Platform Symphony™ - Calypso Integration User Guide

Symphony Version 3.x
May 2007

Comments to: doc@platform.com
Support: support@platform.com

Overview

This document provides instructions for installing and configuring the Symphony—Calypso integration package on the host on which Calypso is installed. Once integrated, Calypso can submit work to Symphony to run on the grid.

Contents

About the Symphony-Calypso integration

The goal of the Platform Symphony-Calypso integration is to increase Calypso performance and scalability by distributing Calypso work units to Symphony to run in parallel.

The integration layer will enable Calypso users to take advantage of Symphony features including resource allocation, workload and service scheduling, workload distribution, remote execution, and failover.

The Calypso integration layer consists of the following components:

- Symphony dispatcher for Calypso
- Symphony service
- logging facility (each component will generate its own set of logging messages in individual files.)
- troubleshooting utility

License agreement

Usage of this integration software is contingent upon acceptance of the terms and conditions of the Platform Computing Corporate Software License Agreement (the "Clickwrap Agreement") accompanying the Symphony software.

Prerequisites for integrating Calypso and Symphony

Install Calypso

Install Calypso on the client hosts and ensure it is working properly. The system environment variable *CALYPSO_HOME* should point to the Calypso installation directory. Refer to the Calypso documentation for more details.

Install JDK/JRE

Install JDK and JRE 1.4.2 and higher on the client and compute hosts. The system environment variable *JAVA_HOME* should point to the JDK/JRE installation directory.

Install Symphony

Install either Symphony or Symphony DE on the client host and Symphony on the compute and management hosts. The system environment variable *SOAM_HOME* should point to the Symphony installation directory.

Calypso version

Calypso version 8.3

Symphony version and platforms

Symphony Version 3.x on the following operating system platforms:

Operating System		Linux kernel, glibc
Windows Server 2003	Windows Server 2003 Standard Edition Windows Server 2003 Enterprise Edition Windows Server 2003 R2 Standard Edition Windows Server 2003 R2 Enterprise Edition	n/a
Windows XP	Windows XP Professional	n/a
Windows 2000	Windows 2000 Server Windows 2000 Professional	n/a
Linux 32-bit		
	Red Hat Enterprise Linux AS 4 SuSE Linux Enterprise Server 9	Kernel 2.6.x, glibc 2.3.x
	Red Hat Enterprise Linux AS 2.1 SuSE Linux Enterprise Server 8	Kernel 2.4.x, glibc 2.2.x
	Red Hat Enterprise Linux AS 3	Kernel 2.4.x, glibc 2.3.x
Solaris (client only)	Solaris 8	sparc-sol8-32

Basic concepts

Application

A service-oriented application is a type of application software, where the business logic is encapsulated in one or multiple software programs called services that are separated from its client logic.

Application profile

The application profile is an XML file that defines the properties of a Symphony application, including the name of the service that performs the calculation and the scheduling parameters to apply.

The application profile contains runtime parameters for workload, service, and the middleware that define how Symphony runs workload. An application profile provides flexibility to dynamically change application parameters without requiring you to change your application code and rebuild the application.

An application profile is associated with an application. An application is associated with one consumer. You must register the application profile of every application you want Symphony to manage.

Symphony client application

A program or executable that needs work done through a service. Requests are submitted via an API to the service.

Symphony service

A service is a self-contained business function that accepts one or more requests and returns one or more responses through a well-defined, standard interface.

The service performs work for a client program. It is a component capable of performing a task, and is identified by a name. Platform Symphony runs services on hosts in the cluster.

The Symphony service is the part of your application that does the actual calculation. The service encapsulates business logic.

Session

A group of tasks that share common characteristics, such as data.

Connection

The connection on which a session is created provides a conduit for the tasks.

Task

A task is the unit of work that runs on each individual host when Symphony workload is running. The task consists of a message request (input) and, when completed by a service, a response (output).

Consumer

A consumer is a generalized notion of something that uses resources.

