specifies if OS Changer VxWorks API Profiling feature is enabled for your project.	Select the check box, if you need profiling for your OS Changer VxWorks APIs.
Enable OS Changer pSOS API Profiling	Specifies if OS Changer pSOS API Profiling feature is enabled for your project.	Select the check box, if you need profiling for your OS Changer pSOS APIs.
Enable OS Changer Nucleus API Profiling	Specifies if OS Changer Nucleus API Profiling feature is enabled for your project.	Select the check box, if you need profiling for your OS Changer Nucleus APIs.

2. On **Application Functions Profiling** tab, you can also perform profiling for your specific APIs as shown in Figure 57.

Figure 57: Application Function Profiling



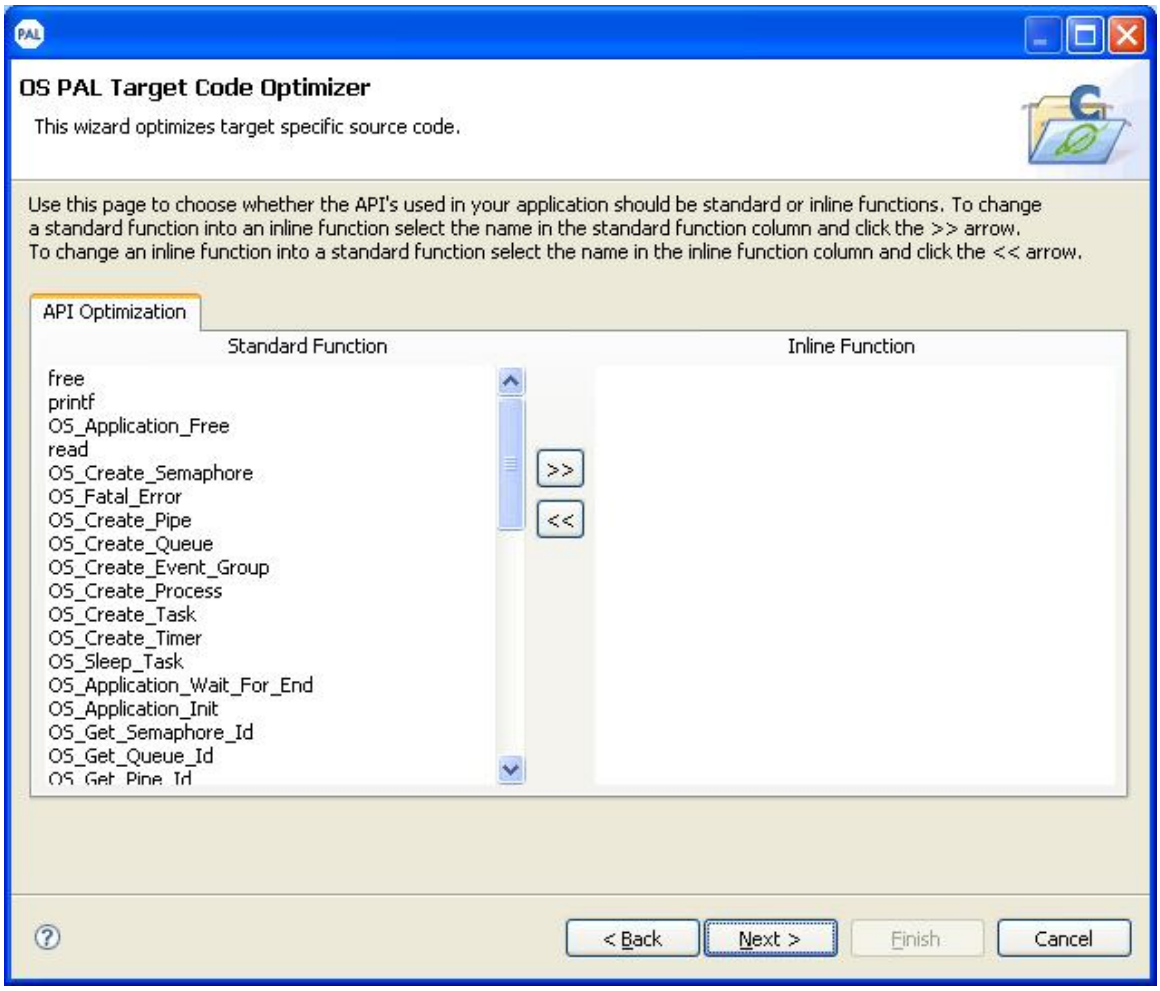
The field descriptions on Application Functions Profiling tab are as follows:

Field	Description	Your Action
Enable Application Functions Profiling	Specifies if the Application Functions Profiling feature is enabled or disabled.	Select the check box to enable Application Functions profiling. This profiling is used for User APIs profiling.
Enter Application Function	Specifies the name of the Application Function for profiling.	Enter the name of the application function. NOTE that this field is case Sensitive.
Add	Specifies if you want to add any application functions.	To add any application function, enter the name in the text box, and click **Add.**
Remove	Specifies if you want to remove any application functions from the list.	To remove any application function from the list, select the name of the application function in the text box, and click **Remove.**

3. Add your APIs by typing in the name of the API next to **Enter Application Function** text box and click **Add** and click **Next**.

4. On **API Optimization** tab, select the APIs that should be moved to a standard function or an inline function and move the selected API using the double arrow button as shown in Figure 58. Inline functions will execute faster but will increase the memory footprint of the application and click **Next**.

Figure 58: API Optimization



Need for Code Optimization: Inline expansion is used to eliminate the time overhead when a function is called. It is typically used for functions that execute frequently. It also has a space benefit for very small functions, and is an enabling transformation for other optimizations.

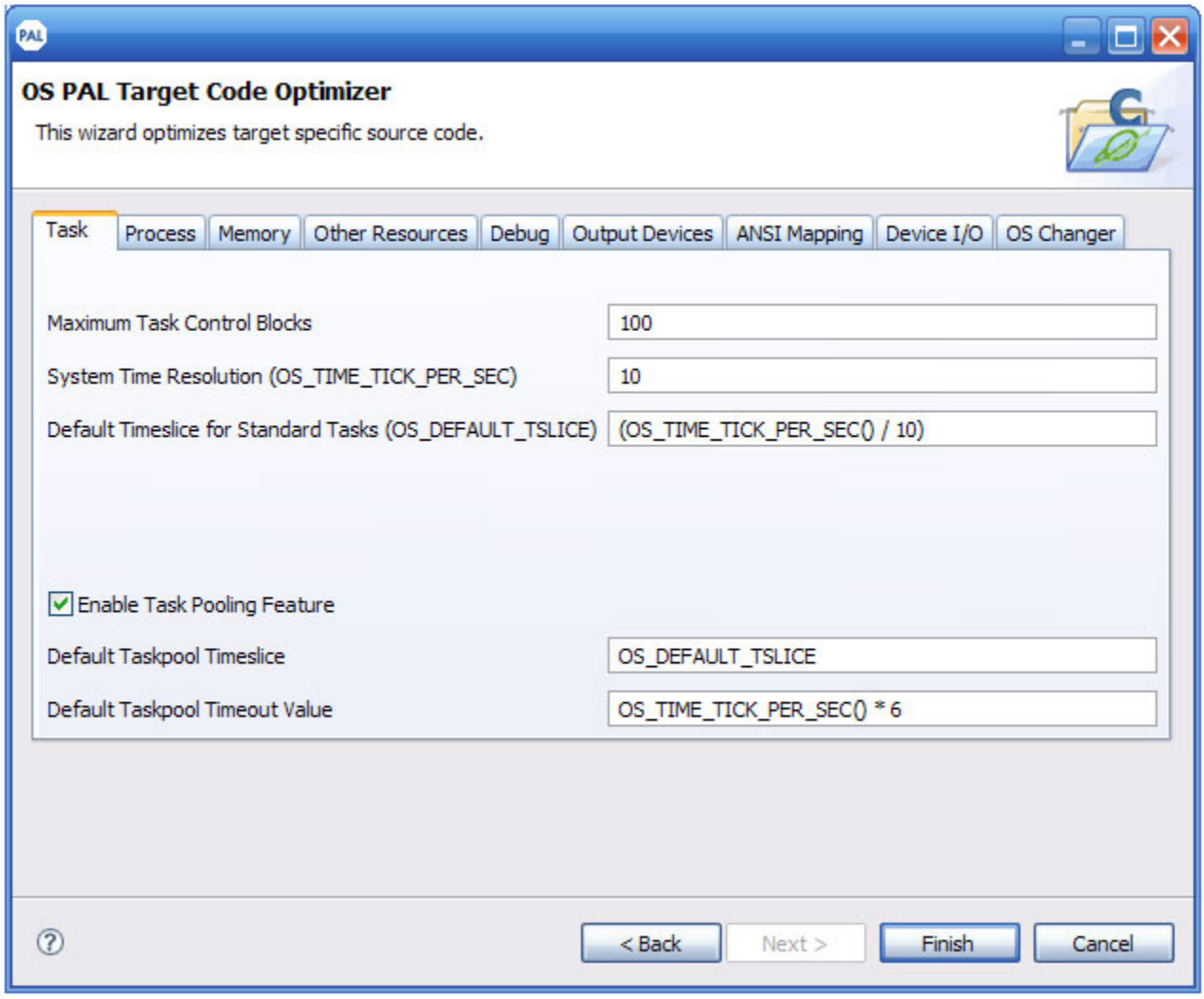
Without inline functions, however, the compiler decides which functions to inline. The programmer has little or no control over which functions are inlined and which are not. Giving this degree of control to the programmer allows her/him to use application-specific knowledge in choosing which functions to inline.

The field descriptions on API Optimization tab are as follows:

Field	Description	Your Action
Standard Function	Specifies if the APIs used in your application are standard functions.	Select the functions used in this application as standard functions for the target OS project. You can select multiple function names at once to place them in the other list. You can select all function names in a list using the selectAll (Ctrl+A) action also.
Inline Function	Specifies that a compiler inserts the complete body of the function in every place in the code where that function is used. It is used to eliminate the time overhead when a function is called and execute it frequently	To select a standard function into an Inline function, select the API and click the right arrow. To select an Inline function into a standard function, select the API under Inline function, and click the left arrow. **NOTE:** You can use optimization for this. If a function is being called repeatedly, they can improve the performance by making this function inline.

5. On **Task** configuration tab, configure the options to your specifications as shown in Figure 59. Applications can create OS Abstractor tasks during initialization and will be able to re-use the task envelope repeatedly by selecting the check box next to **Enable Task Pooling Feature**.

Figure 59: Task Tab



NOTE: In the current release, Task Pooling feature is not supported in ThreadX and Nucleus targets.

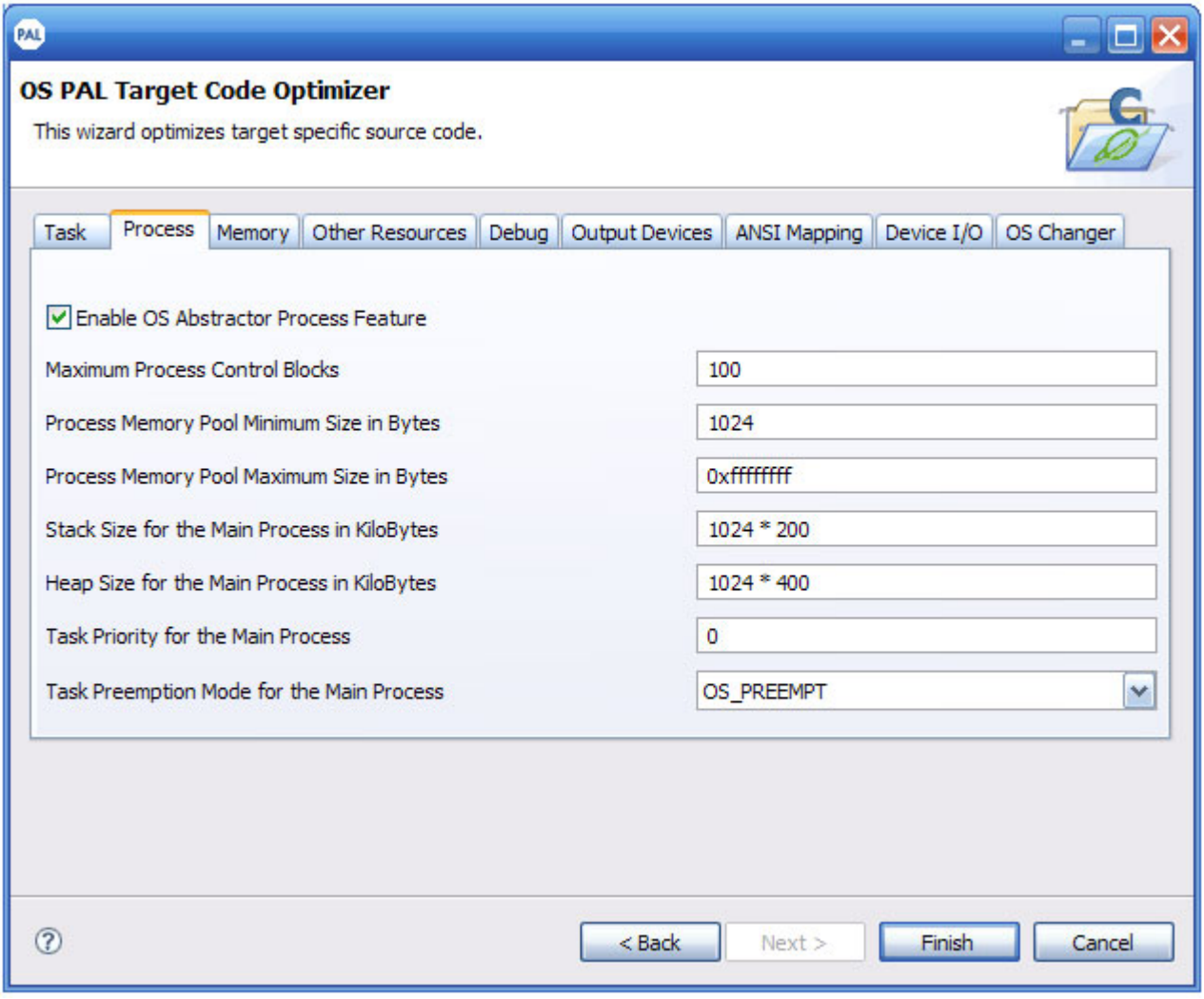
The field descriptions on Task tab are as follows:

Field	Description	Your Action
Maximum Task Control Blocks	Specifies the total number of tasks required by the application.	Enter a value. **NOTE:** The default value is 100. One control block will be used by the OS_Application_Init function when the INCLUDE_OS_PROCESS option is true.
System Time Resolution (OS_TIME_TICK_PER_SEC)	Specifies the system clock ticks (not hardware clock tick). For example, when you call OS_Task_Sleep(5), you are suspending task for a period (5* OS_TIME_RESOLUTION).	Enter a value. **NOTE:** The default value is 10000 micro second (= 10milli sec). This value is derived from the target OS. If you cannot derive the value, refer to the target OS reference manual and set the correct per clock tick value. **NOTE:** Since the system clock tick resolution may vary across different OS under different target, it is recommended that the application use the macro OS_TIME_TICK_PER_SEC to derive the timing requirement instead of using the raw system tick value in order to keep the application portable across multiple OS.
Default Timeslice for Standard Tasks (OS_DEFAULT_TSLICE)	Specifies the default time slice scheduling window width among the same priority pre-emptable threads when they are all in ready state.	Enter a default timeslice for standard tasks. **NOTE**: The default value is 10 ms. If system tick is 10ms, then the threads will be scheduled round-robin at the rate of every 100ms. **NOTE**: On Linux operating system, the time slice cannot be modified per thread. OS Abstractor ignores this setting and only uses the system default time slice configured for the Linux kernel.
Enable Task Pooling Feature	Specifies if the Task pooling feature is enabled for this application. Task pooling feature enhances the performances and reliability of application. If you	To enable task pooling feature, select the check box.

	enable the task pooling feature, applications can create OS Abstractor tasks during initialization and be able to re-use the task envelope repeatedly. To configure task-pooling, set the following pre-processor flag as follows: INCLUDE_OS_TASK_POOLING.	
Default Taskpool Timeslice	Specifies the default Taskpool Timeslice.	Enter the default Taskpool Timeslice. **NOTE**: The default value is OS_DEFAULT_TSLICE.
Default Taskpool Timeout Value	Specifies the default Taskpool timeout value.	Enter the default Taskpool timeout value. **NOTE**: The default value is OS_TIME_TICK_PER_SEC() * 6.

6. On **Process** configuration tab, configure the options to your specifications as shown in Figure 60. Select the check box next to **Enable OS Abstractor Process Feature** to allocate the memory from a shared memory region to allow applications to communicate across multiple processes. By disabling this option, the memory will be allocated from the individual application/process specific pool, which is created during the OS_Application_Init function call.

Figure 60: Process Tab

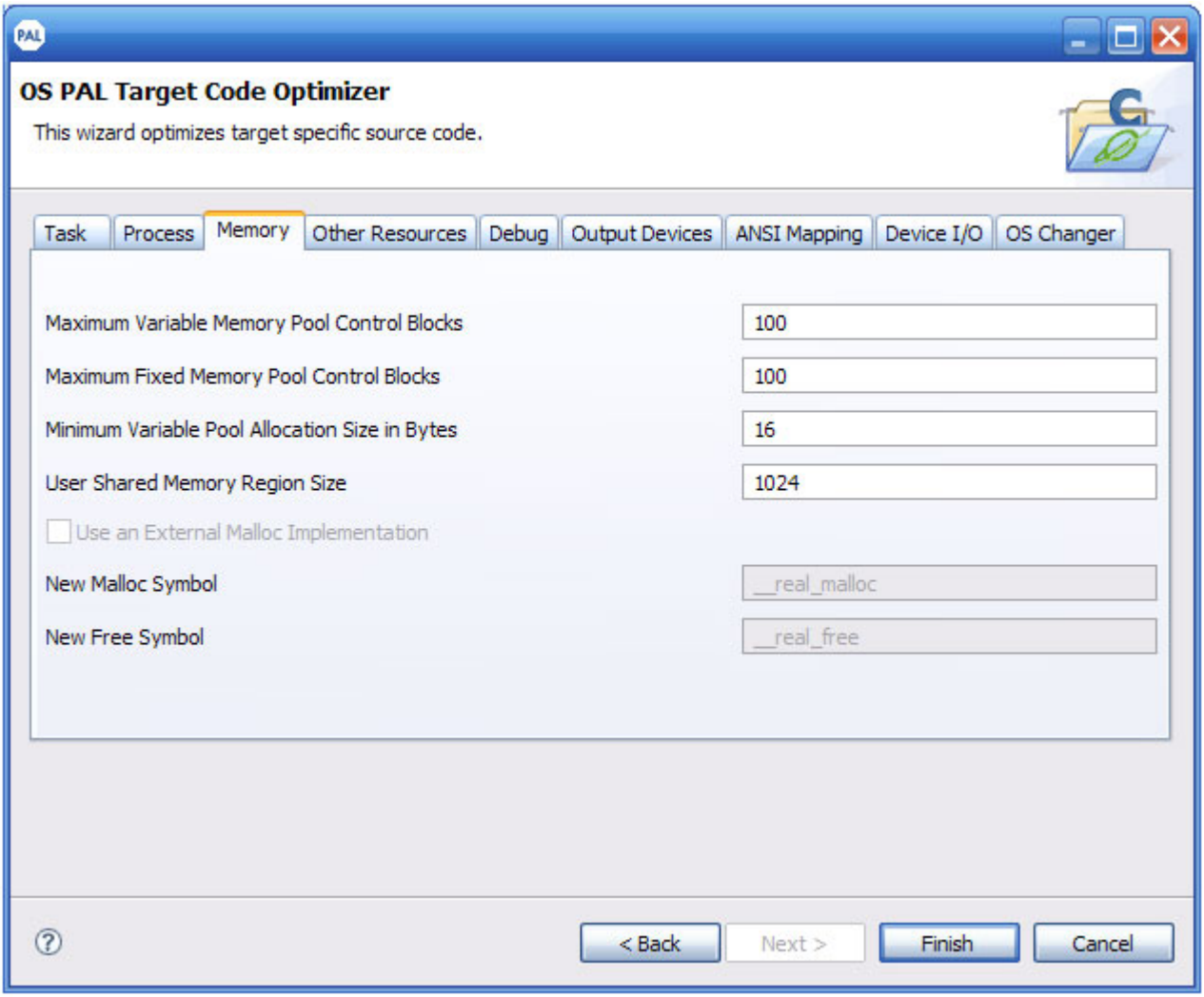


The field descriptions on Process tab are as follows:

Field	Description	Your Action
Enable OS Abstractor Process Feature	Specifies if the OS Abstractor process feature is enabled or disabled.	Select the check box to enable this feature.
Maximum Process Control Blocks	Specifies the total number of processes required by the application	Enter the maximum number of process control blocks for the application. **NOTE**: Default value is 100.
Process Memory Pool Minimum Size in Bytes	Specifies the minimum size of the process memory pool in Bytes.	Enter the minimum size of the process memory pool. **NOTE**: Default value is 1024 Bytes.
Process Memory Pool Maximum Size in Bytes	Specifies the maximum size of the process memory pool in Bytes.	Enter the maximum size of the process memory pool. **NOTE**: Default value is 0xffffffff Bytes.
Stack Size for the Main Process in kilobytes	Specifies the stack size for the main process in Kilobytes.	Enter the stack size for the main process. **NOTE**: Default value is 1024 * 200 Kilobytes.
Heap Size for the Main Process in kilobytes	Specifies the heap size for the main process in Kilobytes.	Enter the heap size for the main process. **NOTE**: Default value is 1024 * 400 Kilobytes.
Task priority for the Main Process	Specifies the task priority for the main process.	Enter the task priority for the mail process. **NOTE**: Default value is 0.
Task Preemption Mode for the Main Process	Specifies the preemption status of this task.	Enter the task preemption status of the task. **NOTE**: The valid parameters are: ▪ OS_PREEMPT – Task can be pre-empted by the system. ▪ OS_NO_PREEMPT – Task cannot be pre-empted.

