

---

# Installing and Using the Platform Symphony Solaris SDK

Platform Symphony  
Version 5.1  
April 2011



Copyright

© 1994-2011 Platform Computing Corporation

All rights reserved.

Although the information in this document has been carefully reviewed, Platform Computing Corporation ("Platform") does not warrant it to be free of errors or omissions. Platform reserves the right to make corrections, updates, revisions or changes to the information in this document.

UNLESS OTHERWISE EXPRESSLY STATED BY PLATFORM, THE PROGRAM DESCRIBED IN THIS DOCUMENT IS PROVIDED "AS IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL PLATFORM COMPUTING BE LIABLE TO ANYONE FOR SPECIAL, COLLATERAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING WITHOUT LIMITATION ANY LOST PROFITS, DATA, OR SAVINGS, ARISING OUT OF THE USE OF OR INABILITY TO USE THIS PROGRAM.

We'd like to hear from you

You can help us make this document better by telling us what you think of the content, organization, and usefulness of the information. If you find an error, or just want to make a suggestion for improving this document, please address your comments to [doc@platform.com](mailto:doc@platform.com).

Your comments should pertain only to Platform documentation. For product support, contact [support@platform.com](mailto:support@platform.com).

Document redistribution and translation

This document is protected by copyright and you may not redistribute or translate it into another language, in part or in whole.

Internal redistribution

You may only redistribute this document internally within your organization (for example, on an intranet) provided that you continue to check the Platform Web site for updates and update your version of the documentation. You may not make it available to your organization over the Internet.

Trademarks

®LSF is a registered trademark of Platform Computing Corporation in the United States and in other jurisdictions.

™ACCELERATING INTELLIGENCE, PLATFORM COMPUTING, PLATFORM SYMPHONY, PLATFORM JOB SCHEDULER, PLATFORM ISF, PLATFORM ENTERPRISE GRID ORCHESTRATOR, PLATFORM EGO, and the PLATFORM and PLATFORM LSF logos are trademarks of Platform Computing Corporation in the United States and in other jurisdictions.

®UNIX is a registered trademark of The Open Group in the United States and in other jurisdictions.

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

Microsoft is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries.

®Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

Intel®, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Other products or services mentioned in this document are identified by the trademarks or service marks of their respective owners.

Third-party license agreements

<http://www.platform.com/Company/third.part.license.htm>

Third-party copyright notices

<http://www.platform.com/Company/Third.Party.Copyright.htm>

---

# Contents

Installing and Using the Platform Symphony Solaris SDK .....	5
Overview .....	5
Prerequisites .....	5
Install the SDK .....	5
Configure SOAM_HOME .....	5
Build the C++ sample client .....	6
Build the Java samples .....	6
Package and deploy the sample service .....	7
Run the client to connect to Symphony 5.1 .....	8



# Installing and Using the Platform Symphony Solaris SDK

## Overview

The Platform Symphony Solaris SDK contains Symphony API libraries and code samples to enable you to develop and run Symphony clients and services on the Solaris platform.

You can then test your client and service on the Symphony grid.

## Prerequisites

### Operating System

Consult the list of supported platforms via the system requirements link on the Knowledge Center.

### Supported Compilers

C++:

- Sun WorkShop 6, update2, C++ 5.3, Patch 111685
- Sun C++ 5.7 2005/01/07

Java:

- Solaris SPARC Platform 32-bit Java 2 SDK, Java 1.4.2 or 1.5
- Ant version (recommended): 1.6.5.

## Install the SDK

1. Download the appropriate file for the Symphony Solaris SDK from the Platform FTP site. For example, the package for Solaris 8:

```
symphonySDK-sparc-sol8-32-5.1.0-buildnumber.tar.gz
```

Note that the gcc package only supports the client and cannot be used to deploy services.

2. Uncompress and untar the package on your Solaris development host. For example, the package for Solaris 8:

```
gunzip symphonySDK-sparc-sol8-32-5.1.0-buildnumber.tar.gz
tar xvf symphonySDK-sparc-sol8-32-5.1.0-buildnumber.tar
```

## Configure SOAM\_HOME

Configure SOAM\_HOME in cshrc, soam or profile, soam:

- For csh, edit cshrc, soam and change the following line to the directory in which you installed the Symphony SDK. For example, if you installed the SDK in the /opt directory:

```
setenv SOAM_HOME $SOAM_HOME
```

to:

```
setenv SOAM_HOME /opt/symphonySDK/SDK51
```

- For bash, edit `profile.soam` and change the following line to the directory in which you installed the Symphony SDK. For example, if you installed the SDK in the `/opt` directory:

```
SOAM_HOME=$SOAM_HOME
```

to:

```
SOAM_HOME=/opt/symphonySDK/SDK51
```

## Build the C++ sample client

1. Go to the `conf` directory under the directory in which you installed the Symphony SDK.  
For example, if you installed the SDK in `/opt/symphonySDK/SDK51`, go to `/opt/symphonySDK/SDK51/conf`.
2. Set the environment:
  - For `csh`, enter  

```
source cshrc.soam
```
  - For `bash`, enter  

```
. profile.soam
```
3. Compile using the Makefile located in `SOAM_HOME/5.1/samples/Cpp/SampleApp`:  

```
% make
```

## Build the Java samples

### Configure your environment for Java

1. Set your `JAVA_HOME` and `bin`.
  - a) Set your `JAVA_HOME` to point to the directory in which the JDK is located.  
For example, if your JDK is set to `/opt/java/j2sdk1.4.2`, set your `JAVA_HOME` to this path.
  - b) Ensure your Java `bin` directory is included in your `Path` environment variable.
2. If you are planning on using Ant to build the samples, set your `ANT_HOME` and `bin`.
  - a) Set your `ANT_HOME` to point to the directory in which you have installed Ant.
  - b) Ensure your Ant `bin` directory is included in your `Path` environment variable.