Log files and levels

The integration software uses the log4j logging framework. Log classes can be found in the log4j properties files located in the `conf` directory. Here are the most commonly-used logging levels in the log4j framework:

- ALL has the lowest possible rank and is intended to turn on all logging.
- DEBUG level designates fine-grained informational events that are most useful for debugging an application.
- ERROR level designates error events that might still allow the application to continue running.
- FATAL level designates very severe error events that will presumably lead the application to abort.
- INFO level designates informational messages that highlight the progress of the application at coarse-grained level.
- TRACE level designates finer-grained informational events than the DEBUG level.
- WARN level designates potentially harmful situations.

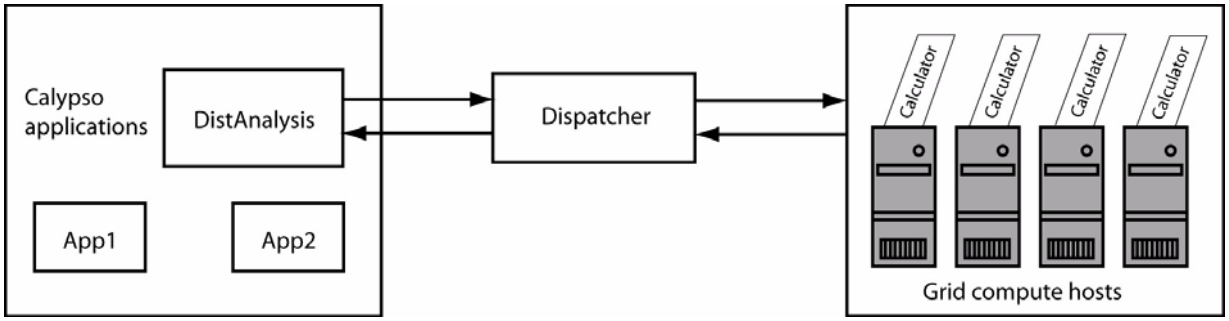
How Calypso works with Symphony

Before describing how Symphony fits in with Calypso, it is helpful to know how Calypso works on its own in a distributed environment.

Calypso model

The user works with the applications. When the user chooses to run the analysis in distributed mode, the application uses the DistAnalysis class. Here is the execution flow for task processing:

- The application splits the job into smaller units of work (tasks).
- The application requests a list of Calculators from the Dispatcher.
- The Dispatcher determines which Calculators to use for the application.
- The application sends tasks to the assigned Calculators.
- The Calculators perform the calculations and send the results back to the application.



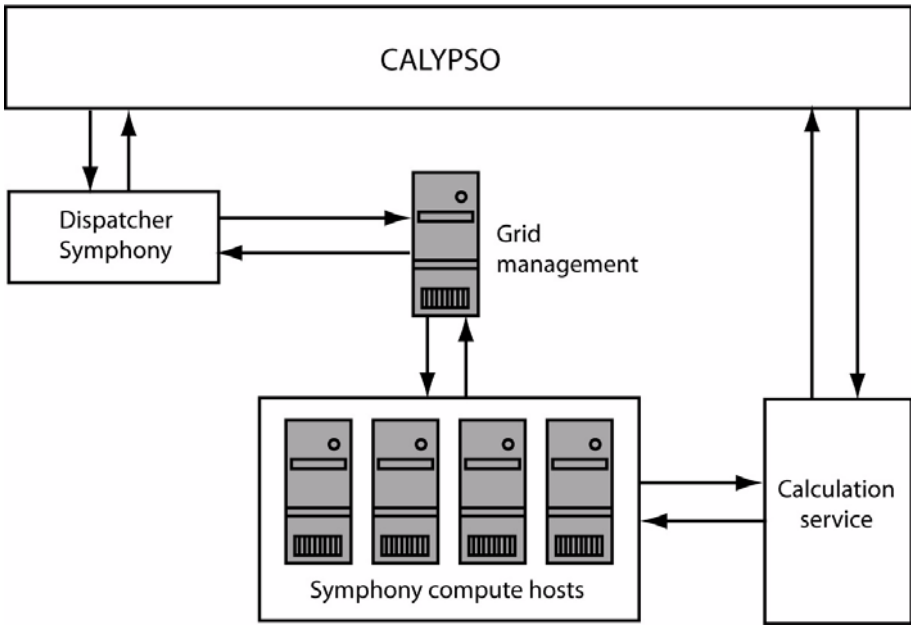
Symphony-Calypso integration model

When Calypso is integrated with Symphony, Calypso acts as the client application that submits work requests to the grid.