7. On **Memory** configuration tab, configure the options to your specifications as shown in Figure 61.

Figure 61: Memory Tab

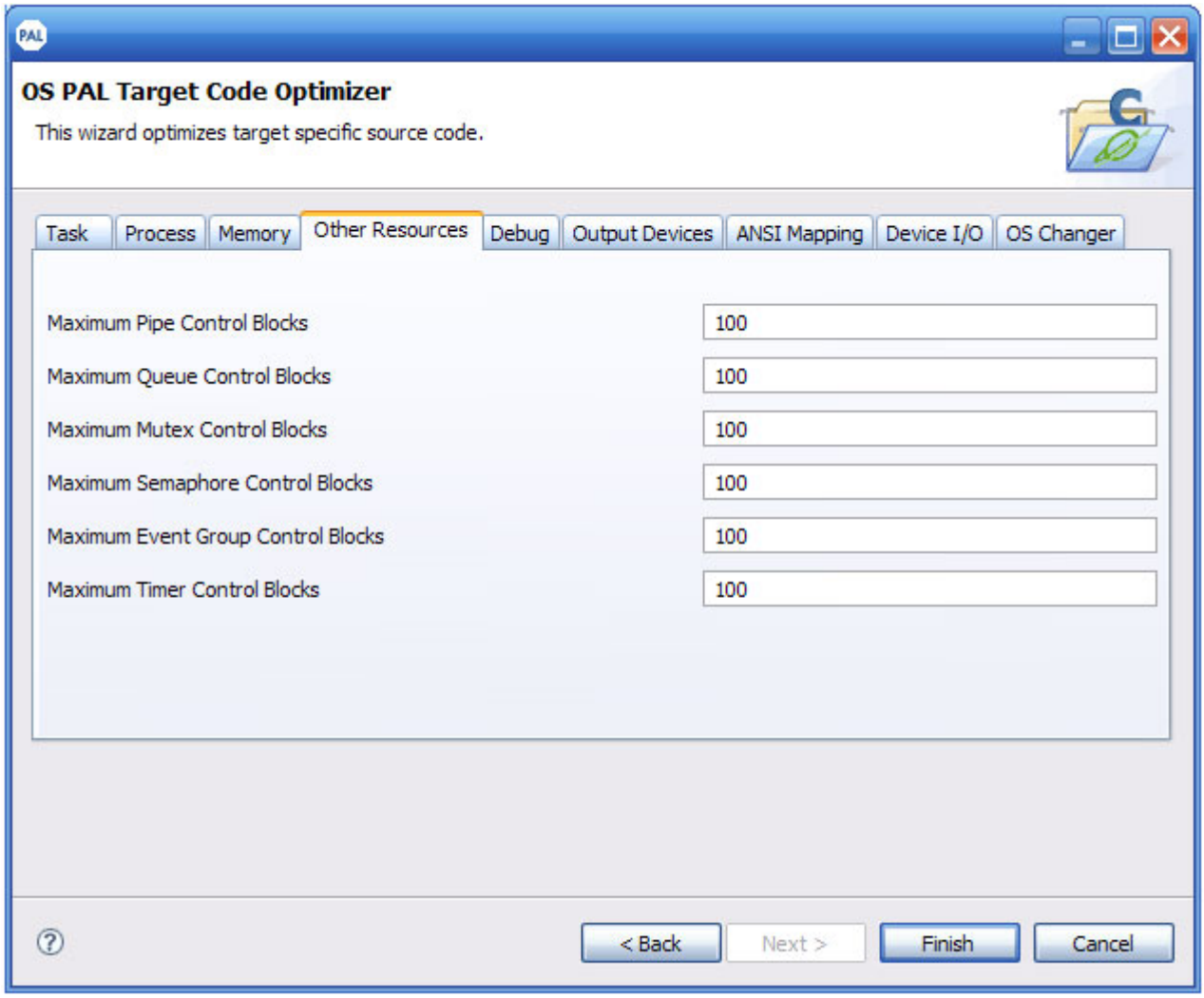


The field descriptions on Memory tab are as follows:

Field	Description	Your Action
Maximum Variable Memory Pool Control Blocks	Specifies the total number of dynamic variable memory pools required by the application.	Enter the maximum number of dynamic variable pools. **NOTE**: Default value is 100.
Maximum Fixed Memory Pool Control Blocks	Specifies the total number of partitioned (fixed-size) memory pools required by the application.	Enter the maximum number of partitioned memory pools. **NOTE**: Default value is 100.
Minimum Variable Pool Allocation Size in Bytes	Specifies the minimum memory allocated by *the malloc()* and/or OS_Allocate_Memory() calls. **NOTE**: Increasing this value further reduces memory fragmentation at the cost of more wasted memory.	Enter the minimum memory allocated. **NOTE**: Default value is 16. Increasing this value further reduces memory fragmentation at the cost of more wasted memory.
User Shared Memory Region Size	Specifies the application defined shared memory region usable across all OS Abstractor processes/applications.	Enter the user shared memory region size. **NOTE**: Default value is 1024 Bytes.

8. On **Other Resources** configuration tab, configure the options to your specifications as shown in Figure 62.

Figure 62: Other Resources Tab

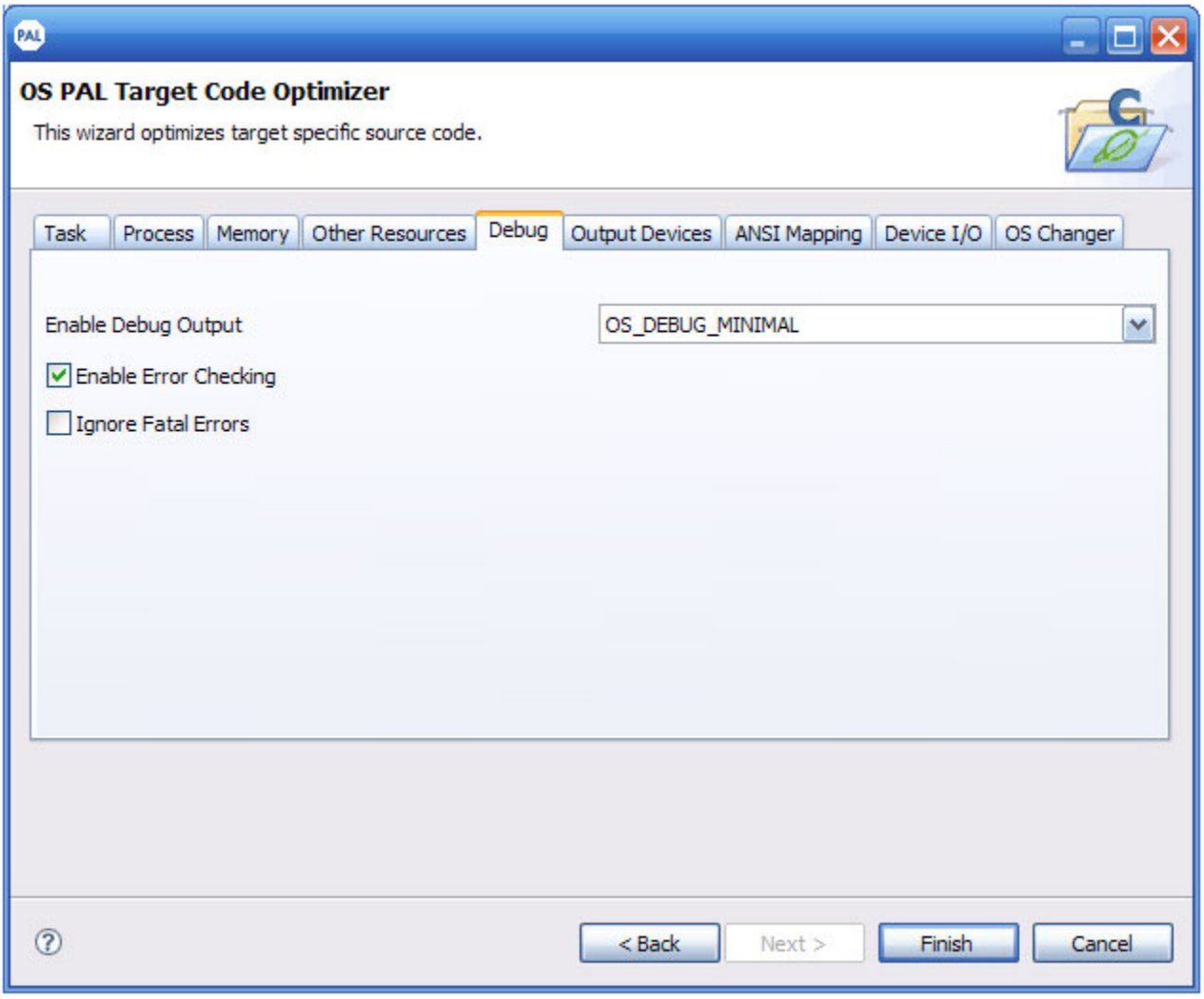


The field descriptions on Other Resources tab are as follows:

Field	Description	Your Action
Maximum Pipe Control Blocks	Specifies the total number of pipes for message passing required by the application.	Enter the maximum number of pipe control blocks. **NOTE**: Default value is 100.
Maximum Queue Control Blocks	Specifies the total number of queues for message passing required by the application.	Enter the maximum number of queue control blocks. **NOTE**: Default value is 100.
Maximum Mutex Control Blocks	Specifies the total number of mutex semaphores required by the application.	Enter the maximum number of mutex control blocks. **NOTE**: Default value is 100.
Maximum Semaphore Control Blocks	Specifies the total number of regular (binary/count) semaphores required by the application.	Enter the maximum number of semaphore control blocks. **NOTE**: Default value is 100.
Maximum Event Group Control Blocks	Specifies the total number of event groups required by the application	Enter the maximum number of event group control blocks. **NOTE**: Default value is 100.
Maximum Timer Control Blocks	Specifies the total number of application timers required by the application	Enter the maximum number of timer control blocks. **NOTE**: Default value is 100.

9. On **Debug** configuration tab, configure the options to your specifications as shown in Figure 63. The application will be checked for API usage errors by selecting the check box next to **Enable Error Checking**. Disabling error checking will increase the application performance and reduce your code size.

Figure 63: Debug Tab

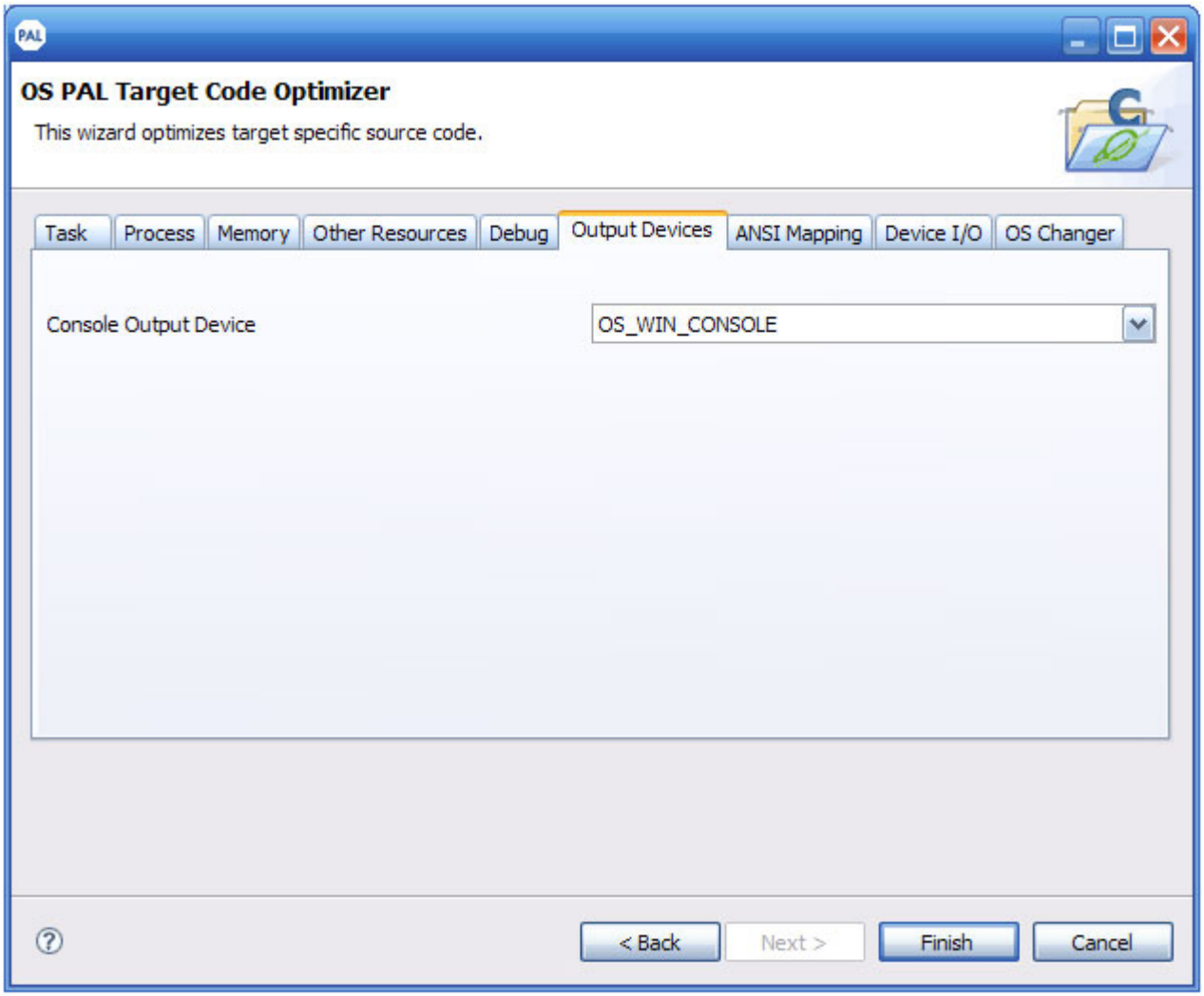


The field descriptions on Debug tab are as follows:

Field	Description	Your Action
Enable Debug Output	Specifies if you want to enable the debug output.	Select the debug output from the dropdown menu: ▪ OS_DEBUG_VERBOSE – print debug info, fatal and compliance errors ▪ OS_DEBUG_MINIMUM – print minimum amount of debug info ▪ OS_DEBUG_VERBOSE **NOTE**: The default value is OS_DEBUG_VERBOSE
Enable Error Checking	Specifies if you want to enable the error checking.	To enable error checking, select the check box. Use this option to increase performance and reduce code size. **NOTE**: By default this feature is enabled.
Ignore Fatal Errors	Specifies if you want to enable the feature to ignore fatal errors.	To enable the feature to ignore fatal errors, select the check box. **NOTE**: By default this feature is disabled.

10. On **Output Devices** configuration tab, select your output device from the drop drown list as shown in Figure 64.

Figure 64: Output Devices Tab

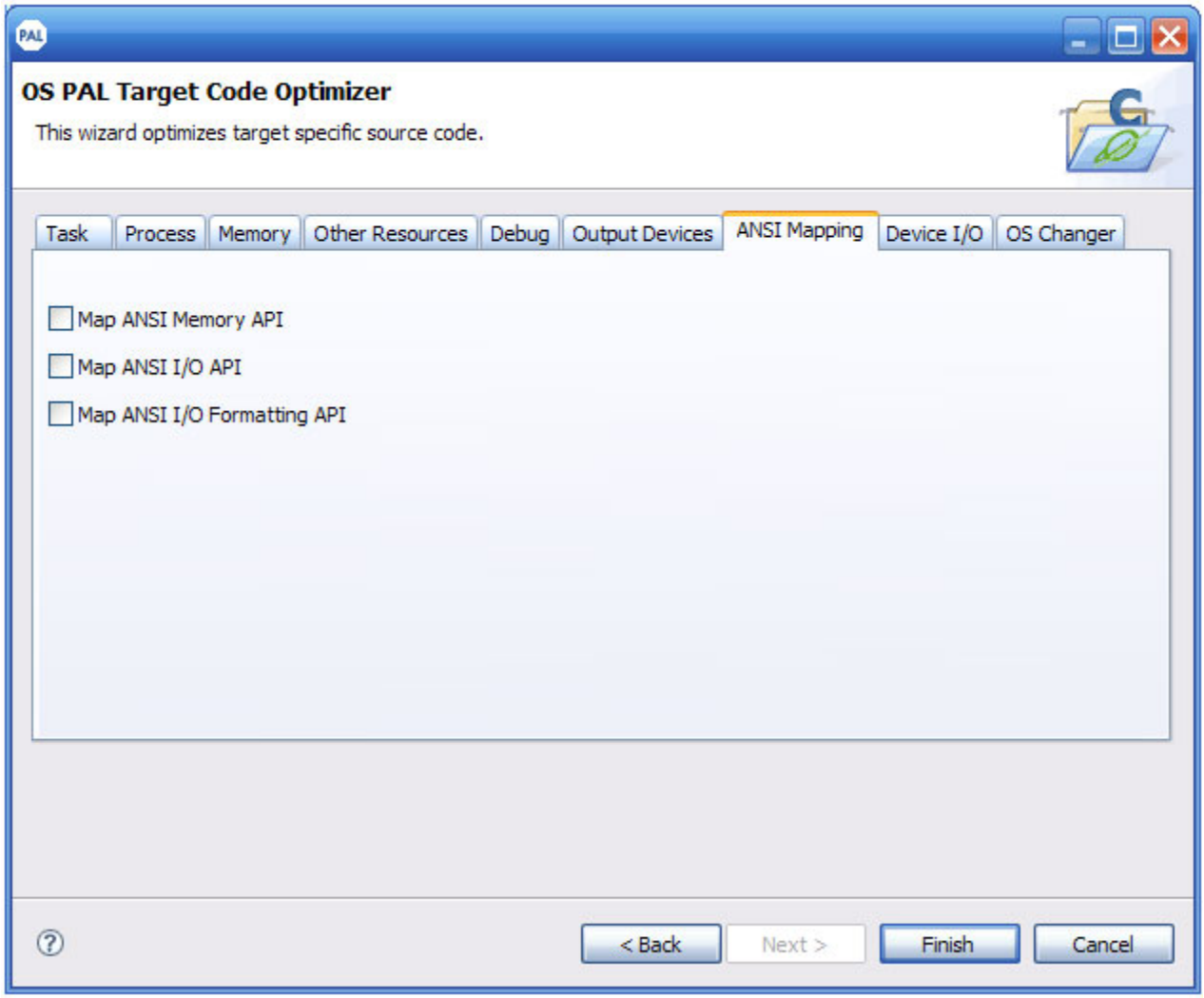


The field descriptions on Output Devices tab are as follows:

Field	Description	Your Action
Console Output Device	Specifies the console output device for the application.	Select the output device from the dropdown menu: ▪ OS_WIN_CONSOLE – print to console ▪ OS_SERIAL_PORT – print to serial **NOTE**: The default value is OS_WIN_CONSOLE User can print to other devices by modifying the appropriate functions within *usr.c* and use OS Abstractor's format i/o calls.

11. On **ANSI Mapping** configuration tab, choose whether to use ANSI mapping as shown in Figure 65. If the ANSI mapping is checked, the application will use MapuSoft's malloc.

Figure 65: ANSI Mapping Tab

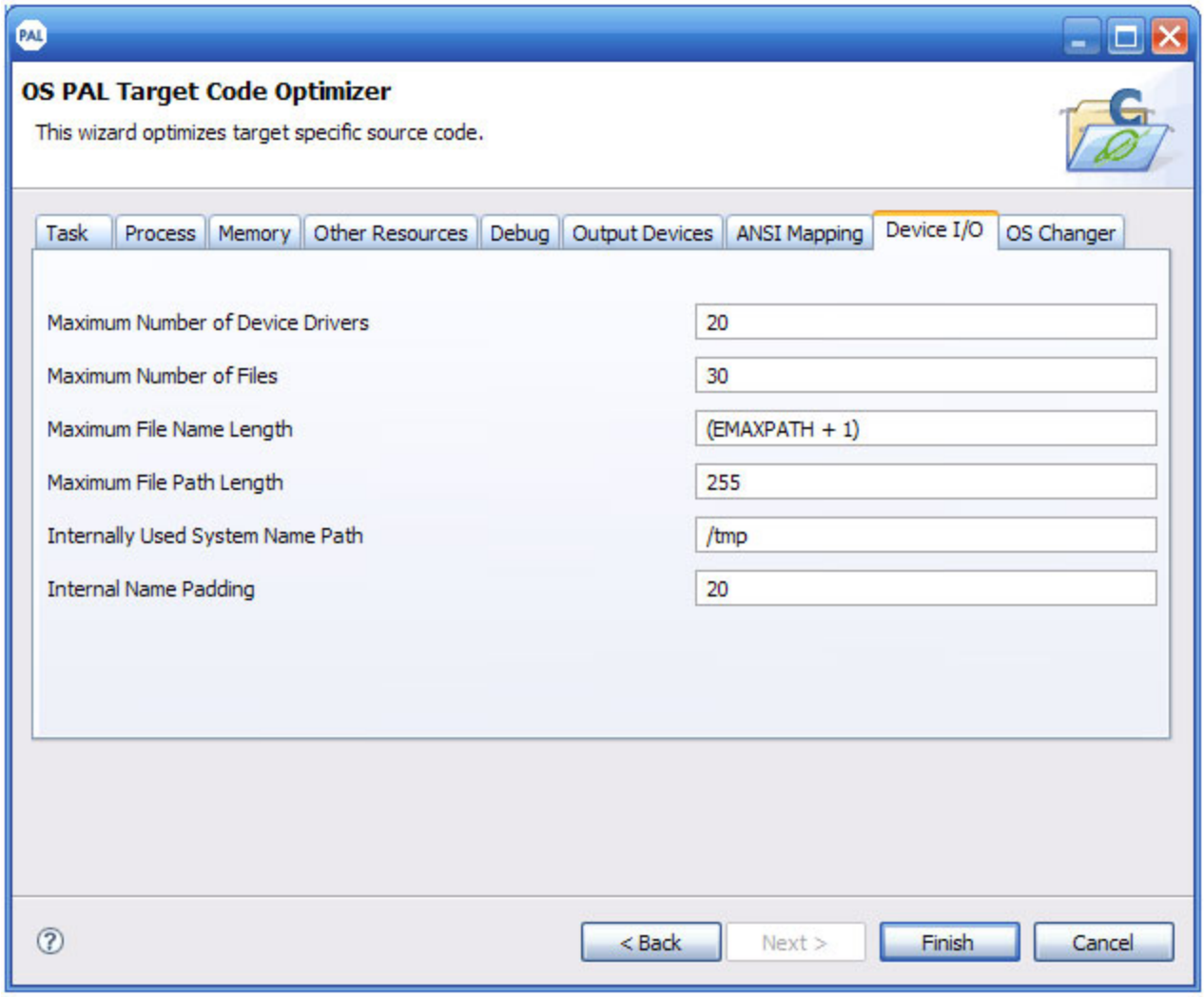


The field descriptions on ANSI Mapping tab are as follows:

Field	Description	Your Action
Map ANSI Memory API	Specifies you want to map ANSI malloc() and free() to OS Abstractor equivalent functions.	To map ANSI to OS Abstractor equivalent functions, select the check box. **NOTE**: By default this feature is enabled.
Map ANSI I/O API	Specifies if you want to map ANSI device I/O functions like open(), close(), read(), write, ioctl(), etc. to OS Abstractor equivalent functions.	To map ANSI I/O functions to OS Abstractor equivalent functions, select the check box. **NOTE**: By default this feature is disabled.
MAP ANSI I/O Formatting API	Specifies if you want to map ANSI printf() and sprintf() to OS Abstractor equivalent functions.	To map ANSI I/O formatting functions to OS Abstractor equivalent functions, select the check box. **NOTE**: By default this feature is disabled.