## Build the Java sample client and service

1. Change to the `conf` directory under the directory in which you installed the Symphony SDK.
2. Set the environment:
  - For `csh`, enter  

```
source $SOAM_HOME/conf/cshrc.soam
```

- For bash, enter  

```
. $SOAM_HOME/conf/profile.soam
```
3. Compile with the Makefile or with the Ant build file.
    - Compile with the Makefile:  
 Change to the \$SOAM\_HOME/5.1/samples/Java/SampleApp directory and run the command:  

```
make
```
    - Compile with the Ant build file:  
 Change to the \$SOAM\_HOME/5.1/samples/Java/SampleApp directory and run the command:  

```
ant
```

## Package and deploy the sample service

### Package and deploy the sample Java service

When you built the sample, the service package was automatically created for you. It is the .zip file located in \$SOAM\_HOME/5.1/samples/Java/SampleApp.

1. Copy the service package SampleServiceJavaPackage.zip and the sample application profile SampleAppJava.xml to any host in your cluster that has Symphony commands installed.
2. On the host to which you have copied the files, set the Symphony environment. For example, if your cluster is installed under /opt/ego directory:
  - For csh or tcsh, use cshrc.pl at form:  

```
source /opt/ego/cshrc.platform
```
  - For sh, ksh, or bash, use profile.pl at form:  

```
./opt/ego/profile.platform
```
3. Deploy the service package with the soamdeploy command.

#### Note:

If the SampleAppJava consumer does not exist in the cluster, you must create it using the PMC before issuing the soamdeploy command.

```
soamdeploy add SampleServiceJava -p SampleServiceJavaPackage.zip -c /SampleAppJava
```

The service package is deployed.

4. Check the list of deployed services with the soamdeploy view command:

```
soamdeploy view -c /SampleApplications/SampleAppJava
```

5. Register the application with the soamreg command:

```
soamreg SampleAppJava.xml
```

The application is registered and enabled.

6. Check the list of registered applications with the soamview app command:

```
soamview app
```

You should be able to see an enabled application with the name SampleAppJava on the list.

If the SampleAppJava application is not enabled, use the soamcontrol app enable command to enable the application.

**soamcontrol app enable SampleAppJava**

## Package and deploy the sample C++ service

You must package the files required by your service to create a service package.

For C++, you must create your own service package.

1. Change to the directory in which the compiled samples are located:

```
cd $SOAM_HOME/5.1/samples/CPP/Output/
```

2. Create the service package by compressing the service executable into a tar file:

```
tar -cvf SampleServiceCPP.tar SampleServiceCPP
```

```
gzip SampleServiceCPP.tar
```

You have now created your service package `SampleServiceCPP.tar.gz`.

3. Copy the service package `SampleServiceCPP.tar.gz` and the sample application profile `SampleApp.xml` to any host in your cluster that has Symphony commands installed.
4. On the host to which you have copied the files, set the Symphony environment. For example, if your cluster is installed under `/opt/ego` directory:
  - For `csch` or `tcsh`, use `cschrc.pl` at form:

```
source /opt/ego/cschrc.platform
```
  - For `sh`, `ksh`, or `bash`, use `profile.pl` at form:

```
./opt/ego/profile.platform
```
5. Deploy the service package with the `soamdeploy` command.

### Note:

If the `SampleAppCPP` consumer does not exist in the cluster, you must create it using the PMC before issuing the `soamdeploy` command.

```
soamdeploy add SampleService -p SampleServiceCPP.tar.gz -c /SampleAppCPP
```

The service package is deployed.

6. Check the list of deployed services with the `soamdeploy view` command:

```
soamdeploy view -c /SampleAppCPP
```

7. Register the application with the `soamreg` command:

```
soamreg SampleApp.xml
```

The application is registered and enabled.

8. Check the list of registered applications with the `soamview app` command:

```
soamview app
```

You should be able to see an enabled application with the name `SampleAppCPP` on the list.

If the `SampleAppCPP` application is not enabled, use the `soamcontrol app enable` command to enable the application.

```
soamcontrol app enable SampleAppCPP
```

## Run the client to connect to Symphony 5.1

- Your cluster must be a Symphony 5.1 grid.
- A service that works with your client must be deployed in the cluster and a corresponding application registered and enabled.
- Your client application must connect to the cluster with the same application name as in the registered application.

The following are instructions for running your client from your development machine. It is assumed the Symphony 5.1 SDK is installed in the `/opt` directory.

To run your client on a machine that does not have the Symphony 5.1 SDK installed, download and install the Symphony client package and follow the instructions in `install_client_unix_sym.pdf`.

1. Configure `SOAM_HOME` in `cschrc.symclient` or `profile.symclient` to the directory in which you have installed the Symphony SDK.
  - For `csch`, edit `cschrc.symclient` and change the following line to the directory in which you installed the Symphony SDK:
 

```
setenv SOAM_HOME $SOAM_HOME
```

 to:
 

```
setenv SOAM_HOME /opt/symphonySDK/SDK51
```
  - For `bash`, edit `profile.symclient` and change the following line to the directory in which you installed the Symphony SDK:
 

```
SOAM_HOME=$SOAM_HOME
```

 to:
 

```
SOAM_HOME=/opt/symphonySDK/SDK51
```
2. Configure the `EGO_MASTER_LIST`, and `EGO_KD_PORT` parameters in `$SOAM_HOME/conf/ego.conf` to connect to the remote Symphony cluster.
3. Set the environment to connect to the cluster.
  - For `csch`, enter
 

```
source cschrc.symclient
```
  - For `bash`, enter
 

```
. profile.symclient
```
4. Run the client application.