The Calypso integration enables Calypso to send tasks to Symphony, get the result back, and monitor the status of running tasks. Here is the execution flow for task processing:

- Calypso chooses pre-registered Symphony as the Dispatcher.
- Calypso splits an analysis request into smaller tasks.
- Calypso sends the task definition to Symphony Dispatcher through the Calypso Grid Interface, which invokes the Symphony API to contact the Symphony grid management host.
- Symphony grid management host dispatches tasks to compute hosts and runs the tasks in parallel.
- Symphony compute host invokes Calypso Calculator (wrapped in the calculation service) to process the tasks. The Calculator will contact Calypso Data Server directly to retrieve all the required information.

- each compute host sends output from the calculation service to the Management host, which, in turn, sends the result back to the Calypso Application.



DistAnalysis class

This class is the integration point. The integration overrides this class and replaces the part that sends jobs to the Dispatcher and receives results from the Calculators. Instead of sending to the Dispatcher, the tasks are sent to Symphony. and instead of getting results from the Calculator, we get results from Symphony.

Symphony Calculation Service

The Symphony Calculation Service runs on the compute hosts. It is implemented as a wrapper for the Calypso Calculator. The service gets tasks from Symphony and performs the calculation and sends the result back to the application through Symphony.

Installing the application (Windows)

Perform the following steps on the client host.

1. Create a destination directory for the Calypso integration package.
2. Set the *%SYM_CAL_INT_HOME%* environment variable to the destination directory.
3. Add the directory path `<SYM_CAL_INT_HOME>\tools` to the Path environment variable so that you can run the troubleshooting tools from any directory.
4. Add the following jar files to the classpath so that Calypso can invoke the Symphony dispatcher:
   - `%SYM_CAL_INT_HOME%\dispatcher\SymphonyDispatcher.jar`
   - `%SYM_CAL_INT_HOME%\dispatcher\log4j-1.2.14.jar`
   - `%SOAM_HOME%\<version>\<os name>\lib\JavaSoamApi.jar`
5. Copy `Cal803_Sym301.jar` to the *%SYM_CAL_INT_HOME%* directory. At the command prompt, run the following command to extract the integration package files:

   ```
   jar xvf Cal803_Sym301.jar
   ```

   The following table lists the files that are typically in the integration package. The actual file list may vary.

Files	Location	Description
SymphonyDispatcher.jar	%*SYM_CAL_INT_HOME*%\dispatcher	Symphony dispatcher
log4j-1.2.14.jar	%*SYM_CAL_INT_HOME*%\dispatcher	log4j library
isvClient.log4j.properties	%*SYM_CAL_INT_HOME*%\dispatcher	log4j property file
dispatcher.properties	%*SYM_CAL_INT_HOME*%\dispatcher	dispatcher property configuration file
CalypsoApp.xml	%*SYM_CAL_INT_HOME*%\service	Service application file
DistAnalysisSymphonyService.class	%*SYM_CAL_INT_HOME*%\service\package	Symphony Java service
isvService.log4j.properties	%*SYM_CAL_INT_HOME*%\service\package	log4j property file for service
log4j-1.2.14.jar	%*SYM_CAL_INT_HOME*%\service\package	log4j library
CalypsoPing.class	%*SYM_CAL_INT_HOME*%\tools	CalypsoPing tool
calypsoping.bat	%*SYM_CAL_INT_HOME*%\tools	CalypsoPing wrapper for Windows
testdsserver.bat	%*SYM_CAL_INT_HOME*%\tools	Test DS tool wrapper for Windows
TestDSServer.class	%*SYM_CAL_INT_HOME*%\tools	Test DS tool
DistAnalysisSymphonyService.jar	%*SYM_CAL_INT_HOME*%\tools	Calypsoping service
calypso.jar	%*SYM_CAL_INT_HOME*%\tools	Package for emulated Calypso environment for use with Calypsoping tool
calypsopingApp.xml	%*SYM_CAL_INT_HOME*%\tools	Calypsoping application profile