12. On **Device I/O** configuration tab, configure the options to your specifications as shown in Figure 66.

Figure 66: Device Input or Output Tab



The field descriptions on Device I/O tab are as follows:

Field	Description	Your Action
Maximum Number of Device Drivers	Specifies the maximum number of drivers allowed in the OS Abstractor driver table structure. **NOTE**: This excludes the native drivers the system, since they do not use the OS Abstractor driver table structure.	Enter the maximum number of device drivers. **NOTE**: Default value is 20.
Maximum Number of Files	Specifies the maximum number of files that can be opened simultaneously using the OS Abstractor file control block structure. **NOTE**: One control block is used when an OS Abstractor driver is opened. These settings do not impact the OS setting for max number of files.	Enter the maximum number of files that can be opened simultaneously. **NOTE**: Default value is 30.
Maximum File Name Length	Specifies the maximum length of the file name.	Enter the maximum number of files that can be opened simultaneously. **NOTE**: Default value is Maximum File Path Length +1.
Maximum File Path Length	Specifies the maximum length of the directory path name including the file name for OS Abstractor use excluding the null char termination.	Enter the maximum length of the file path. **NOTE**: Default value is 255. This setting does not impact the OS setting for the max path/file name.
Internally Used System Name Path	Specifies the temporary directory of the file path.	Enter the temporary directory of the file path. **NOTE**: Default value is /tmp.
Internal Name Padding	Specifies the padding for the internal name.	Enter the padding for the internal name. **NOTE**: Default value is 20.

13. If your project uses pSOS OS Changer, in **OS Changer** configuration tab, assign the number of unsigned arrays used to store the task's data as shown in

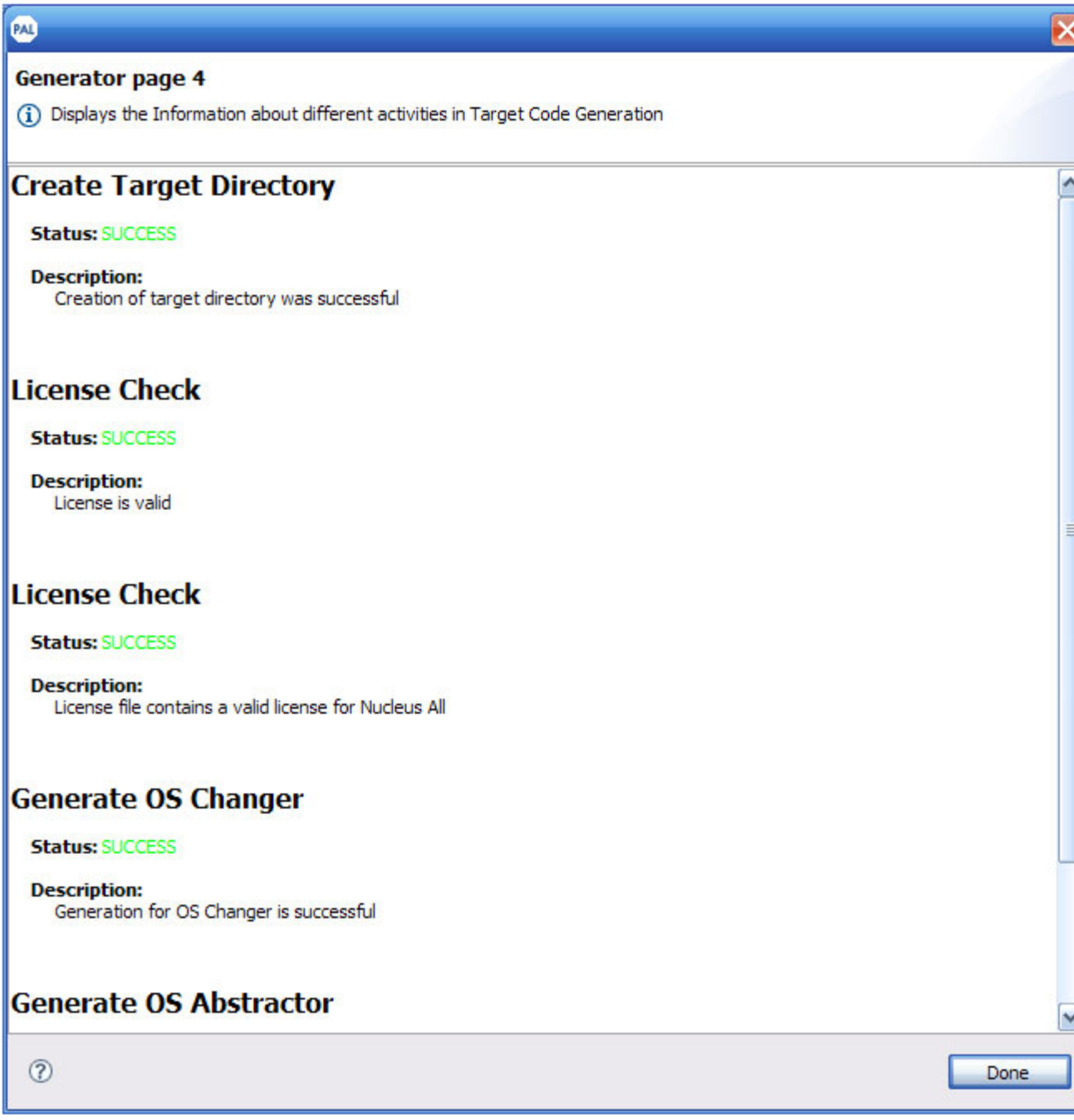
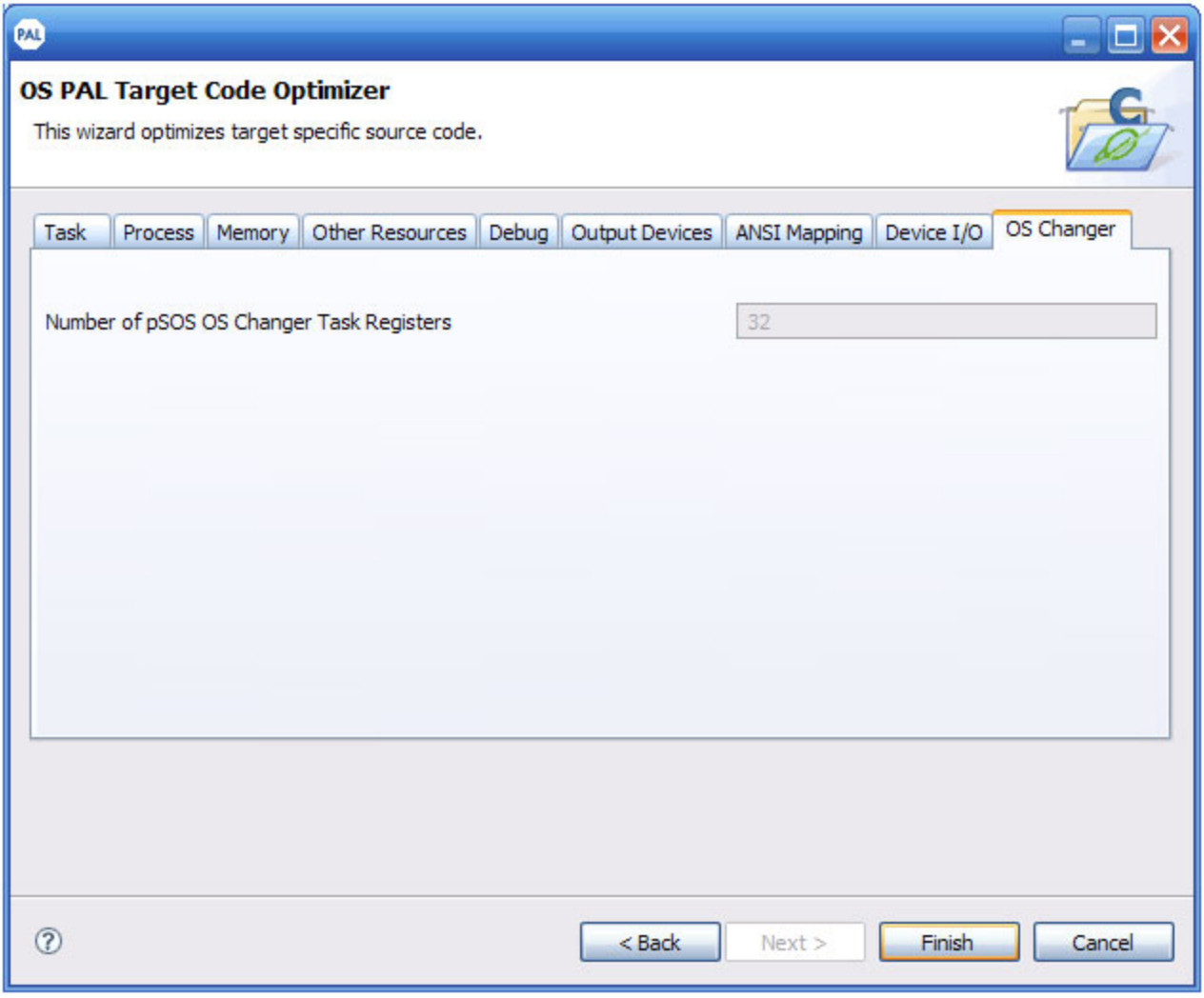


Figure 67.

Figure 67: OS Changer Tab

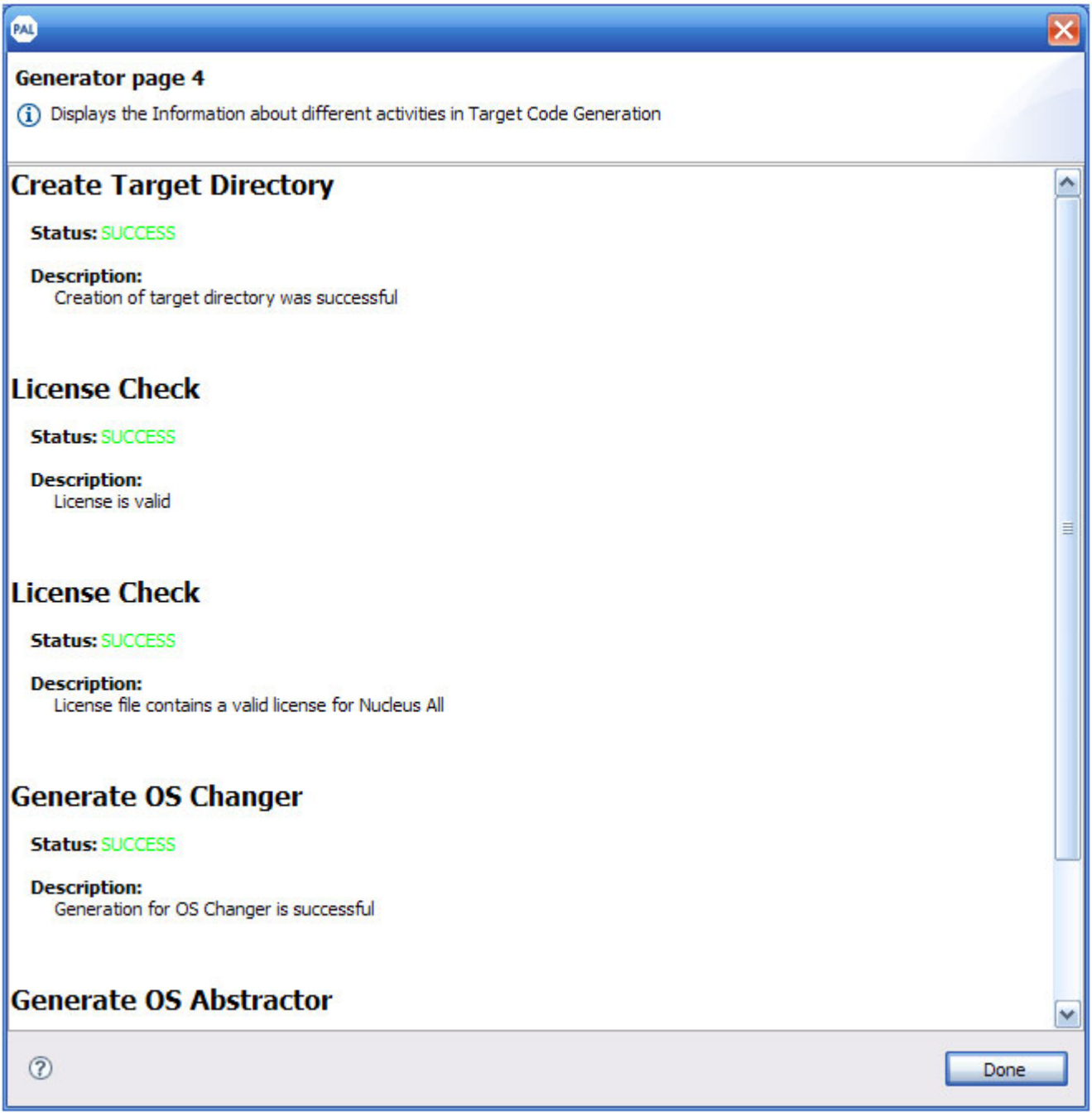

OS PAL Target Code Optimizer
This wizard optimizes target specific source code.
Task
Process
Memory
Other Resources
Debug
Output Devices
ANSI Mapping
Device I/O
OS Changer
Number of pSOS OS Changer Task Registers
32
< Back
Next >
Finish
Cancel

The field descriptions on OS Changer tab are as follows:

Field	Description	Your Action
Number of pSOS OS Changer Task Registers	Specifies the number of pSOS OS Changer Task Registers.	Enter the number of pSOS OS Changer task registers. **NOTE**: Default value is 32.

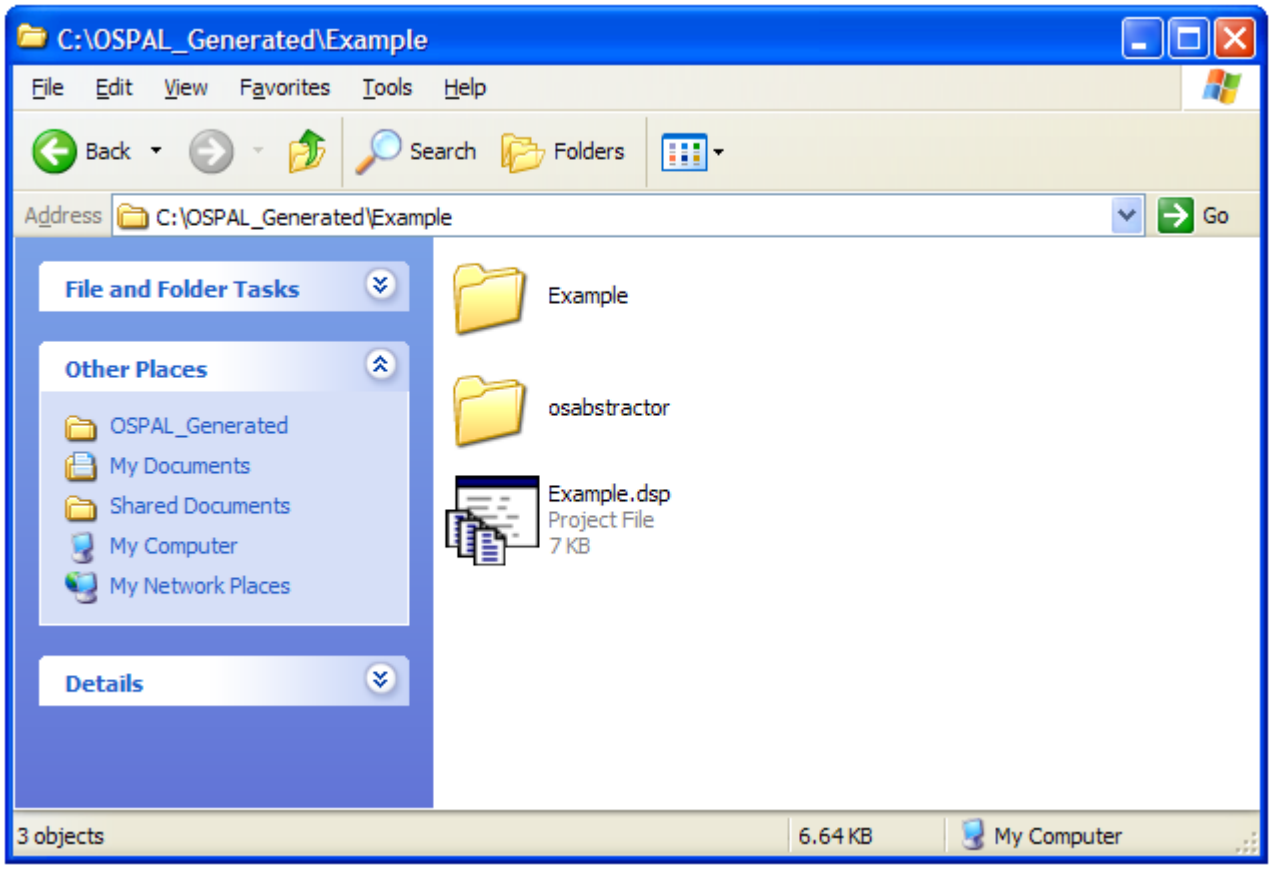
14. Click **Finish**. The target code will be generated into the destination path you defined in step 5 as shown in Figure 68. **NOTE**: If it is not able to generate the target code, the system will throw up an error.

Figure 68: Target Code Generation Output



15. You can view the OS Pal generated code in Figure 69.

Figure 69: OS PAL Generated Example



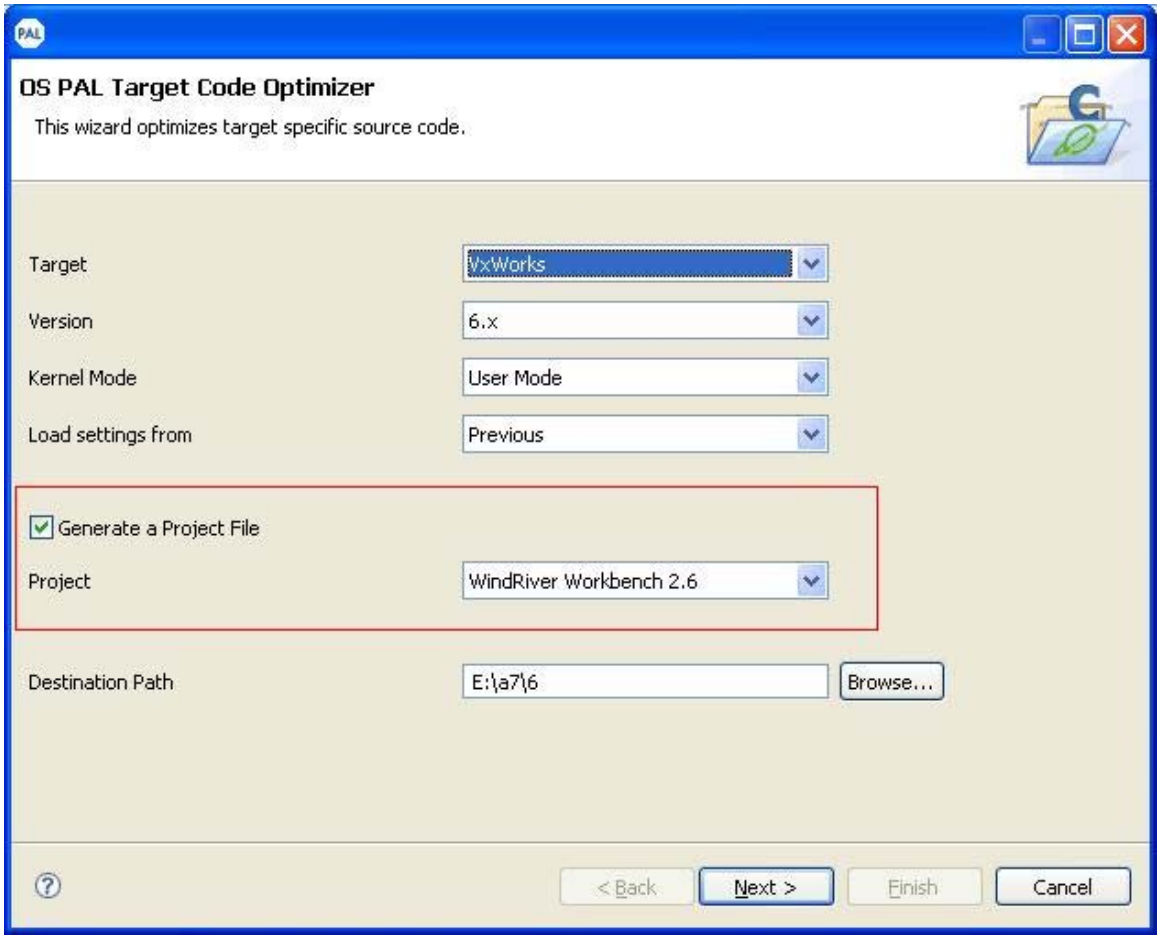
Generating Project Files for your Target

NOTE: This feature requires a target license. Click http://mapusoft.com/contact/ to send a request to receive licenses and documentation.

OS PAL provides the ability to generate project files for project files for Wind River's Workbench, QNX's Momentics, Sun Microsystem's Sun Studio, Eclipse's CDT, and makefiles.

When generating code for your target, select the check box next to **Generate a Project File** and choose your IDE as shown in Figure 70.

Figure 70: Generating Project Files



Running OS PAL Generated Code on your Target

NOTE: This feature requires a license and documentation.
Click http://mapusoft.com/contact/ to send a request to receive licenses and documentation.

After Generating the code for your target OS using the OS PAL target code optimizer,

1. Using a cross-compiler, compile, link, and download the OS PAL generated code to your target.
2. Port low level drivers and hardware interrupt code as required (refer to OS Abstractor I/O and device driver APIs sections in the reference manual).
3. Resolve any run time errors.

UPDATING OS PAL

Getting Updates for OS PAL

NOTE: This feature requires OSPAL Host License. Click http://www.mapusoft.com/contact/ to send a request to receive licenses and documentation.

You can get latest OS PAL updates from http://www.mapusoft.com using the following two options:

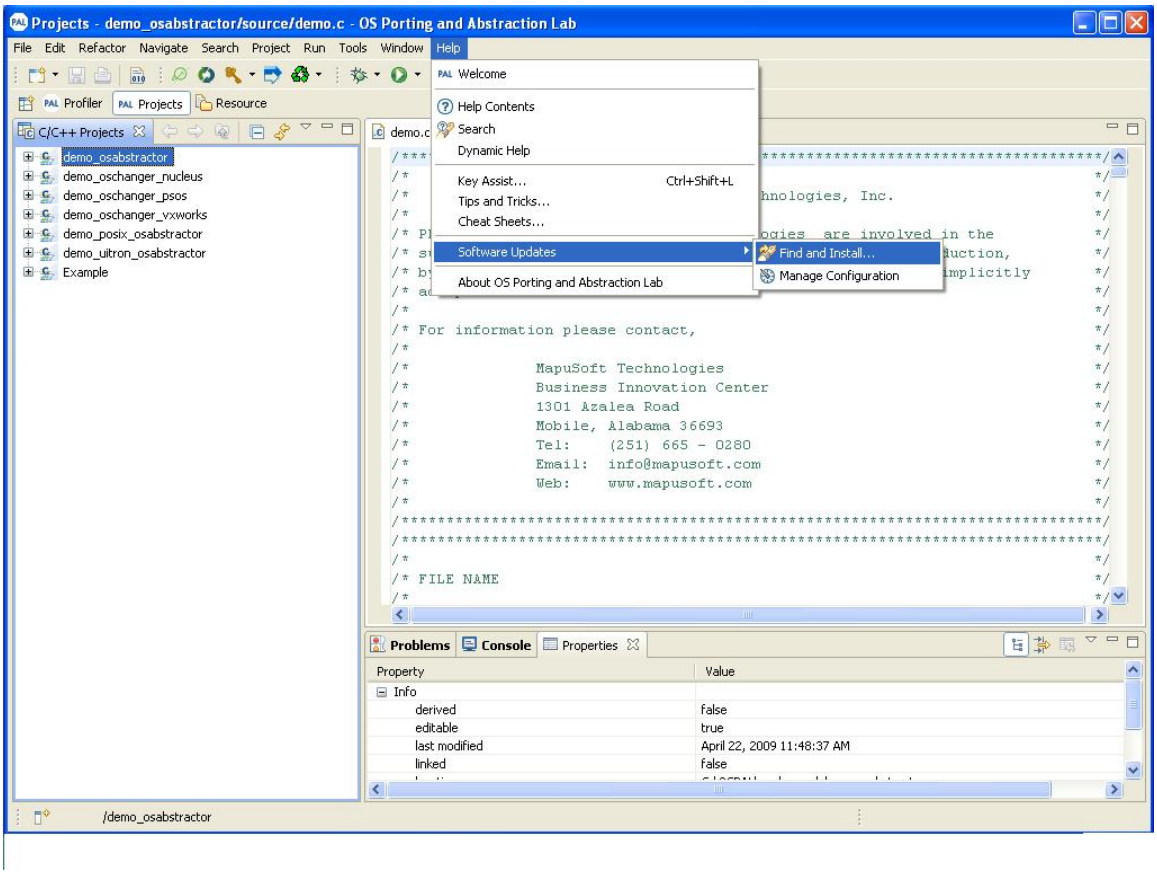
- **Remote Update**: By using Remote Update Site, the system will automatically contact http://www.mapusoft.com/ website and search for the latest updates. You need internet connectivity for this to work.
- **Local Update**: By using Local Update Site, you can do OSPAL updates without connecting to the Internet. For this to work, you need to get the updated files from http://www.mapusoft.com/ by e-mail or CD.

Updating Software Using Remote Update Site

To update software using Remote update site:

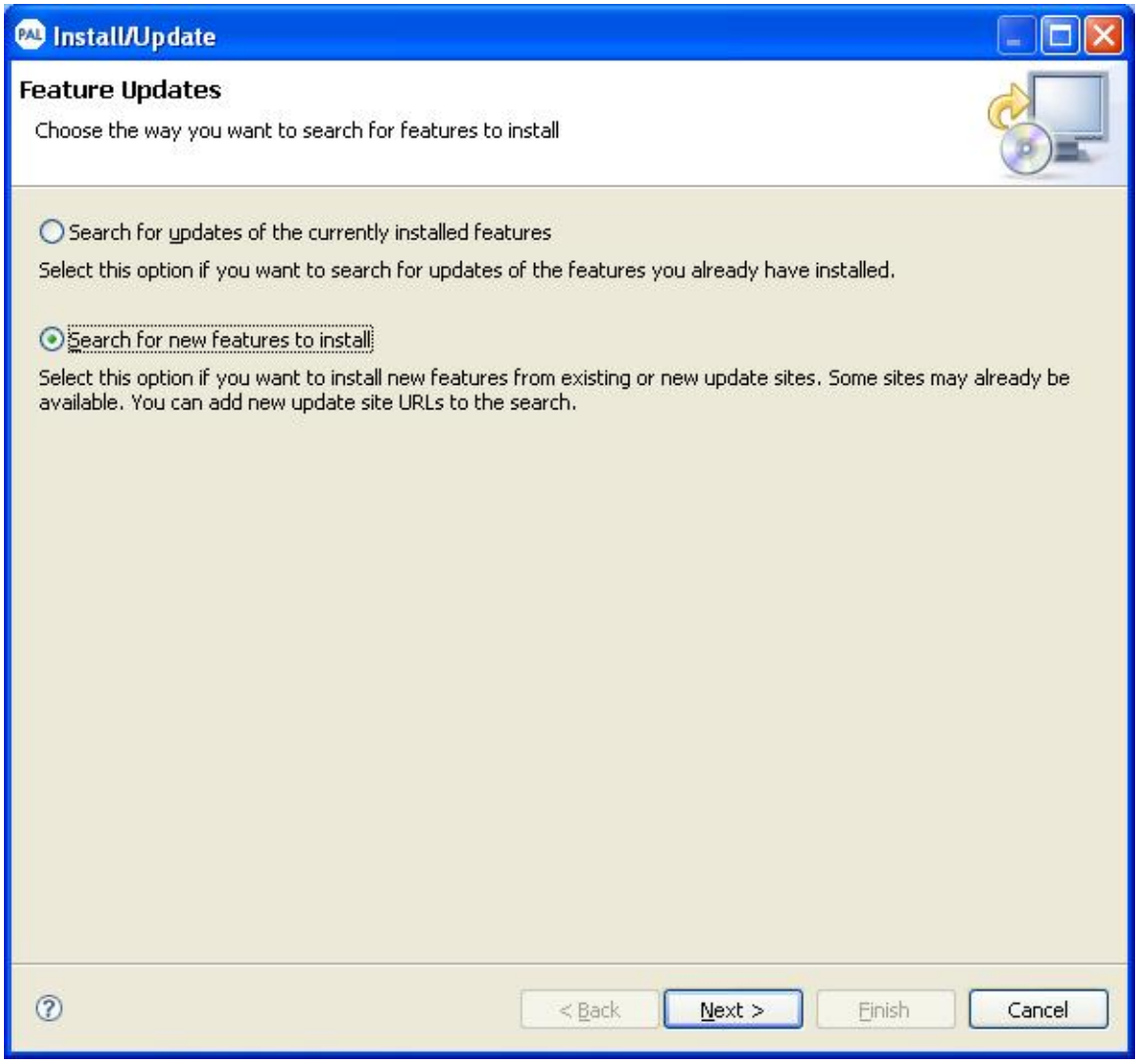
1. From OS PAL main menu, select **Help > Software Updates > Find and Install** as shown in Figure 71.

Figure 71: Software Updates Using Remote Site



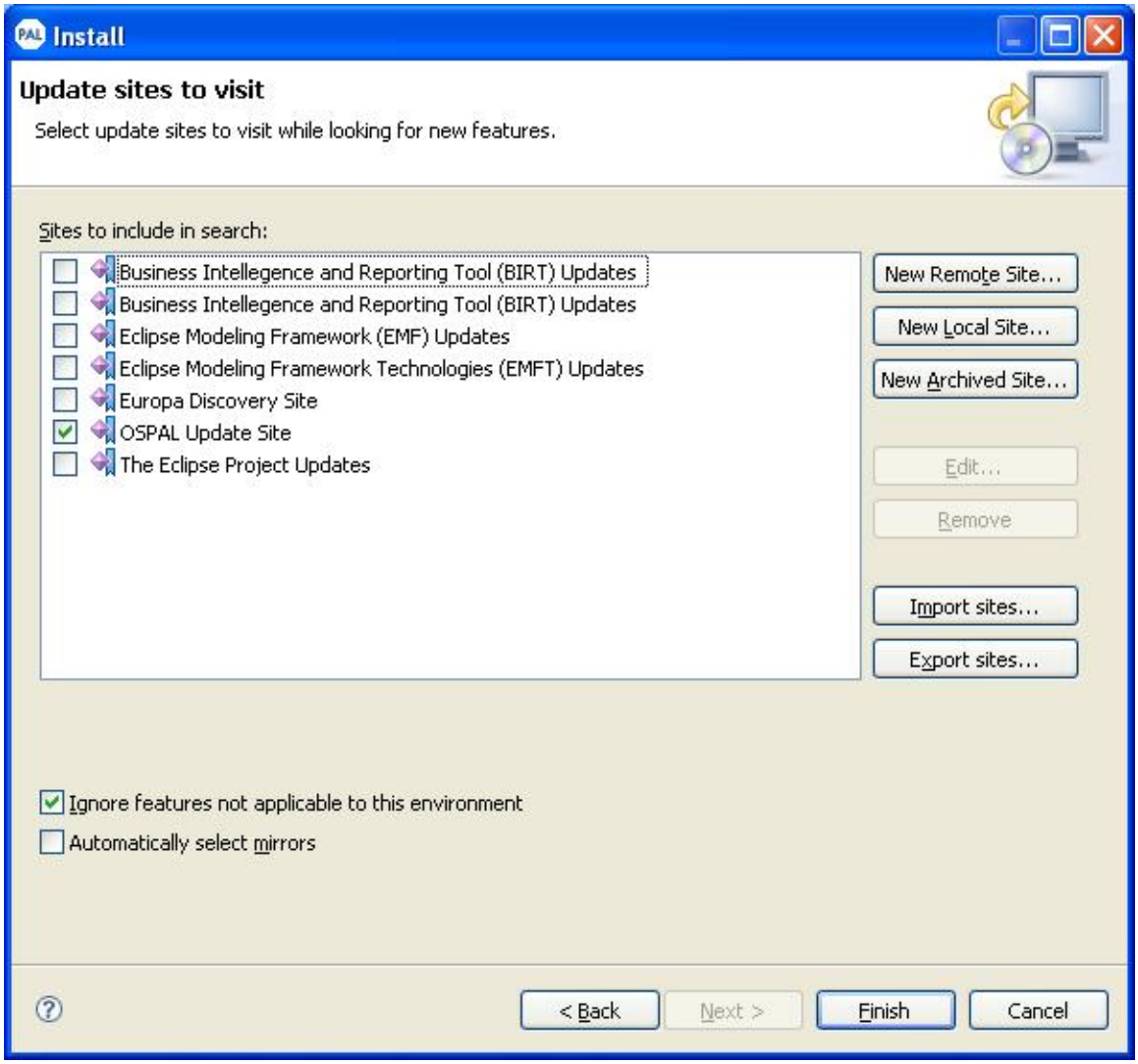
2. Select **Search for new features to install** and click **Next** as shown in Figure 72:

Figure 72: Remote Update Feature Updates



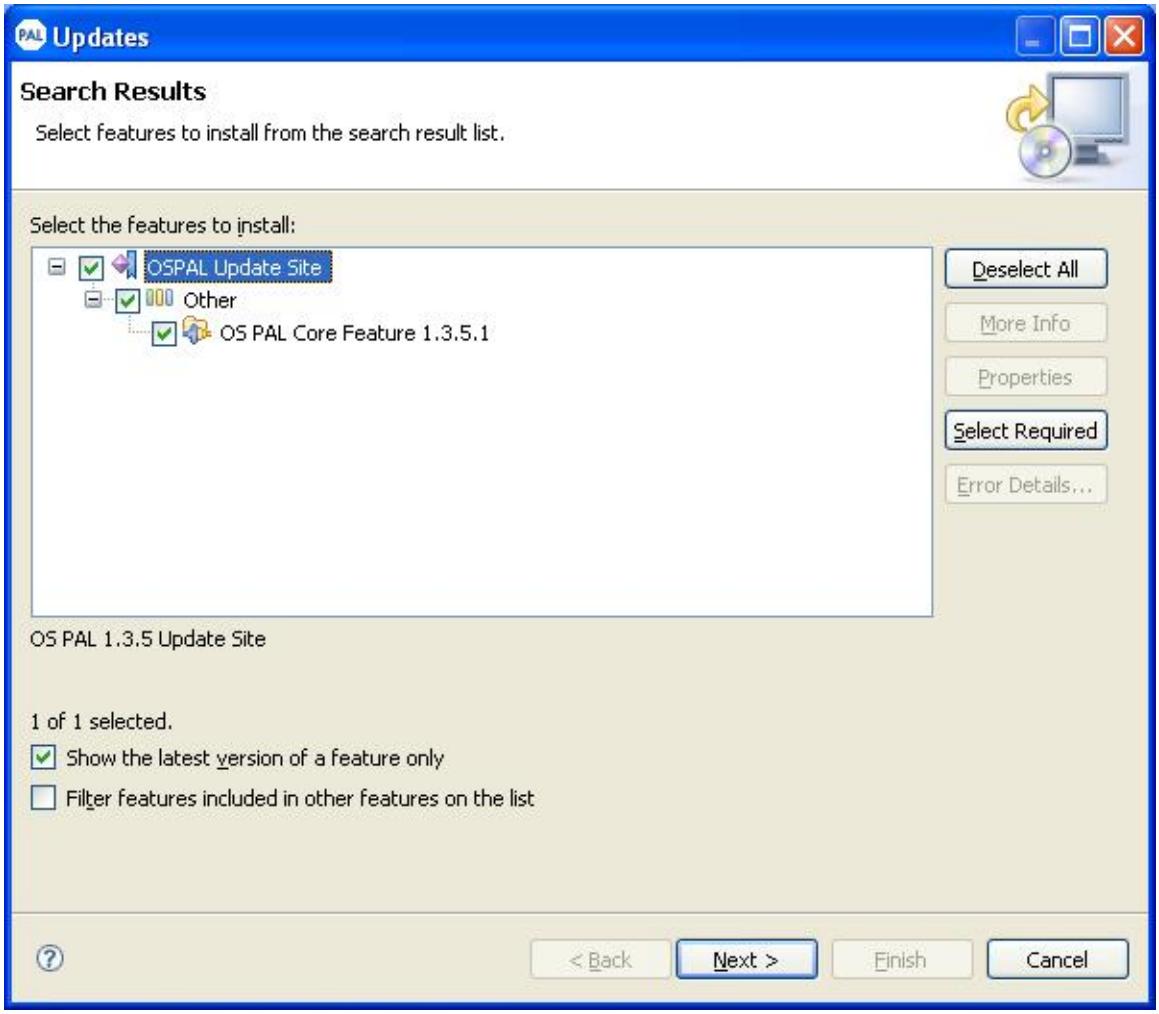
3. On Updates sites to visit window, select the check box next to **OS PAL Update Site** and click **Finish** as shown in Figure 73:

Figure 73: Remote Update Site to Visit



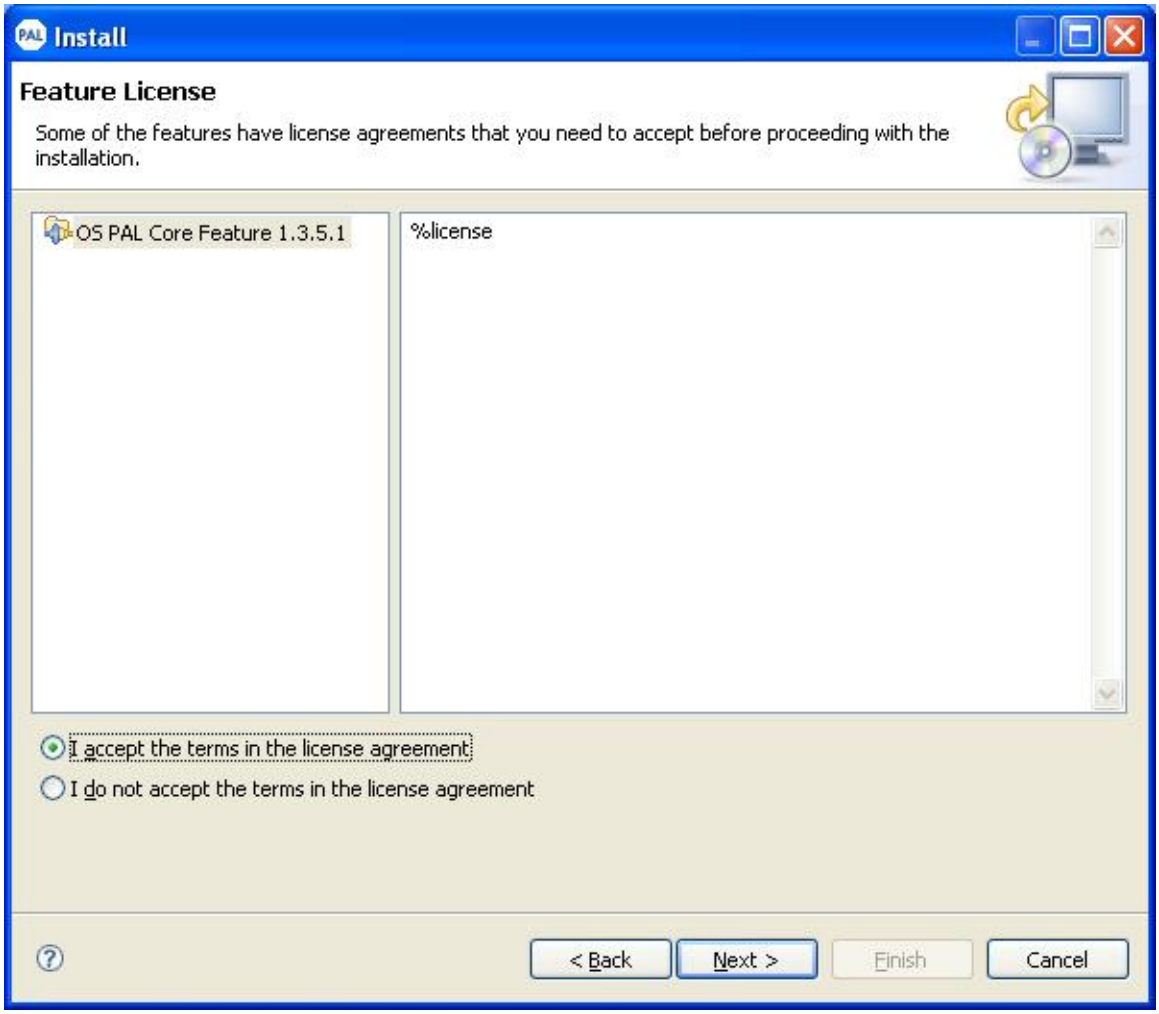
4. On OS PAL Updates Search Results window, select the features under the **OS PAL Update Site** tree parent and click **Next** as shown in Figure 74:Figure 74.

Figure 74: Remote Update Search Results



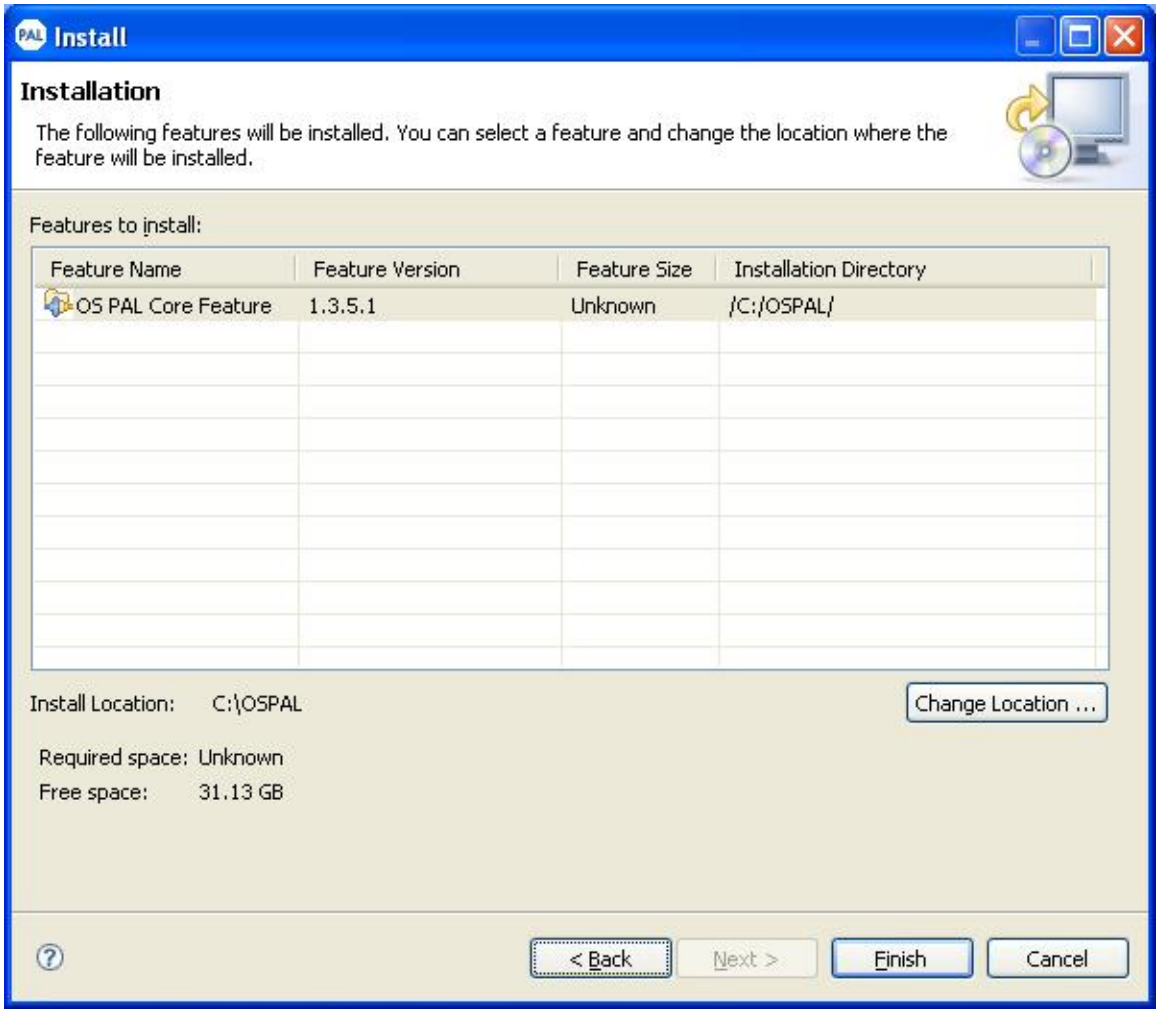
5. On OS PAL Feature License window, select the radio button next to **I accept the terms in the license agreements** and click **Next** as shown in Figure 75.