Files	Location	Description
isvClient.log4j.properties	%*SYM_CAL_INT_HOME*%\tools	log4j property file
SymphonyDispatcher.jar	%*SYM_CAL_INT_HOME*%\tools	Dispatcher for use with Calypsoping tool
dispatcher.properties	%*SYM_CAL_INT_HOME*%\tools	calypsoping dispatcher property configuration file

6 Configure the application; refer to Configuring the application (Windows).

Installing the application (Linux)

Perform the following steps on the client host.

1. Create a destination directory for the Calypso integration package.
2. Set the $*SYM_CAL_INT_HOME* environment variable to the destination directory.
3. Add the directory path `<SYM_CAL_INT_HOME>/tools` to the Path environment variable so that you can run the troubleshooting tools from any directory.
4. Add the following jar files to the classpath so that Calypso can invoke the Symphony dispatcher:
   - `$SYM_CAL_INT_HOME/dispatcher/SymphonyDispatcher.jar`
   - `$SYM_CAL_INT_HOME/dispatcher/log4j-1.2.14.jar`
   - `$SOAM_HOME/<version>/<os name>/lib/JavaSoamApi.jar`
5. Copy `Cal803_Sym301.jar` to the $*SYM_CAL_INT_HOME* directory. Run the following command to extract the integration package files:

   ```
   jar xvf Cal803_Sym301.jar
   ```

   The following table lists the files that are typically in the integration package. The actual file list may vary.

Files	Source Directory	Description
SymphonyDispatcher.jar	Cal803_Sym301/release/dispatcher	Symphony dispatcher
log4j-1.2.14.jar	Cal803_Sym301/release/dispatcher	log4j library
isvClient.log4j.properties	Cal803_Sym301/release/dispatcher	log4j property file
dispatcher.properties	Cal803_Sym301/release/dispatcher	dispatcher property configuration file
CalypsoApp.xml	Cal803_Sym301/release/service	Service application file
DistAnalysisSymphonyService.class	Cal803_Sym301/release/service/package	Symphony Java service
isvService.log4j.properties	Cal803_Sym301/release/service/package	log4j property file for service
log4j-1.2.14.jar	Cal803_Sym301/release/service/package	log4j library
CalypsoPing.class	Cal803_Sym301/release/tools	CalypsoPing tool
calypsoping.sh	Cal803_Sym301/release/tools	CalypsoPing wrapper for Windows
testdsserver.sh	Cal803_Sym301/release/tools	Test DS tool wrapper for Windows
TestDSServer.class	Cal803_Sym301/release/tools	Test DS tool
DistAnalysisSymphonyService.jar	Cal803_Sym301/release/tools	Calypsoping service
calypso.jar	Cal803_Sym301/release/tools	Package for emulated Calypso environment for use with Calypsoping tool
calypsopingApp.xml	Cal803_Sym301/release/tools	Calypsoping application profile

Files	Source Directory	Description
isvClient.log4j.properties	Cal803_Sym301/release/tools	log4j property file
SymphonyDispatcher.jar	Cal803_Sym301/release/tools	Dispatcher for use with Calypsoping tool
dispatcher.properties	Cal803_Sym301/release/tools	calypsoping dispatcher property configuration file

6 Configure the application; refer to Configuring the application (Linux).

Configuring the application (Windows)

1. The integration package includes templates for the CalypsoApp and Calypsoping application profiles. Modify these application profiles on the client host.
   a. In `CalypsoApp.xml` and `CalypsopingApp.xml`, replace the following placeholders with the appropriate directory paths or values:
      - Replace @SOAM_HOME@ with <*SOAM_HOME*>
      - Replace @MACHINE_TYPE@ with <*MACHINE_TYPE*>

        For example, on 32-bit Windows hosts, replace @MACHINE_TYPE@ with `win32-vc7`.
      - Replace @VERSION_NUM@ with <*VERSION_NUM*>

        For example, if Sym301 is installed, replace @VERSION_NUM@ with `3.0` or if Sym3.1 is installed, replace it with `3.1`.

   b. If Sym3.1 is installed, add "packageName ="DistAnalysisSymphonyService"" to the Service element in `CalypsoApp.xml` and `CalypsopingApp.xml`.

      Example of Service element in `CalypsoApp.xml` file:

```
<Service name="CalypsoService" description="Calypso Integration Service"
packageName="DistAnalysisSymphonyService">
         ...
```

   c. The application profiles have default settings for the service log level, log file name, and log file size. If necessary, modify the settings in the Service section of `CalypsoApp.xml` and `CalypsopingApp.xml`.

      Example of Service section in `CalypsoApp.xml` file:

```
<Service name="CalypsoService" description="Calypso Integration Service"
packageName="DistAnalysisSymphonyService">
         <osTypes>
            <osType name="NTX86"
                   startCmd="java -DSOAM_DEPLOY_DIR=${SOAM_DEPLOY_DIR}
                            -DLOG4J_LOG_LEVEL=DEBUG
                     -DLOG4J_APPENDER_FILE=@SOAM_HOME@/work/isv.${LOG4J_HOSTNAME}.log
                            -DLOG4J_MAX_FILE_SIZE=100000KB
                            -classpath @SOAM_HOME@; ...
         </osType>
```

2. If Sym3.1 is installed, open the `calypsoping.bat` file and set *VERSION_NUM* = 3.1 (default is 3.0).

   Example of `calypsoping.bat` file:

```
@echo off

set VERSION_NUM=3.1

...
```

3 Set the client log level, log file name, and log file size in `%SYM_CAL_INT_HOME%\dispatcher\isvClient.log4j.properties`.

4 Generate the service package; refer to Generating the service package (Windows).

Configuring the application (Linux)

1. The integration package includes templates for the CalypsoApp and Calypsoping application profiles. Modify these application profiles on the client host.
   a. In `CalypsoApp.xml` and `CalypsopingApp.xml`, replace the following placeholders with the appropriate directory paths or values :
      - Replace @SOAM_HOME@ with <*SOAM_HOME*>
      - Replace @MACHINE_TYPE@ with <*MACHINE_TYPE*>

        For example, on Linux hosts with Red Hat AS 3, replace @MACHINE_TYPE@ with `linux2.4-glibc2.3-x86`.
      - Replace @VERSION_NUM@ with <*VERSION_NUM*>

        For example, if Sym301 is installed, replace @VERSION_NUM@ with `3.0` or if Sym3.1 is installed, replace it with `3.1`.

   b. The application profiles have default settings for the service log level, log file name, and log file size. If necessary, modify the settings in the Service section of `CalypsoApp.xml` and `CalypsopingApp.xml`.

      Example of Service section in `CalypsoApp.xml` file:

```
            <osType name="LINUX86"
                   startCmd="java -DSOAM_DEPLOY_DIR=${SOAM_DEPLOY_DIR}
                             -DLOG4J_LOG_LEVEL=DEBUG
                      -DLOG4J_APPENDER_FILE=@SOAM_HOME@/work/isv.${LOG4J_HOSTNAME}.log
                             -DLOG4J_MAX_FILE_SIZE=100000KB
                             -classpath @SOAM_HOME@: ...
            </osType>
        </osTypes>
    </Service>
```

2. Set the client log level, log file name, and log file size in `$SYM_CAL_INT_HOME/dispatcher/isvClient.log4j.properties`.
3. Generate the service package; refer to Generating the service package (Linux).