Figure 75: Remote Update Host Target Feature License



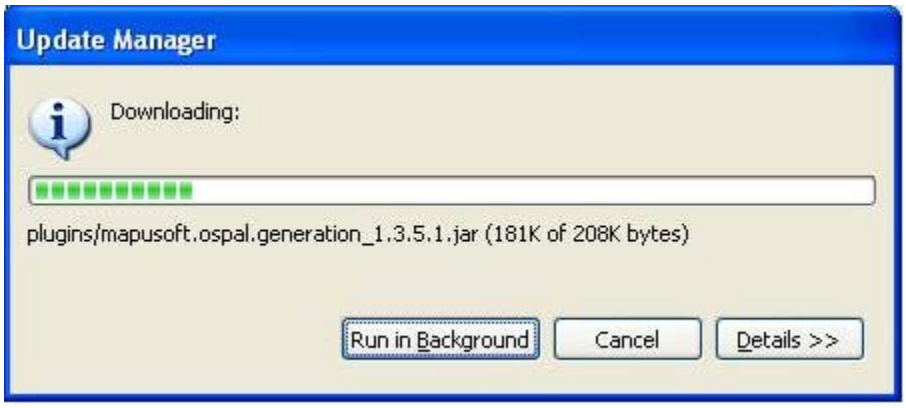
6. On Installation window, you can view the features that are going to be installed and the Installation Directory as shown in Figure 76. **NOTE**: You can change the Installation Directory if you want to, by clicking **Change location** and click **Finish**.

Figure 76: Remote Update Installation Window



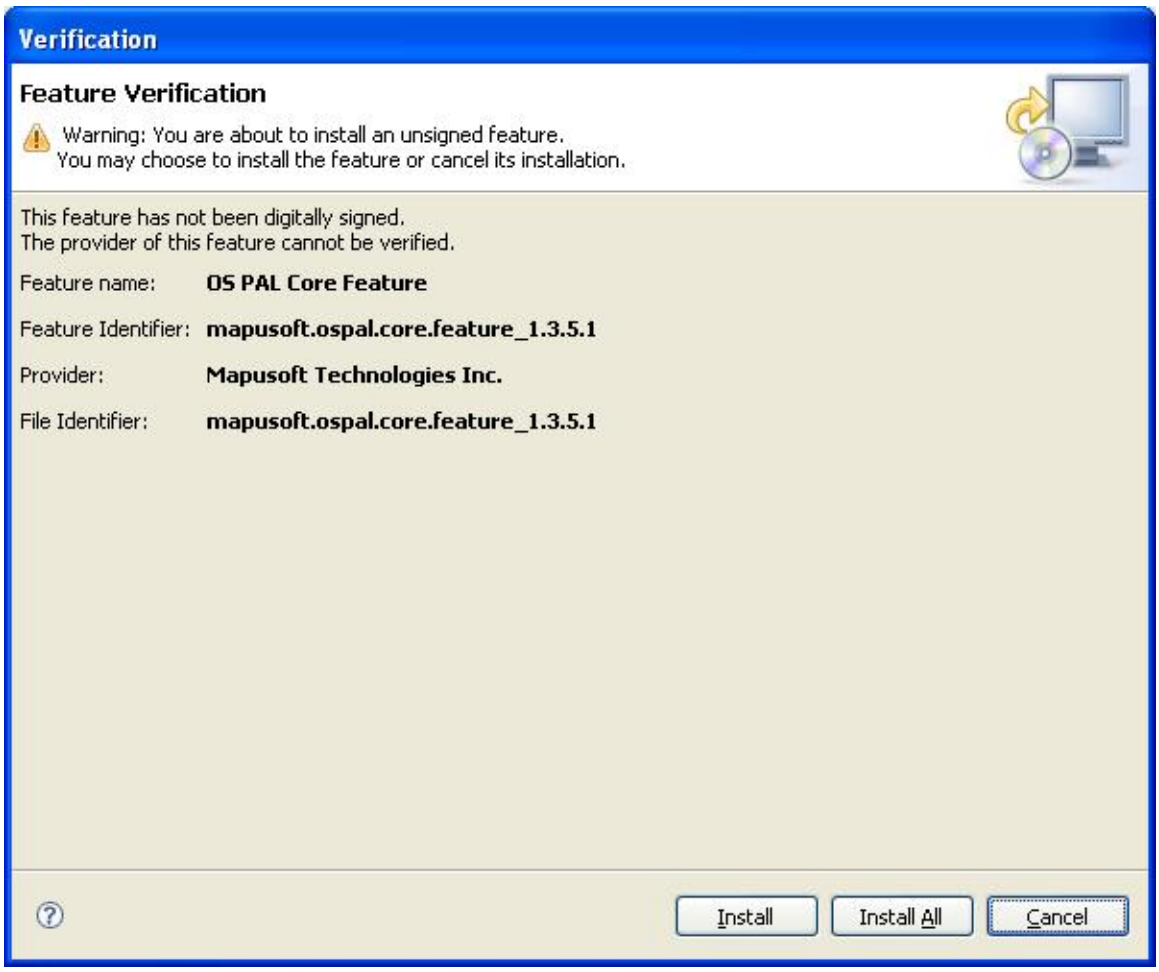
7. On the Update Manager window, you can view the new plugins being downloaded as shown in Figure 77.

Figure 77: Remote Updates Download



8. On Feature Verification window, click **Install All** as shown in Figure 78.

Figure 78: Remote Update Feature Verification

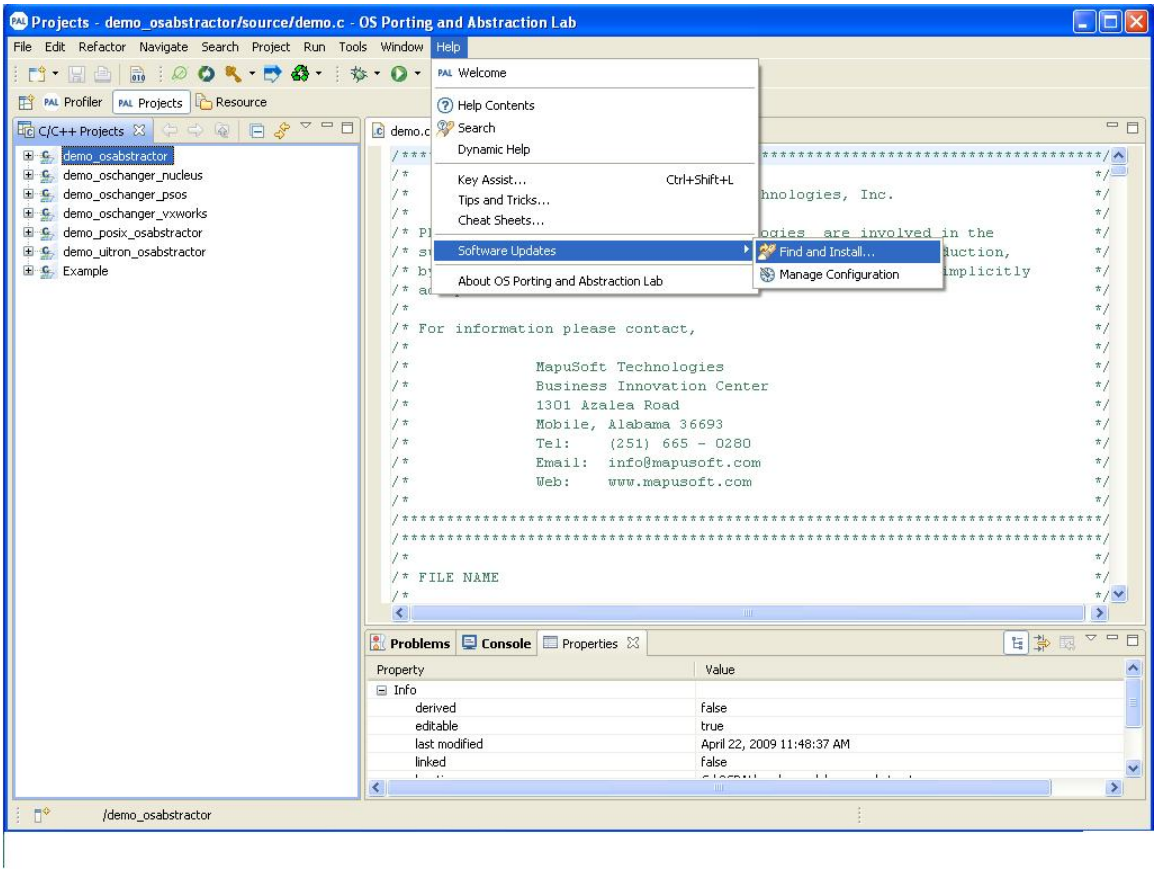


9. Once all the features and plug-ins have been downloaded successfully and their files installed into the product on the local computer, a new configuration that incorporates these features and plug-ins will be formulated. Click **Yes** when asked to exit and restart the Workbench for the changes to take effect. You have now successfully installed new feature updates to your OS PAL using the Remote Update Site.

Updating Software Using Local Update Site

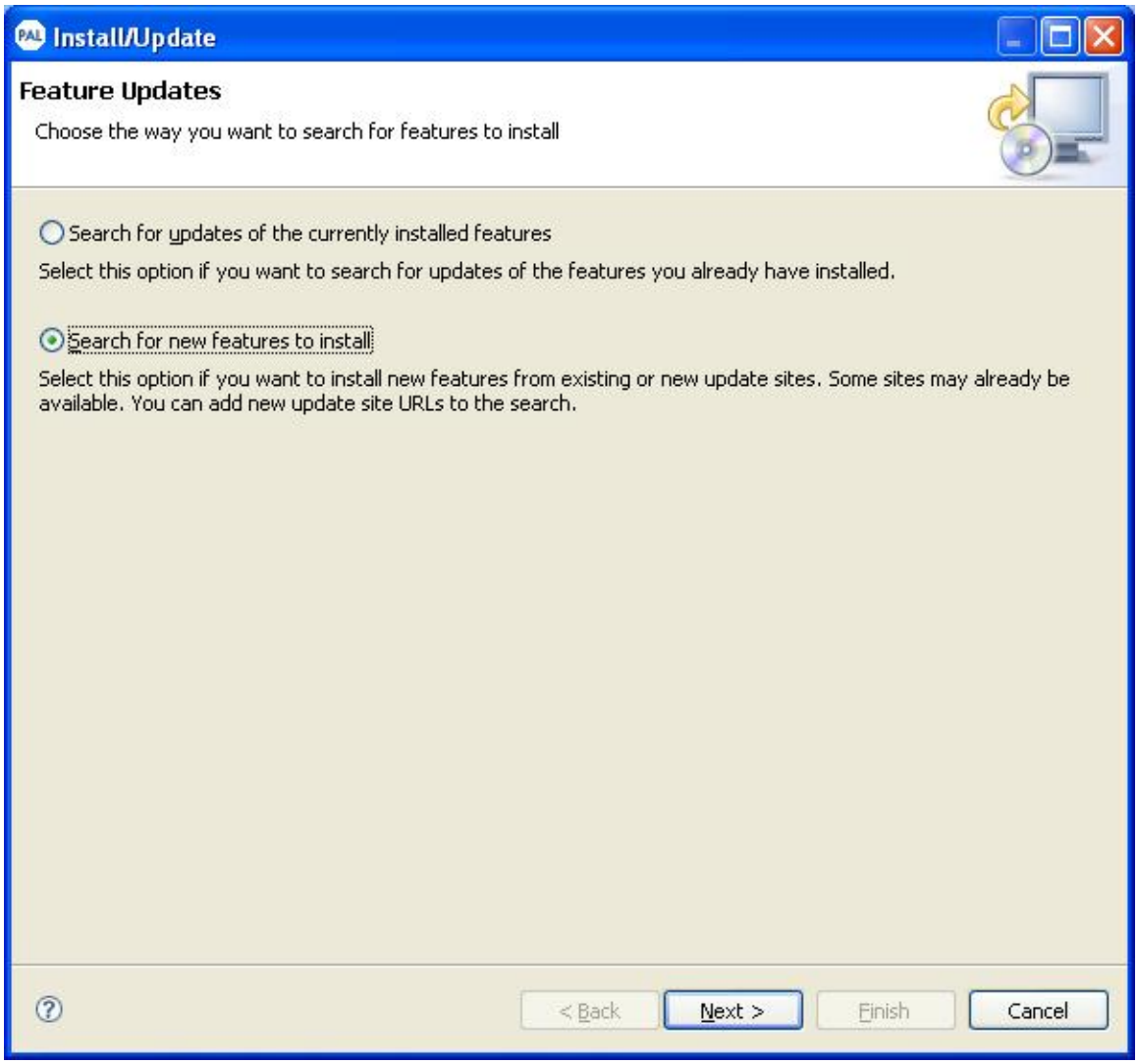
1. From OS PAL main window, select **Help > Software Updates > Find and Install** as shown in Figure 79.

Figure 79: OS PAL Software Updates



2. Select **Search for new features to install** and click **Next** as shown in Figure 80.

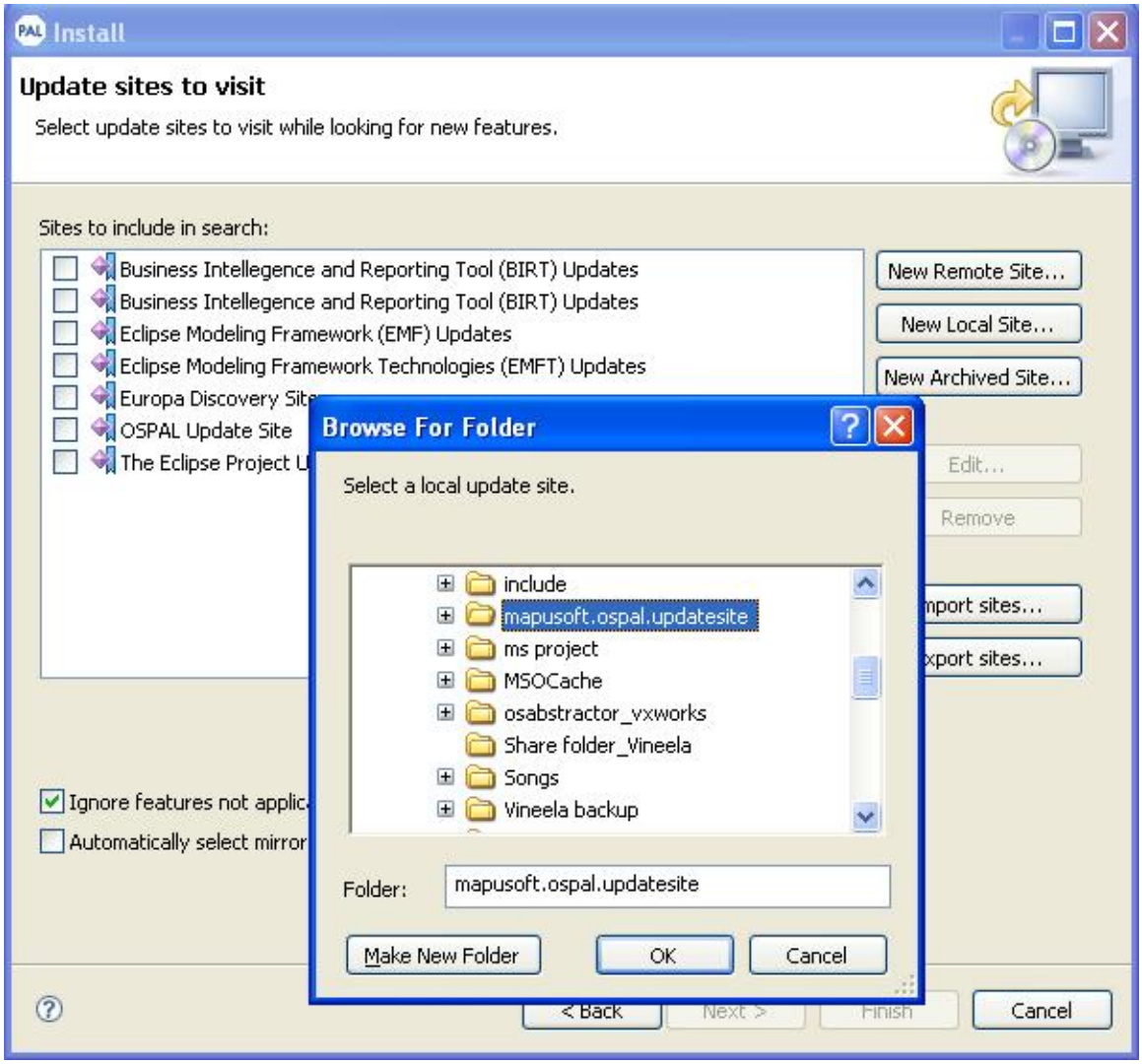
Figure 80: Installing New Feature Updates



3. On Updates sites to visit window, select **New Local Site** and browse for the folder provided by MapuSoft, named as *mapusoft.ospal.updatesite* and click **OK** as shown in Figure 81.

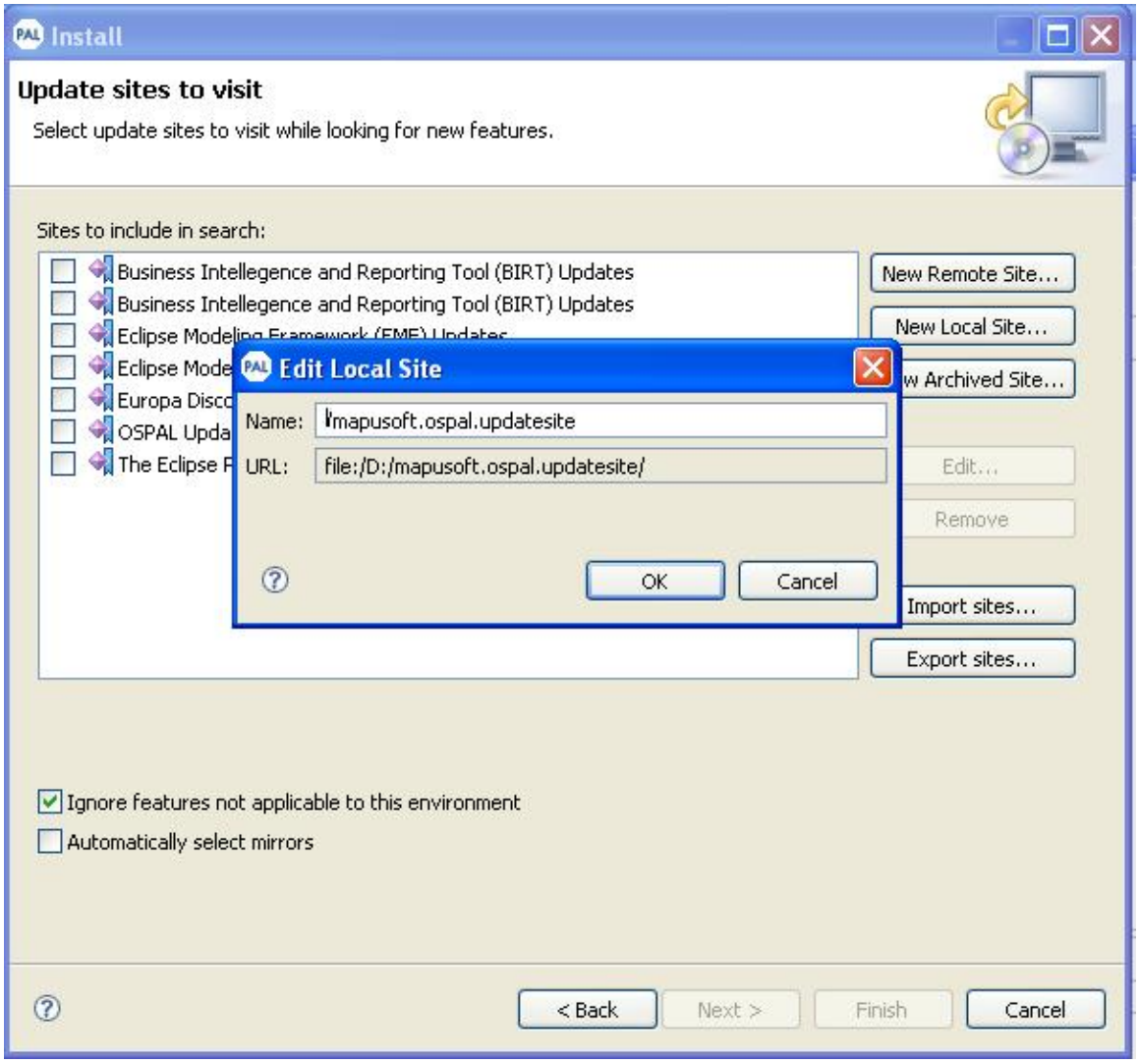
   NOTE: If the system does not allow you to give the same site name, select the previous *updatesite* folder from the list and click **Remove**. Or, you can also save the Updatesite folder in any other location on your local disk.

Figure 81: Installing Updates by Using Local Update Site



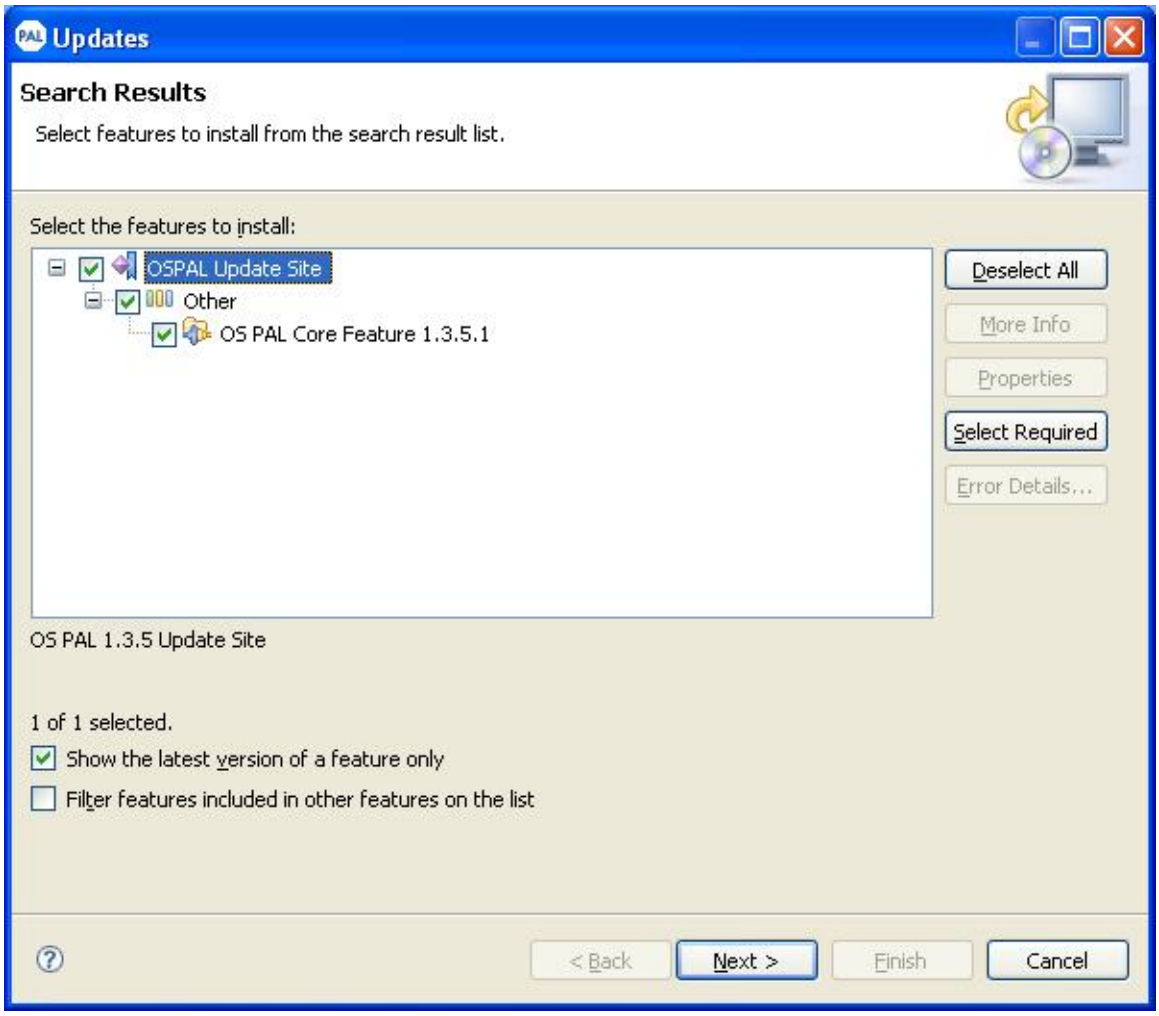
4. On Edit Local Site pop up window, next to **Name** text box, provide a different name and click **OK**. The name can be any name that is not already present on the list as shown in Figure 82 and click **Finish.**

Figure 82: Update Sites to Visit



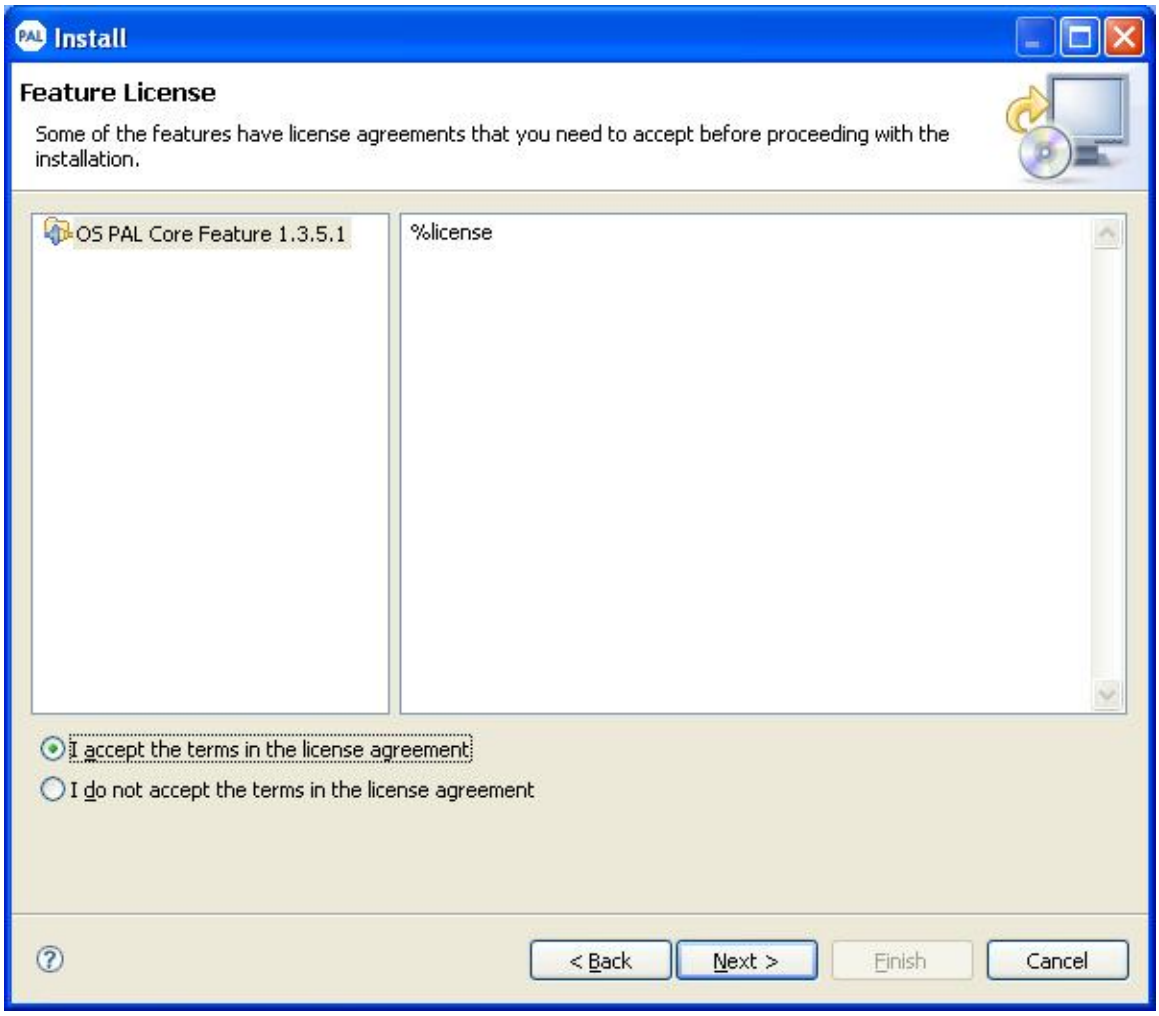
5. On OS PAL Updates Search Results window, select the features under **Others** tree parent and click **Next** as shown in Figure 83.

Figure 83: Search Results



6. On OS PAL Feature License window, select the radio button next to **I accept the terms in the license agreements** and click **Next** as shown in Figure 84.

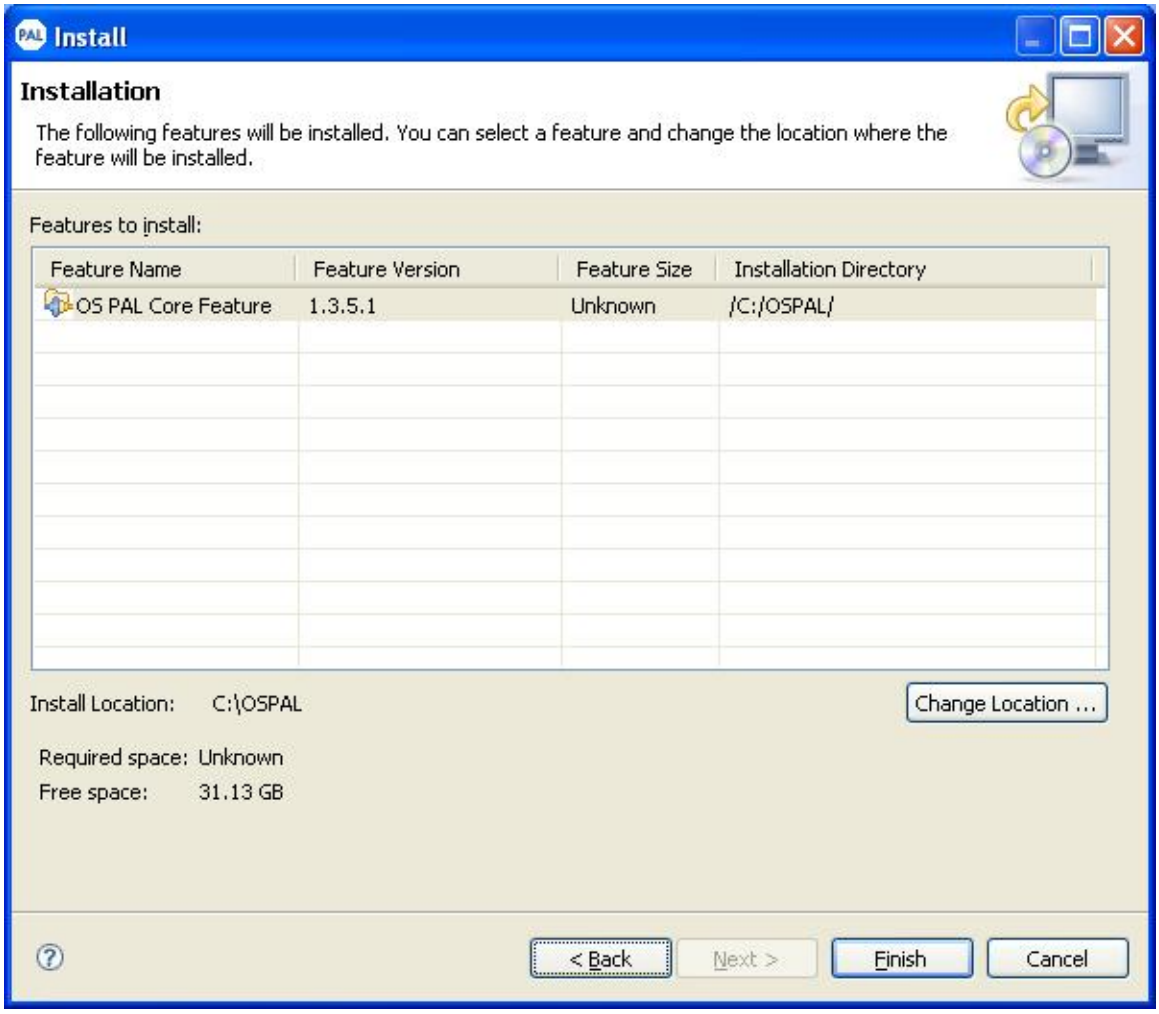
Figure 84: Feature License Agreement



7. On Installation window, you can view the features that are going to be installed and the Installation Directory as shown in Figure 85 and click **Finish**.

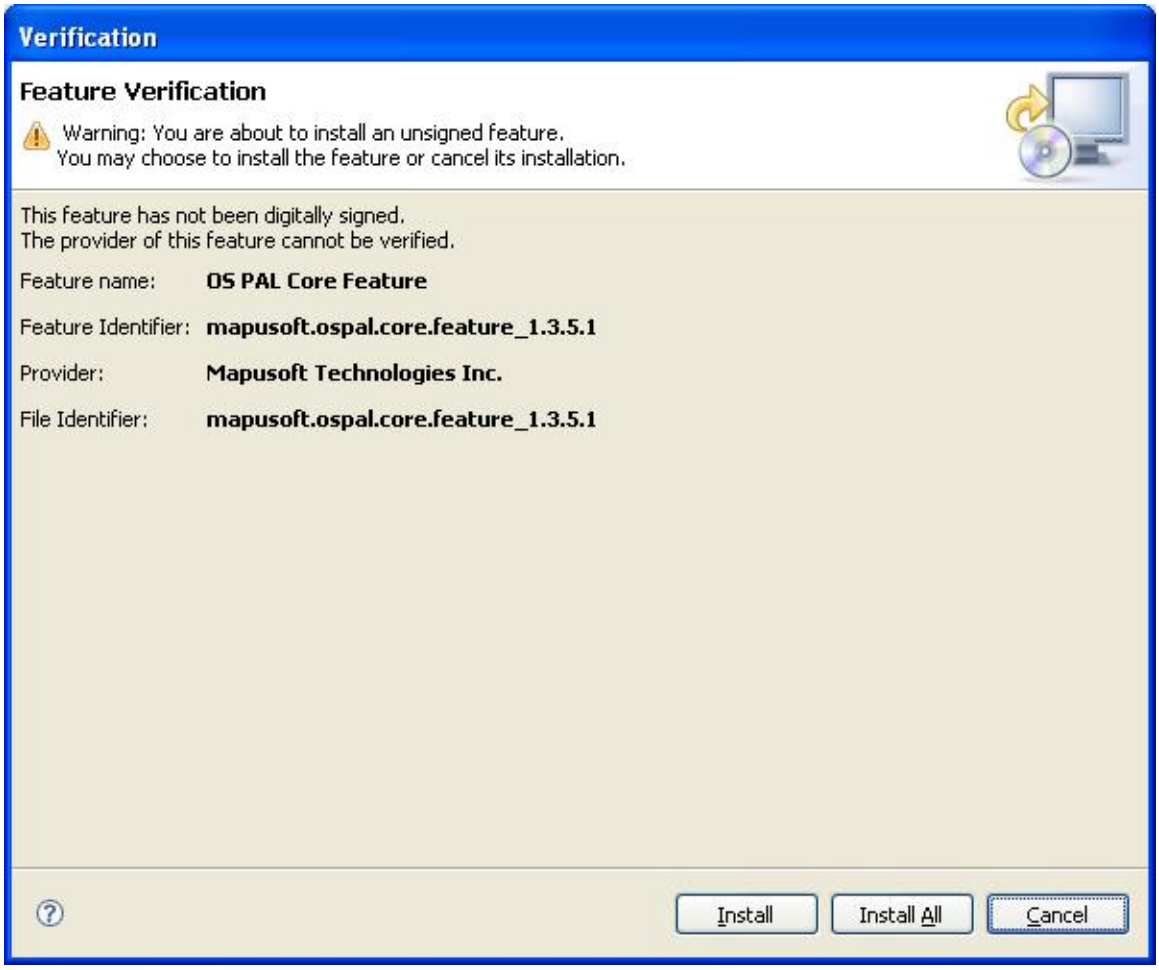
   NOTE: You can change the Installation Directory if you want to, by clicking Change location.

Figure 85: Feature Installation



8. On Feature Verification window, click **Install All** as shown in Figure 86.

Figure 86: Feature Verification



9. Once the new feature and plug-ins have been downloaded successfully and their files installed into the product on the local computer, a new configuration that incorporates these features and plug-ins will be formulated. Click **Yes** when asked to exit and restart the workbench for the changes to take effect.
10. You have now successfully installed new feature updates to your OS PAL.

Creating a Standalone OS Changer/ OS Abstractor Package

NOTE: This feature requires a standalone generation license. Click http://mapusoft.com/contact/ to send a request to receive licenses and documentation.

OS PAL can also create standalone packages to complete the porting and development outside of OS PAL with your own tools and environment.

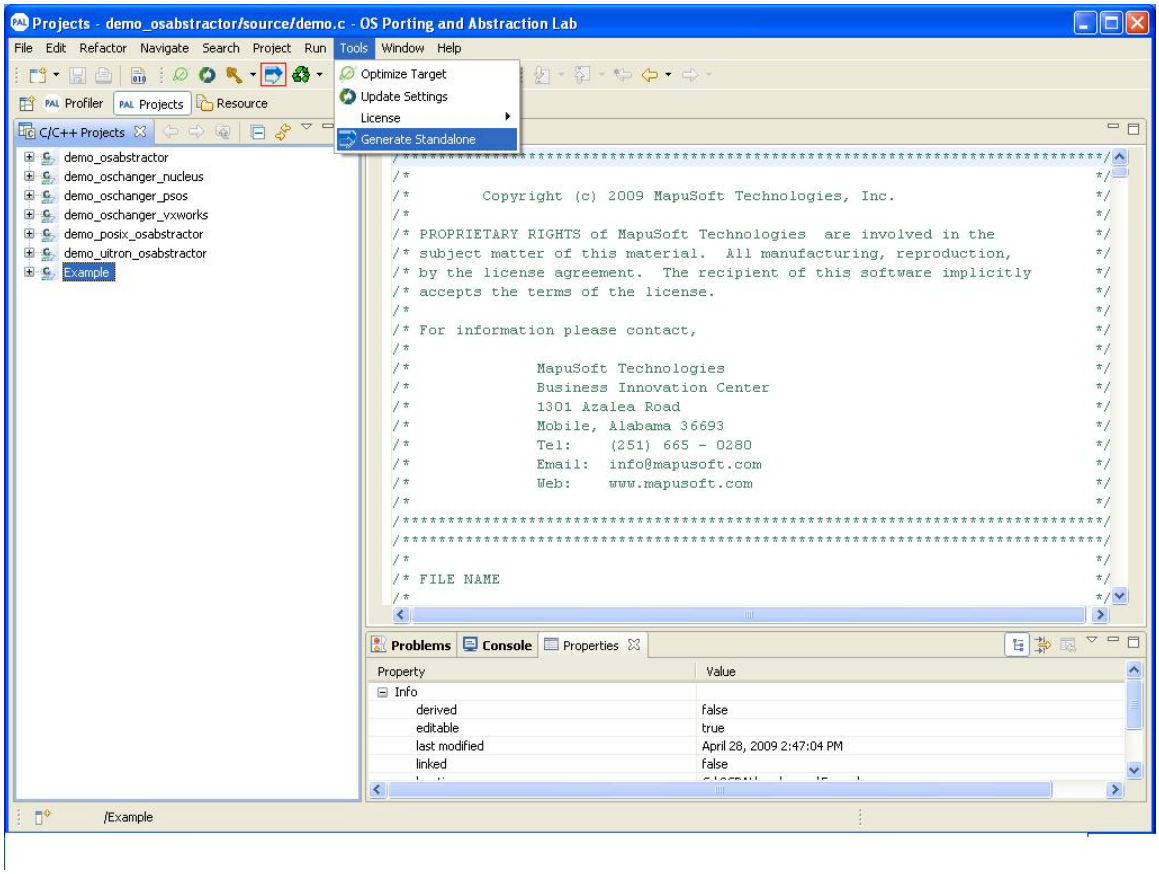
To create a Standalone OS Changer/OS Abstractor Package, follow the steps:

1. Create the Standalone Project

To create a standalone project:

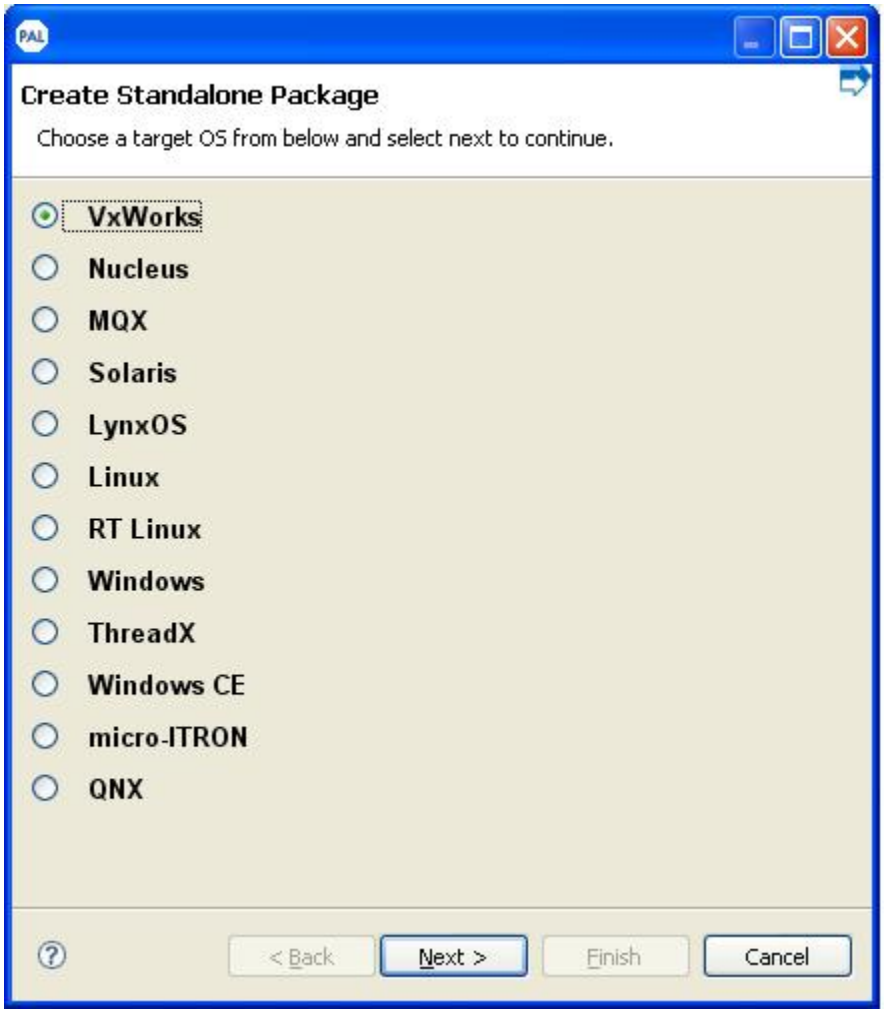
a) From OS PAL main menu, click **Standalone Project** button on the tool bar as highlighted in Figure 87.

Figure 87: Creating Standalone Projects



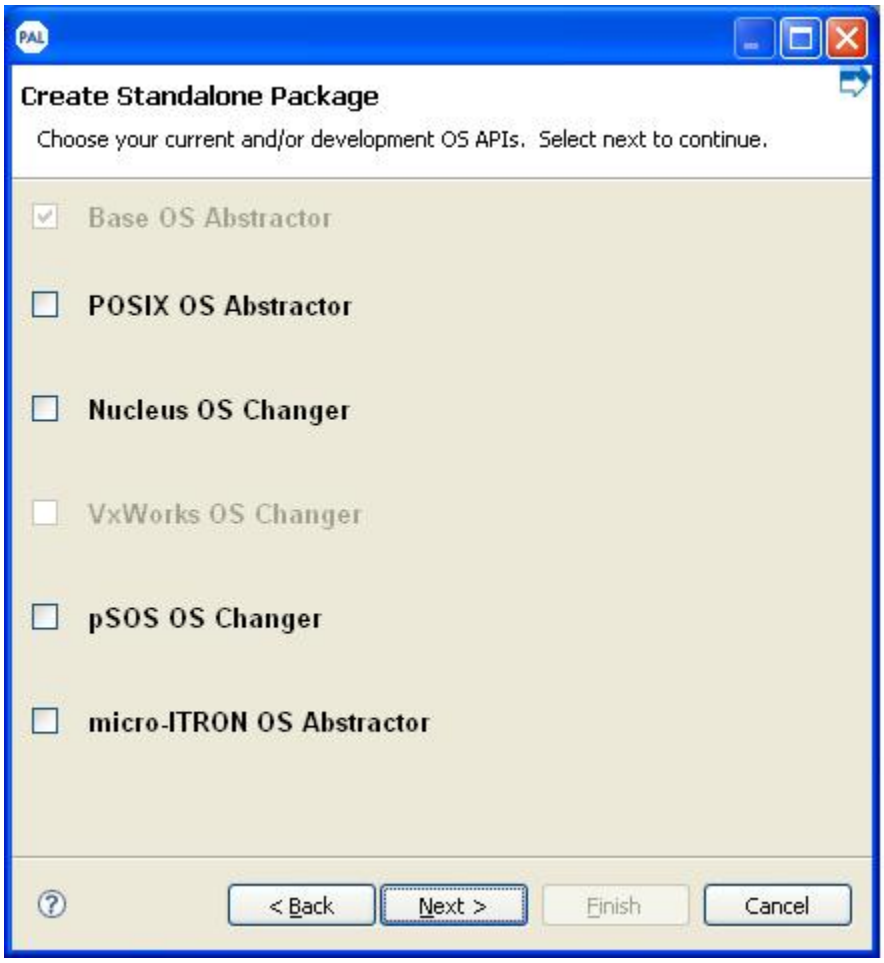
b) Select the Target OS from the list and click **Next** as shown in Figure 88.