Generating the service package (Windows)

1. Copy `calypso.jar` from `%CALYPSO_HOME%\jars` to `%SYM_CAL_INT_HOME%\service\package`.
2. Copy all Calypso software patches, if applicable, to `%SYM_CAL_INT_HOME%\service\package`.
3. Select the JDBC driver that is appropriate for your database. The following table lists the databases and their associated drivers. For example, if you are using Oracle 10, the JDBC driver should be `ojdbc14.jar`. Copy the driver from `%CALYPSO_HOME%\jars` to `%SYM_CAL_INT_HOME%\service\package`.

JDBC Driver	Database
jconn2.jar	Sybase
oracle92jdbc14.jar	Oracle 9
ojdbc14.jar	Oracle 10

4. Copy the Calypso system environment file `calypsosystem.properties.xxx` to `%SYM_CAL_INT_HOME%\service\package.`

NOTE: The system environment file is located in the Calypso environment. The format of the filename is calypsosystem.properties.*xxx* where *xxx* is user-defined.

5. Generate the Symphony service:
   a. Open a command prompt window.
   b. Change the current directory.

```
cd %SYM_CAL_INT_HOME%\service
```

   c. Create the service package in the `%SYM_CAL_INT_HOME%\service` directory.

```
jar cvf DistAnalysisSymphonyService.jar -C package.
```

6. Deploy the service package with the `soamdeploy` command.
   - Symphony 3.0

```
soamdeploy add -p DistAnalysisSymphonyService.jar -a
CalypsoApp.xml
```

   - Symphony 3.1

```
soamdeploy add DistAnalysisSymphonyService -p
DistAnalysisSymphonyService.jar -c
/SampleApplications/SOASamples
```

7. Deploy the calypsopingApp service package.
   a. Change the current directory.

```
cd %SYM_CAL_INT_HOME%\tools
```

b Deploy the calypsopingApp service package with the `soamdeploy` command.

- Symphony 3.0

```
soamdeploy add -p DistAnalysisSymphonyService.jar -a
CalypsopingApp.xml
```

- Symphony 3.1

```
soamdeploy add DistAnalysisSymphonyService -p
DistAnalysisSymphonyService.jar -c
/SampleApplications/SOATesting
```

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

- Symphony 3.0

```
soamdeploy view
```

- Symphony 3.1

```
soamdeploy view -c /SampleApplications/SOASamples
soamdeploy view -c /SampleApplications/SOATesting
```

9 Register the application with the `soamreg` command:

```
soamreg CalypsopingApp.xml
```

The application is registered and enabled.

10 Enable the `CalypsopingApp` service.

a `soamcontrol app disable symping`

b `soamcontrol app enable calypsopingApp`

11 Add Symphony dispatcher in Calypso. The dispatcher location points to `%SYM_CAL_INT_HOME%\dispatcher`

12 Configure Symphony dispatcher; refer to Configuring Symphony dispatcher.

Generating the service package (Linux)

1. Copy `calypso.jar` from `$CALYPSO_HOME/jars` to `$SYM_CAL_INT_HOME/service/package`.
2. Copy all Calypso software patches, if applicable, to `$SYM_CAL_INT_HOME/service/package`.
3. Select the JDBC driver that is appropriate for your database. The following table lists the databases and their associated drivers. For example, if you are using Oracle 10, the JDBC driver should be `ojdbc14.jar`. Copy the driver from `$CALYPSO_HOME/jars` to `$SYM_CAL_INT_HOME/service/package`.

JDBC Driver	Database
jconn2.jar	Sybase
oracle92jdbc14.jar	Oracle 9
ojdbc14.jar	Oracle 10

4. Copy the Calypso system environment file `calypsosystem.properties.xxx` to `$SYM_CAL_INT_HOME/service/package`.

NOTE: The system environment file is located in the Calypso environment. The format of the filename is calypsosystem.properties.*xxx* where *xxx* is user-defined.

5. Generate the Symphony service:
   a. Change the current directory.

      ```
      cd $SYM_CAL_INT_HOME/service
      ```

   b. Create the service package in the `$SYM_CAL_INT_HOME/service` directory.