Figure 88: Select Target OS



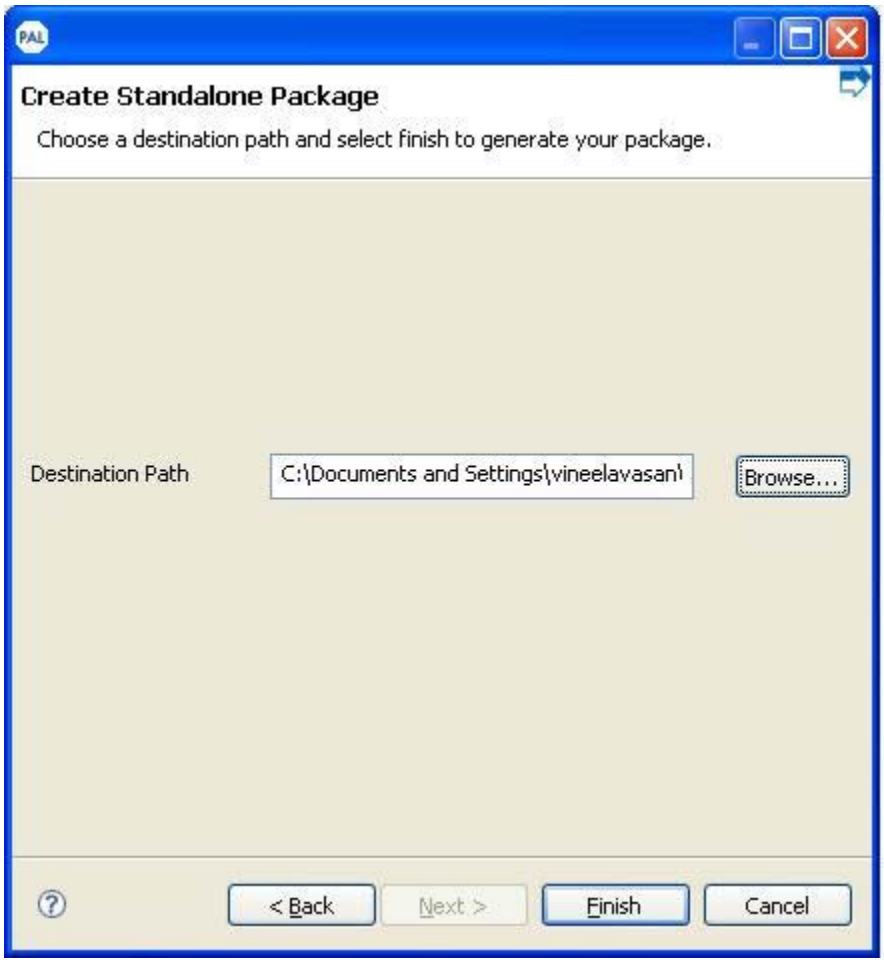
c) Select the OS Changer or OS Abstractor products needed to create the standalone project and click **Next** as shown in Figure 89.

Figure 89: Select OS Changer or OS Abstractor



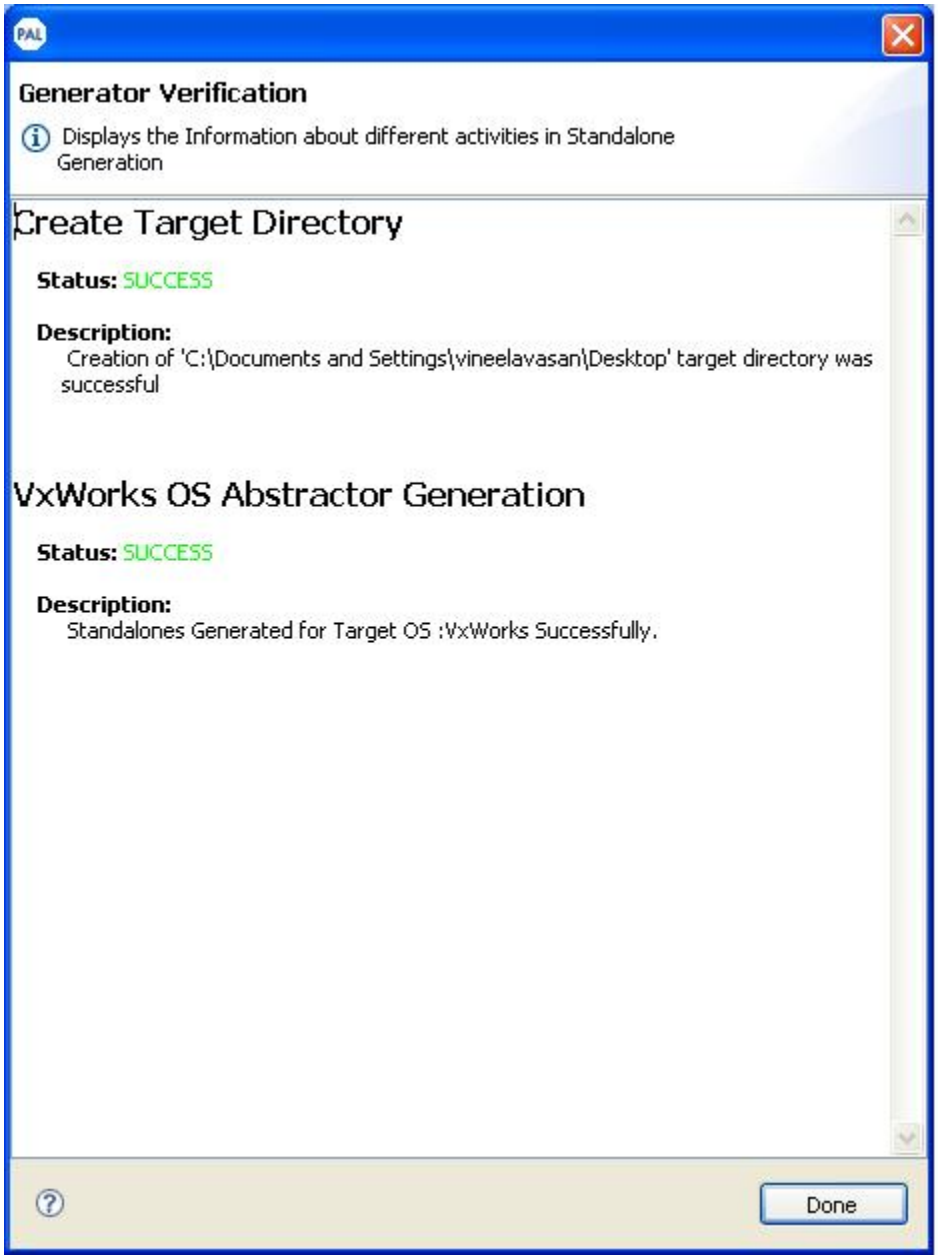
d) Select the destination path to save the generated package and click **Finish** as shown in Figure 90.

Figure 90: Select Destination Path



e) The successful standalone generation is shown in Figure 91.

Figure 91: Standalone Generation



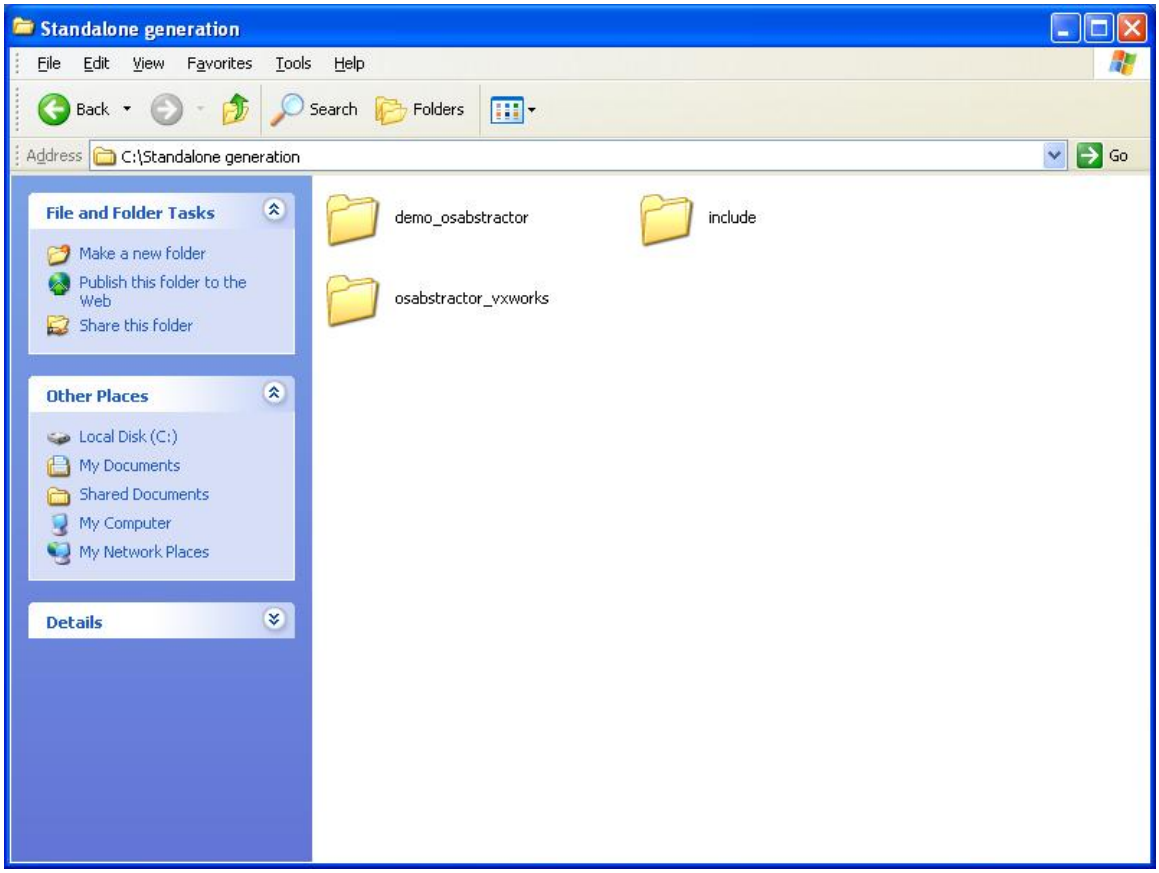
2. Show the Standalone Package Created by OS PAL

The standalone package includes libraries with full source code to manually link into the applications. Once the application is recompiled with MapuSoft's products it will run on the new OS.

To view the generated standalone package:

a) Open the folder created by OS PAL.
b) Click on the folder to view the files included in the package as shown in Figure 92, which include:
   - A sample application
   - Libraries containing the MapuSoft products needed to run their application on the new OS

Figure 92: Standalone Generation Folder



PORTING AN APPLICATION WITH OS PAL

NOTE: This feature requires a license and documentation.
Click http://mapusoft.com/contact/ to send a request to receive licenses and documentation.

If you prefer not to use the OS PAL framework for your porting, MapuSoft can also provide the standalone OS Changer, micro-ITRON OS Abstractor and POSIX OS Abstractor libraries which then can be used directly under your target development environment. MapuSoft can also provide you project files associated with building the library and a sample application for your target environment.

OS PAL allows you to port POSIX, VxWorks, Nucleus, micro-ITRON, and pSOS embedded applications from one OS to another.

Porting is a two step process:

1. Use the OS PAL target code optimizer to move the application to the target OS environment.
2. Port the legacy application into the OS PAL host environment
3. Importing manually

Porting Legacy Applications Using OS PAL

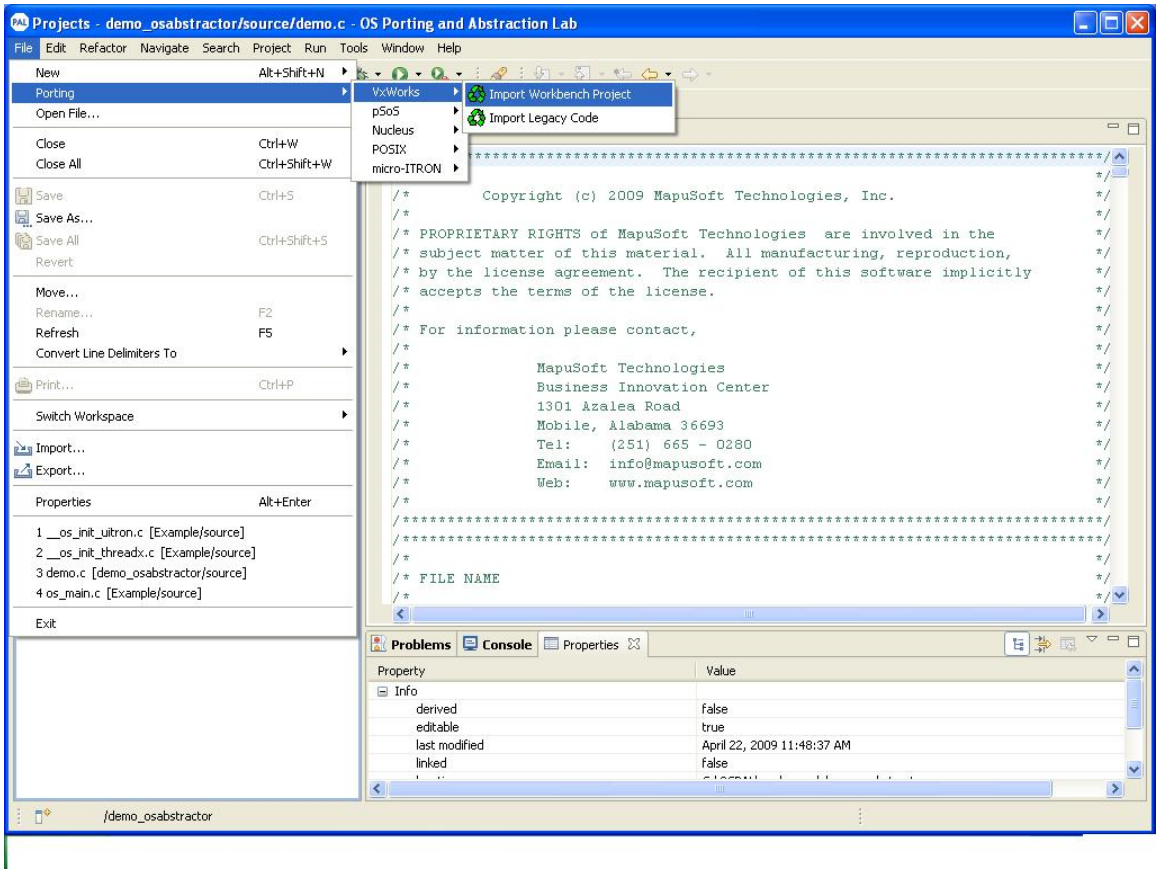
Legacy applications porting into the OS PAL host environment can be done in three different ways:

Method 1– Importing a VxWorks Project

NOTE: This feature requires a license. Click http://mapusoft.com/downloads/ospal-evaluation/ to request an evaluation license.

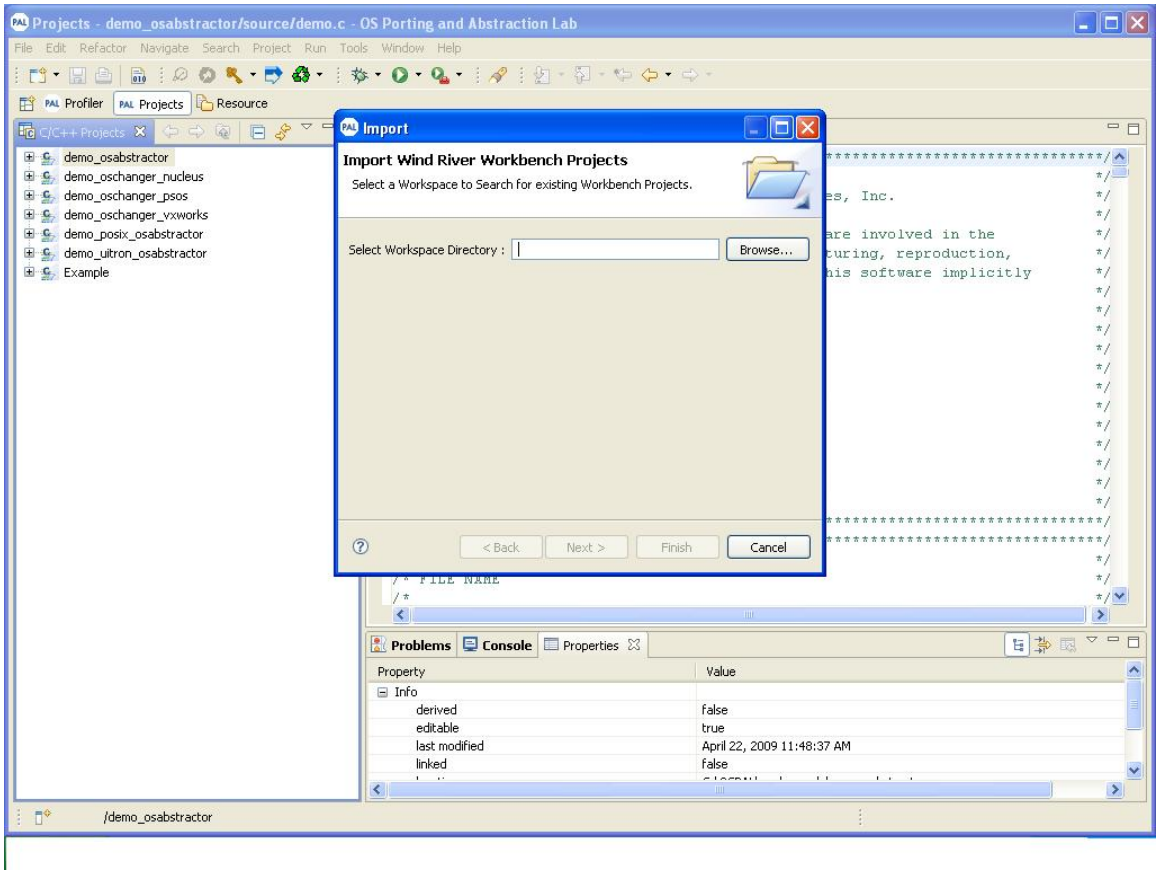
1. From OS PAL main window, select any project under C/C++ **Projects** tab on the left pane.
2. Select **File > Porting > VxWorks > Import Workbench Project** as shown in Figure 93. You can also click on the Porting icon from the task bar.

Figure 93: Importing a VxWorks Project in OS PAL



3. On OS PAL Import window, select a workspace directory to search for existing workbench projects by clicking on **Browse** button next to the text box, and click **Next** as shown in Figure 94.

Figure 94: OS PAL C Project Wizard Window



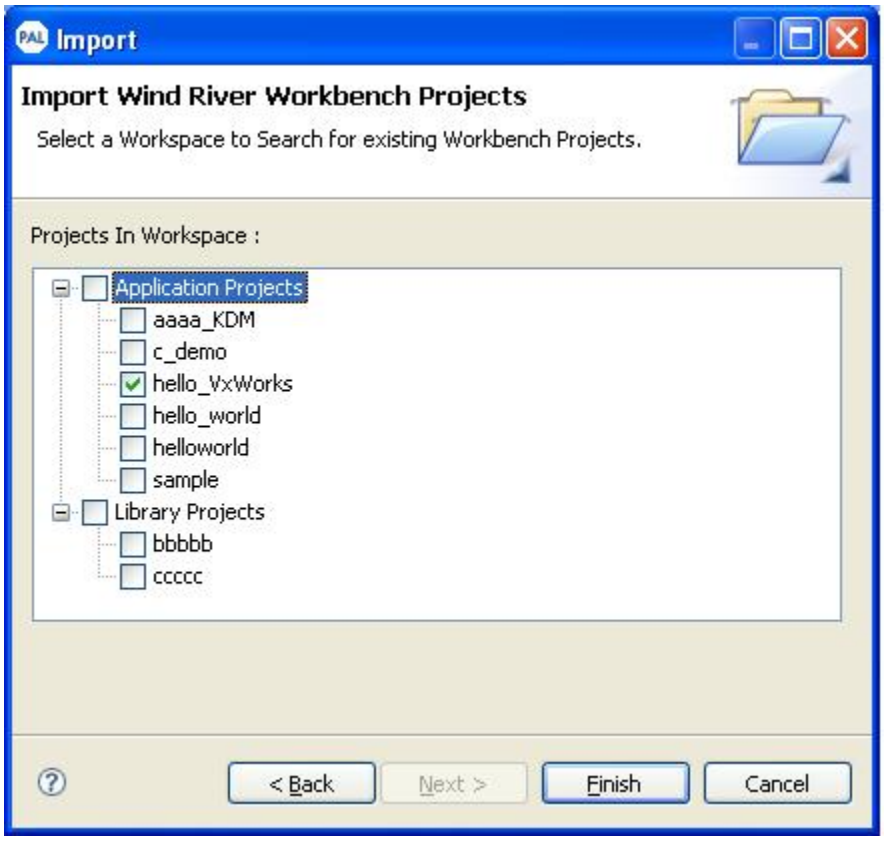
4. In the Projects in Workspace window, the projects list is displayed in a checkbox Tree. Applications and Libraries are separated into respective categories.

The following are the project types:

- **Application Projects** - Provides an executable application. This project type folder contains three templates. The makefile for the Executable project type is automatically created by the CDT.
- **Library Projects**- An executable module that is compiled and linked separately. When you create a project that uses a shared library (libxx.so), you define your shared library's project as a Project Reference for your application. The makefile for this project type is automatically created by the CDT.