      ```
      jar cvf DistAnalysisSymphonyService.jar -C package .
      ```

6. Deploy the service package with the `soamdeploy` command.
   - Symphony 3.0

     ```
     soamdeploy add -p DistAnalysisSymphonyService.jar -a
     CalypsoApp.xml
     ```

   - Symphony 3.1

     ```
     soamdeploy add DistAnalysisSymphonyService -p
     DistAnalysisSymphonyService.jar -c
     /SampleApplications/SOASamples
     ```

7. Deploy the calypsopingApp service package.
   a. Change the current directory.

      ```
      cd $SYM_CAL_INT_HOME/tools
      ```

   b. Deploy the calypsopingApp service package with the `soamdeploy` command.
   - Symphony 3.0

```
soamdeploy add -p DistAnalysisSymphonyService.jar -a
CalypsopingApp.xml
```

- Symphony 3.1

```
soamdeploy add DistAnalysisSymphonyService -p
DistAnalysisSymphonyService.jar -c
/SampleApplications/SOATesting
```

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

```
soamdeploy view -c /SampleApplications/SOASamples
```

```
soamdeploy view -c /SampleApplications/SOATesting
```

9 Register the application with the `soamreg` command:

```
soamreg CalypsopingApp.xml
```

The application is registered and enabled.

10 Enable the CalypsopingApp service.

a `soamcontrol app disable symping`

b `soamcontrol app enable calypsopingApp`

11 Configure Symphony dispatcher; refer to Configuring Symphony dispatcher.

Configuring Symphony dispatcher

The Symphony dispatcher is responsible for dispatching Calypso tasks into the Symphony environment and returning the results back to Calypso.

Before using the Symphony dispatcher, you must create the Symphony dispatcher type and make it available to Calypso. SymphonyDispatcher.sql contains the SQL statements that must be run on the Calypso database in order to create a dispatcher of type Symphony.

NOTE: Since Calypso menus are user-configurable, your set-up may vary from the menu sequence indicated in this procedure.

1 Create a new dispatcher type in Calypso.

   a Start the Calypso SQL Execute application.

   b Load the `SymphonyDispatcher.sql` file and click the **Execute** button.

2 Add Symphony dispatcher to Calypso.

NOTE: The Data Server and Event Server must be running before starting the Main Entry application.

   a Start the Calypso Main Entry application.

   b From the main menu, select **Configuration > System > Dispatcher Config**.

   c In the **Dispatcher Config** window, click the **New** button and provide a name for the new configuration, e.g., Symphony Config.

   d In the **Dispatcher Type** dropdown list, select **Symphony**.

   e Input the Symphony parameters.

   f Click the **Save** button to save the configuration. Click **Close**.

3 Test the application.

   ❖ Windows:

   Refer to Testing the application (Windows).

   ❖ Linux:

   Refer to Testing the application (Linux).

Testing the application (Windows)

1 Use calypsoping to test the installation and configuration of the application.

NOTE: The calypsoping tool must be run in the Calypso environment, i.e., on a host that has Calypso installed.

a At the command prompt, enter:

```
calypsoping
```

NOTE: If the command does not run, check that <*SYM_CAL_INT_HOME*>\tools was added to the Path environment variable.

Sample output:

```
Starting calypsoping!
SOAM_HOME=C:\SOAM\
CALYPSO_HOME=D:\Sym3Calypso
SYM_CAL_INT_HOME=C:\share
Preparing Calypso Jobs, Total Jobs: 10
Job Created, JobID: 1, Message: Calypso Job
Job Created, JobID: 2, Message: Calypso Job
Job Created, JobID: 3, Message: Calypso Job
Job Created, JobID: 4, Message: Calypso Job
Job Created, JobID: 5, Message: Calypso Job
Job Created, JobID: 6, Message: Calypso Job
Job Created, JobID: 7, Message: Calypso Job
.....
```

2 To test the data server connection, enter `testdsserver.bat` with associated input parameters at the command prompt.

NOTE: The testdsserver tool must be run in the Calypso environment, i.e., on a host that has Calypso installed.