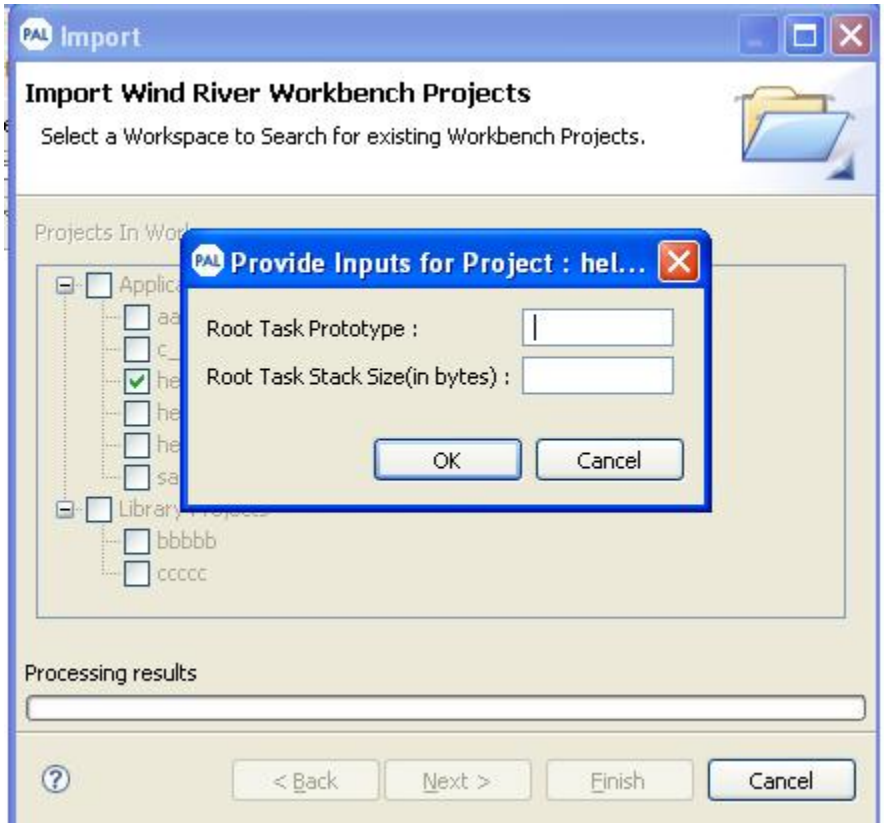
5. Select or deselect any one or all of the projects by selecting the check box next to the project name and click **Finish** to import the project as shown in Figure 95.

Figure 95: Selecting the Application Window



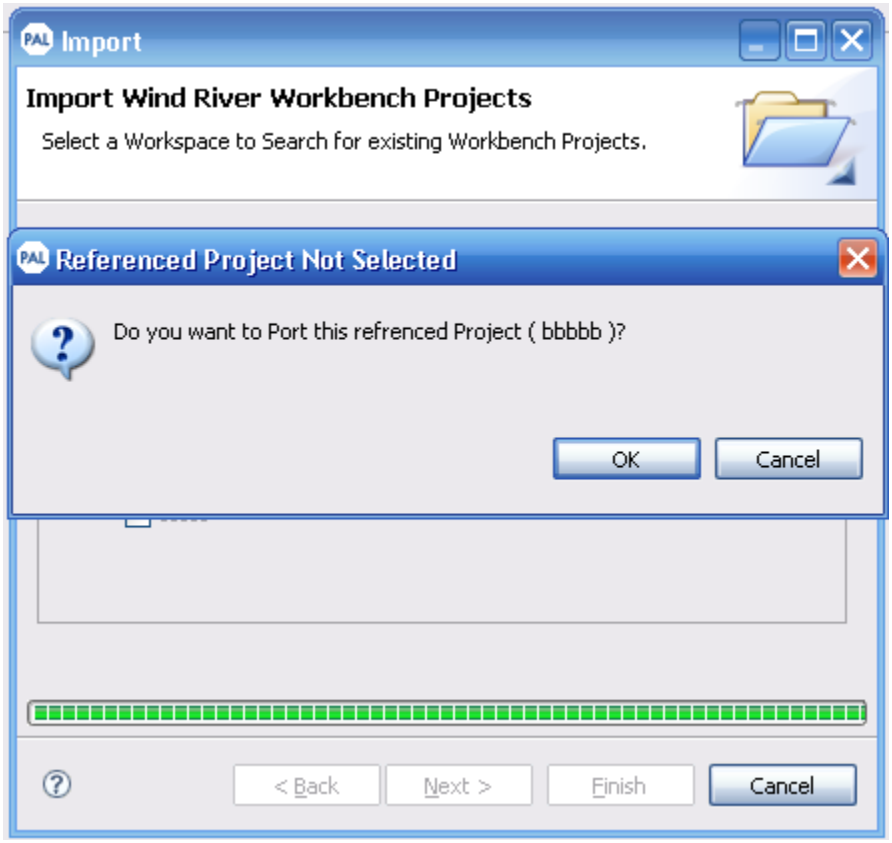
6. If you select any application type project, provide the inputs for the project and click **OK** as shown in Figure 96. If you do not want to provide the inputs, you can just click **Cancel.**

Figure 96: Provide Inputs for Projects Window



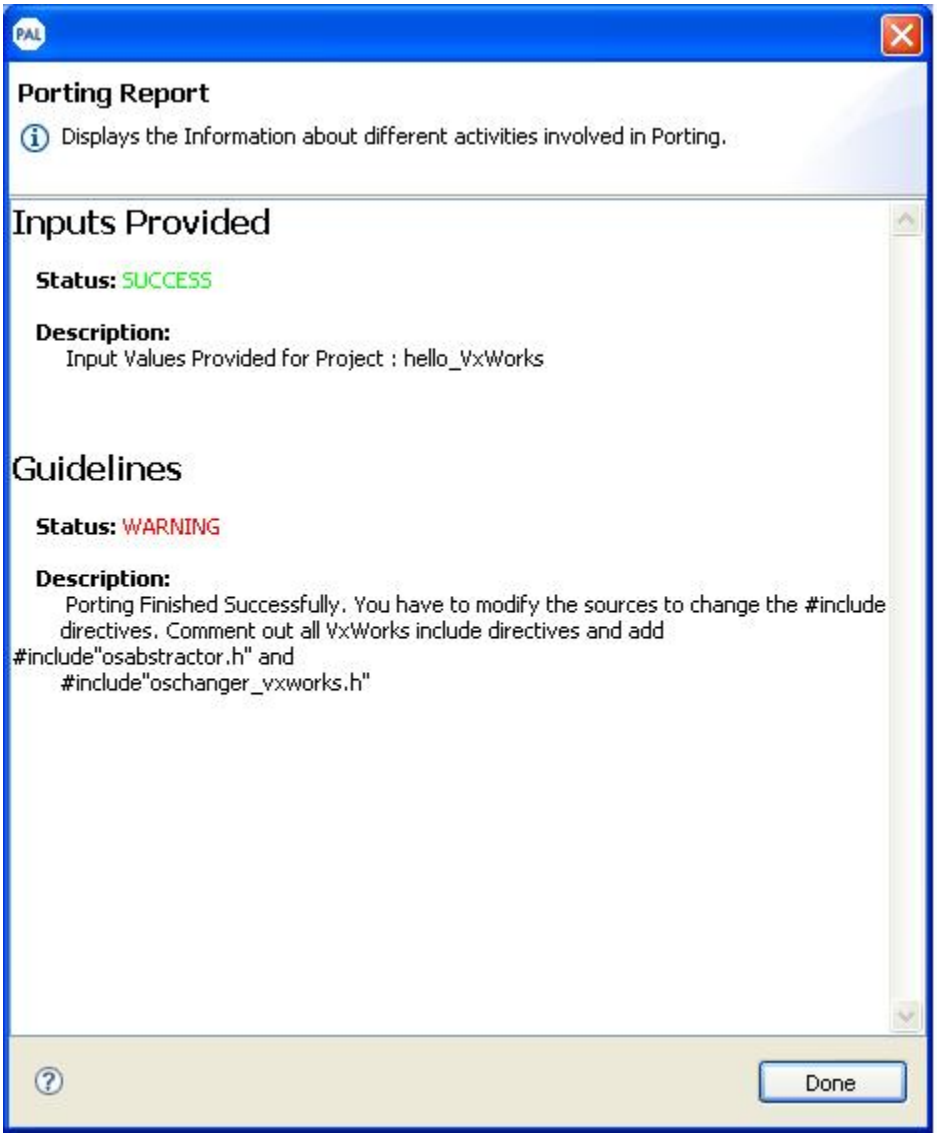
7. If you select an application project and if it contains any referenced projects not selected by you, then a Confirmation dialogue box is displayed on your screen to ask if you want to port the project or not as shown in Figure 97. If you want to port, click **OK**. You can see the porting processing results on the window.

Figure 97: Referenced Projects Porting Confirmation



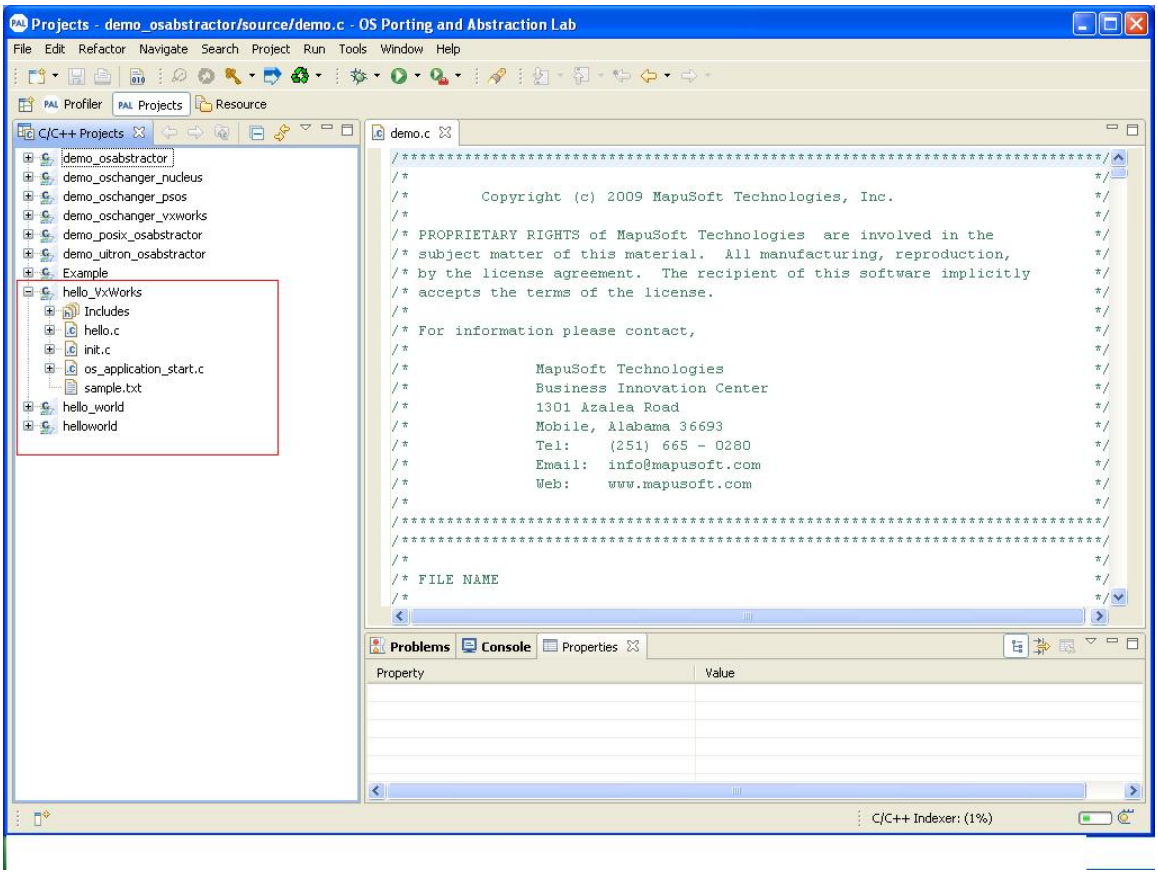
8. After the porting is successfully done, the porting report page is displayed as shown in Figure 98. Click **Done** to complete the process.

Figure 98: Porting Reports Page



9. In OS PAL projects perspective, the ported projects are displayed as shown in the Figure 99.

Figure 99: OS PAL Project Perspective of the Ported Projects



You have successfully imported your VxWorks application to OS PAL.

Method 2–Importing Legacy Code

Porting Legacy Applications using OS PAL Porting Plugin

A description for porting VxWorks and Legacy applications using OSPAL is described with an example here.

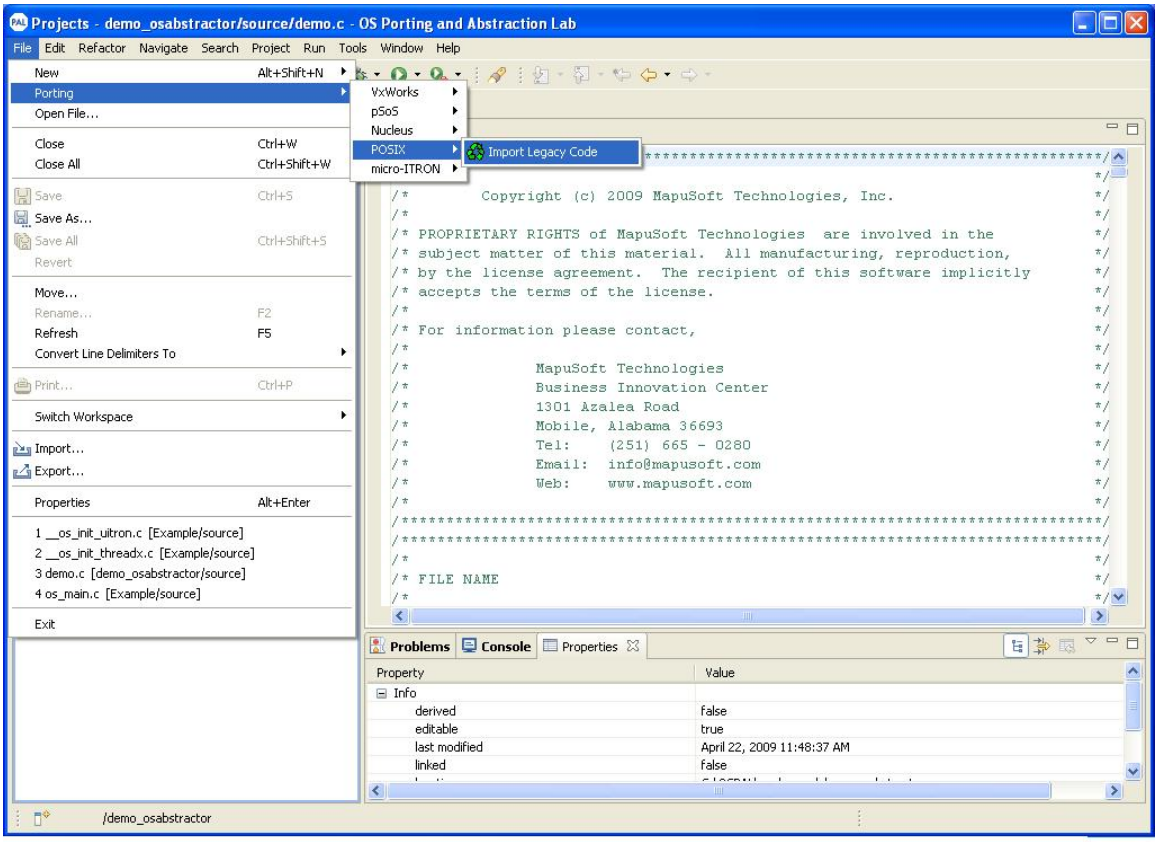
NOTE: This feature requires a license. Click http://mapusoft.com/downloads/ospal-evaluation/ to request an evaluation license.

You can import legacy code for the following compatible codes:

- VxWorks
- pSOS
- Nucleus
- POSIX
- micro-ITRON

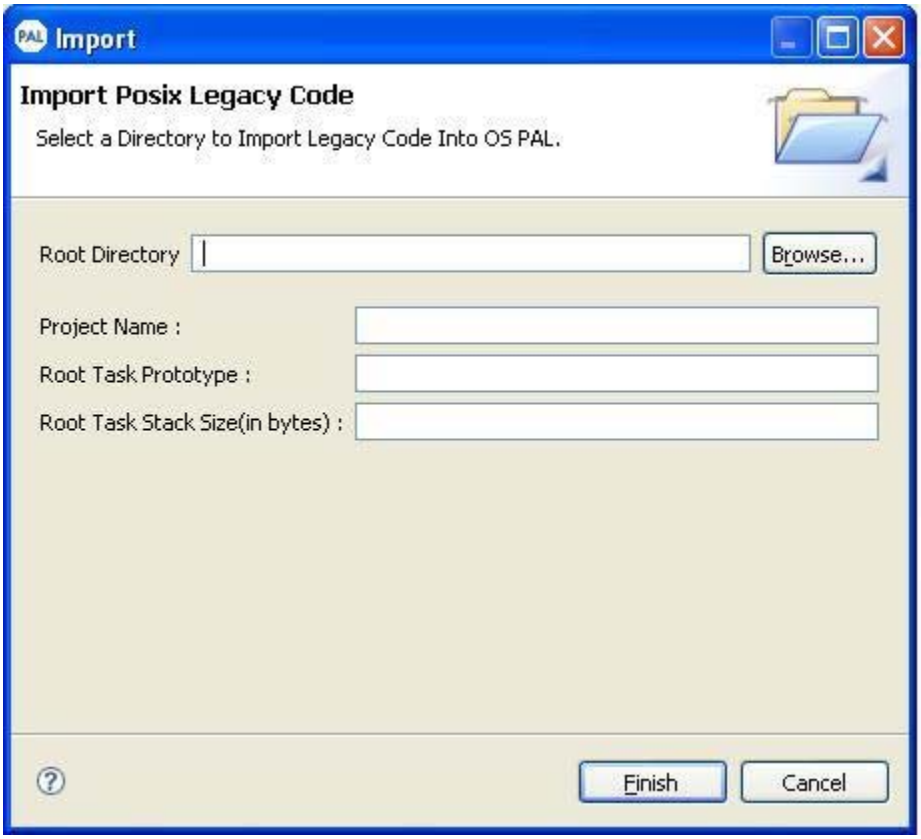
1. Select File > **Porting** > **VxWorks/pSOS/Nucleus/POSIX/micro-ITRON** > **Import Legacy Code** as shown in Figure 100. You can also click on the Porting icon from the task bar. (For instance: importing POSIX Legacy Code).

Figure 100: Importing Legacy Code in OS PAL



2. On OS PAL Import window, select the root directory from where you want to import the legacy code by clicking on **Browse** button next to the text box, and click **Next** as shown in Figure 101.

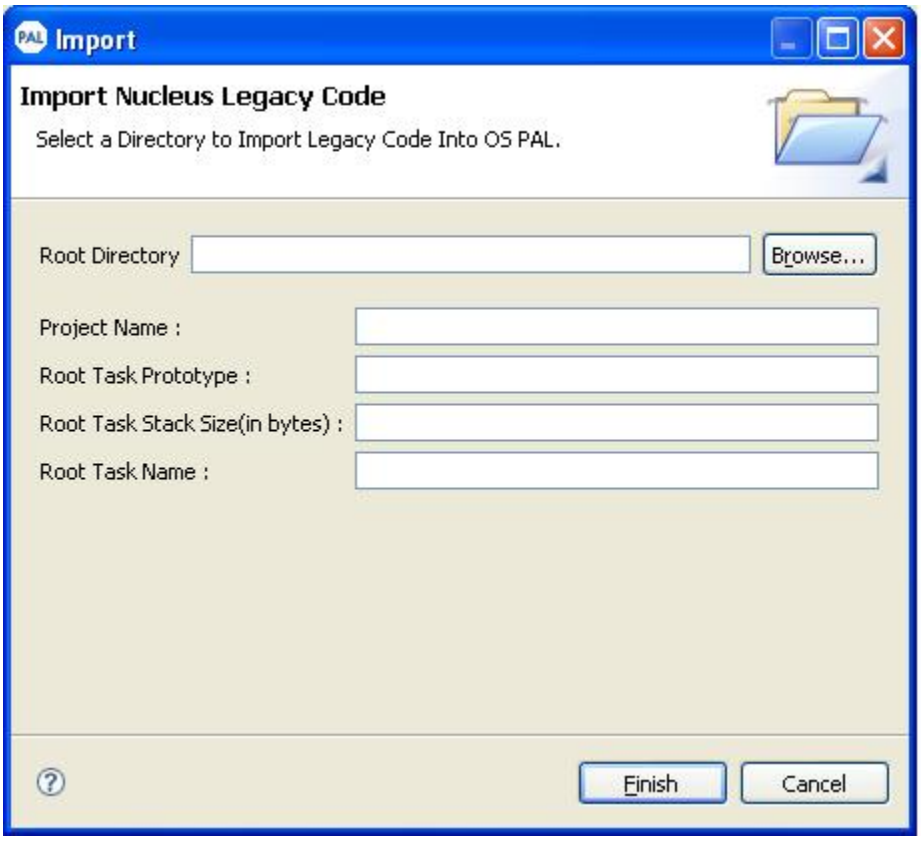
Figure 101: OS PAL Import Legacy Code Window



3. Enter the project name for which you want to import the legacy code in the **Project Name** text box.
4. Enter the root task prototype, next to **Root Task Prototype** text box.
5. Enter the root task stack size, next to the **Root Task Stack Size** text box. The value should be in bytes.
6. Click **Finish** to complete the importing of legacy code into OS PAL.

7. If you are importing a **Nucleus Legacy Code**, enter also the root task name next to the **Root Task Name** text box as shown in Figure 102.

Figure 102: Importing Nucleus Legacy Code window



You have successfully imported the legacy code and a project with your given project name is created in the current workspace.

Method 3– Manually Porting Legacy Applications using OS PAL Import Feature

Step 1: In your source code, remove references to the original OS include files and use *osabstractor.h* and the header files of your ported legacy API instead.

1) Remove the original OS specific initialization code and use OS_Application_Init function call instead (refer to the OS Abstractor reference manual).
2) Create an OS PAL project for your legacy application and select the legacy API that your application will need (Eg: to port VxWorks application, you need to check the "Include VxWorks OS Changer API's").
3) If your application uses any APIs that are not supported under OS PAL, re-write the code using OS Abstractor APIs.
4) Import your legacy application into the new project.
5) Compile and link your application and resolve all compiler and linker errors.
6) Run or debug your application under OS PAL host in an x86 environment. You should rewrite/replace any hardware specific code in your application for this step.

Step 2: Moving from OS PAL Host to target using OS PAL target code optimizer:

1) Generate the code for your target OS using the OS PAL target code optimizer.
2) Using cross-compiler compile, link, and download the OS PAL generated code to your target.
3) Port low level drivers and hardware interrupt code as required (refer to OS Abstractor I/O and device driver APIs sections in the reference manual).
4) Resolve any run time errors.

Revision History

Document Title: OS PAL User Guide in MS Word
Release Number: 1.3.5

Release	Revision	Orig. of Change	Description of Change
1.3.5	0.1	Vineela	• Included MapuSoft logo on the first page • Included the Revision history table at the end of the doc • Updated the API Profiling feature • Updated porting Legacy Application section • Updated UITRON with micro-ITRON • Updated all the screen shots • Added generating Timing Report content