Command syntax:

testdsserver.bat *username password appName envName envLocation*

Example:

```
testdsserver.bat calypso_user calypso Symphony systemTestENV d:\
Sym3Calypso
    Starting client ...
    Client Started on port 3864...
    [LOG|SYSTEM|2007 January 25, 14:14:07
    (921)|LOG|main|172.20.33.147 (user)]
    ## Env File Used (not correct with JavaWebStart):
    systemTestENV; UserHome: C:
```

```
\Documents and Settings\user
[END]
[LOG|SYSTEM|2007 January 25, 14:14:07
(968)|LOG|main|172.20.33.147 (user)]
Category Item(s) logged: None
[END]
[LOG|SYSTEM|2007 January 25, 14:14:07
(968)|LOG|main|172.20.33.147 (jcui)]
Level(s) logged: ALL
[END]
```

NOTE: If the data server is unavailable or cannot be reached, the tool will try to connect 10 times, then exit.

Testing the application (Linux)

1 Use calypsoping to test the installation and configuration of the application.

NOTE: The calypsoping tool must be run in the Calypso environment, i.e., on a host that has Calypso installed.

a Change the current directory.

```
cd $SYM_CAL_INT_HOME/tool
```

b Change the mode of calypsoping.sh.

```
chmod +x calypsoping.sh
```

c Run the command.

```
calypsoping.sh
```

NOTE: If the command does not run, check that <*SYM_CAL_INT_HOME*>/tools was added to the $PATH environment variable.

Sample output:

```
Starting calypsoping!
SOAM_HOME=/scratch/dev/user/Sym301DE
CALYPSO_HOME=/scratch/dev/user/Calypso
SYM_CAL_INT_HOME=/scratch/dev/user/Cal_Int
Prepareing Calypso Jobs, Total Jobs: 5
Job Created, JobID: 1, Message: Calypso Job
Job Created, JobID: 2, Message: Calypso Job
Job Created, JobID: 3, Message: Calypso Job
Job Created, JobID: 4, Message: Calypso Job
Job Created, JobID: 5, Message: Calypso Job
Creating Symphony Dispatcher...
Initializing Connection to Dispatcher...
Sending Jobs to Dispatcher...
Progress: Finish Sending Jobs to Symphony Grid, Total Jobs
 Sent: 5
Progress: Job Received from Grid, JobID: 1 (5 remaining)
Job Received, JobID:1, Message: Calypso Job Output
Progress: Job Received from Grid, JobID: 2 (4 remaining)
Job Received, JobID:2, Message: Calypso Job Output
Progress: Job Received from Grid, JobID: 3 (3 remaining)
Job Received, JobID:3, Message: Calypso Job Output
Progress: Job Received from Grid, JobID: 4 (2 remaining)
Job Received, JobID:4, Message: Calypso Job Output
Progress: Job Received from Grid, JobID: 5 (1 remaining)
```

```
Job Received, JobID:5, Message: Calypso Job Output
Ending calypsoping!
```

2 To test the data server connection, enter `testdsserver.sh` with associated input parameters at the command prompt.

NOTE: The testdsserver tool must be run in the Calypso environment, i.e., on a host that has Calypso installed.

Command syntax:

testdsserver.sh *username password appName envName envLocation*

Example:

```
sh ./testdsserver.sh calypso_user calypso Symphony systemTestENV
/scratch/dev/user/Calypso
    Starting client ...
    Client Started on port 51182...
    [LOG|SYSTEM|2007 February 08, 17:16:50
    (927)|LOG|main|172.20.1.107 (symlinux1)]
    ## Env File Used (not correct with
    JavaWebStart):systemTestENV; UserHome: /home/user
    [END]
    [LOG|SYSTEM|2007 February 08, 17:16:50
    (934)|LOG|main|172.20.1.107 (symlinux1)]
    Category Item(s) logged: None
    [END]
    [LOG|SYSTEM|2007 February 08, 17:16:50
    (935)|LOG|main|172.20.1.107 (symlinux1)]
    Level(s) logged: ALL
    [END]
```

NOTE: If the data server is unavailable or cannot be reached, the tool will try to connect 10 times, then exit.