Cockpit Phase Handlers 1.0 Component Specification

1. Design

The TopCoder Client Cockpit application defines a set of phase types. This component provides the plugins to the Phase Management component, whose logic is to check if these phases can be executed. Extra logic to execute the phases is also provided. The component also provides an EmailMessageGenerator interface and a default implementation that will be used to generate the email message to be sent.

1.1 Design Patterns

Strategy – Different EmailMessageGenerator implementations can be plugged into the handlers.

1.2 Industry Standards

JNDI

1.3 Required Algorithms

1.3.1 How to determine the end cases for the Schedule phase

-if `Contest.startDate` has been reached and `Contest.endDate` has not been reached the Schedule phase will end (use `System.currentTimeMillis()` to get current date) and the phase will be changed to Active

1.3.2 How to determine the end cases for the Active phase

-if `Contest.endDate` has been reached (use `System.currentTimeMillis()` to get current date) and the minimum number of submissions (retrieved from project.attributes using the "MinimumSubmissions" key) has been reached the Active phase will end and the phase will be changed to Action Required

-if `Contest.endDate` has been reached (use `System.currentTimeMillis()` to get current date) and the minimum number of submissions has not been reached the Active phase will end and the phase will be changed to Insufficient Submissions – ReRun Possible

-to determine if the minimum number of submissions has been reached get the minimum number of submissions from `project.attributes` using the "MinimumSubmissions" key and compare it with the number of submissions of the contest(`Contest.submissions`)

1.3.3 How to determine the end cases for the Action Required phase

-if the client has chosen a winner (`Contest.results` is not empty) the Action Required phase will end and the phase will be changed to Completed.
-if if there are less than 24 hours before `Contest.winnerAnnouncementDeadline` date the In Danger phase will be started; to check if there less than 24 hours before the deadline get the current date (`System.currentTimeMillis()`) and deduct this from Contest. winnerAnnouncementDeadline.getTime(). If the difference is less then 24*60*60*1000 and the difference is positive then there are less than 24 hours before the deadline

1.3.4 How to determine the end cases for the In Danger phase

-if the client has chosen a winner (`Contest.results` is not empty) the In Danger phase will end and the phase will be changed to Completed.
-if `Contest.winnerAnnouncementDeadline` has been reached the Abandoned phase will be started (see 1.3.3 algorithm, second part)

1.3.5 How to determine the end cases for the Extended phase

-if the `Contest.endDate` has been reached and the minimum number of submissions has been reached (see 1.3.2 for details) the Action Required phase is started
- if the `Contest.endDate` has been reached and the minimum number of submissions has not been reached (see 1.3.2 for details) the Insufficient Submissions phase is started

1.3.6 How to determine the end cases for the Repost phase

-if the `Contest.endDate` has been reached and the minimum number of submissions has been reached (see 1.3.2 for details) the Action Required phase is started

- if the `Contest.endDate` has been reached and the minimum number of submissions has not been reached (see 1.3.2 for details) the Insufficient Submissions – ReRun Possible phase is started

1.3.7 How to determine the end cases for the Insufficient Submissions phase

-if the `Contest.winnerAnnouncementDeadline` is reached the Abandoned phase is started (see 1.3.3 algorithm, second part)

1.3.8 How to set the data of the template

-get the template fields

```
TemplateFields root = template.getFields();
-get the nodes
Node[] nodes = root.getNodes();
-populate the nodes
for (int i = 0; i <  nodex.length; i++) {
    if (nodes[i] instanceof Field) {
        -get the value of the node from phase.attributes using
         nodes[i].name as key
        -if null was returned get the value of the node from
         phase.project.attributes using nodes[i].name as key
        -if the value is instance of Long,Date or String set the value to the
         node using the setter (set it to value.toString())
    }
}
-set the data to the template
String message = template.applyTemplate(root);
```

Note: the phase changing (specified in the above algorithms) is not performed by the handlers (it is specified to provide a better overall understanding); the phase starting and ending (opening and closing) will be performed by the PhaseManager implementations in start(phase,operator) and end(phase,operator) methods; the handlers perform additional logic like sending emails and changing the Contest status (or phase)

1.4 Component Class Overview

AbstractPhaseHandler

This class is an abstract handler that will be extended by all the other concrete handlers. It uses Configuration API component for configuration purposes. It has a method that will send an email to resources associated with timeline notification for the project, in case the phase ended or started. It has two other methods that will be used to send an email to the client in case there are less than eight (or one) hours before the deadline for the winner announcement. It uses Document Generator component to create the body of the message and Email Engine to send the email.
The fields are initialized in the constructor and never changed so there are no thread safety issues.

DraftPhaseHandler

This phase handler will be used by the PhaseManager implementations to:
-check if the Draft phase can be started or ended
-send an email to resources associated with timeline notification for the project if the phase has started or ended and change the status of the contest to "Draft" if the phase has started.
This class is not completely thread safe because, when the cache is empty, two calls of perform method, at the same time, may cause unexpected results. Also,

two calls of perform method, at the same time, for the same phase object may also cause unexpected results.

ScheduledPhaseHandler
This phase handler will be used by the PhaseManager implementations to:
-check if the "Scheduled" phase can be started or ended
-send an email to resources associated with timeline notification for the project if the phase has started or ended
This class is not completely thread safe because, when the cache is empty, two calls of perform method, at the same time, may cause unexpected results. Also, two calls of perform method, at the same time, for the same phase object may also cause unexpected results.

ActivePhaseHandler
This phase handler will be used by the PhaseManager implementations to:
-check if the Active phase can be started or ended
-send an email to resources associated with timeline notification for the project if the phase has started or ended and change the status of the contest to "Active" if the phase has started
This class is not completely thread safe because, when the cache is empty, two calls of perform method, at the same time, may cause unexpected results. Also, two calls of perform method, at the same time, for the same phase object may also cause unexpected results.

ActionRequiredPhaseHandler
This phase handler will be used by the PhaseManager implementations to:
-check if the "Action Required" phase can be started or ended
-send an email to resources associated with timeline notification for the project if the phase has started or ended and change the status of the contest to "ActionRequired" if the phase has started.
This class is not completely thread safe because, when the cache is empty, two calls of perform method, at the same time, may cause unexpected results. Also, two calls of perform method, at the same time, for the same phase object may also cause unexpected results.

InDangerPhaseHandler
This phase handler will be used by the PhaseManager implementations to:
-check if the In Danger phase can be started or ended
-send an email to resources associated with timeline notification for the project if the phase has started or ended and change the status of the contest to "InDanger" if the phase has started
-send an eight hour reminder email to the client if less than eight hours remain until the deadline for the winner announcement
-send an one hour reminder email to the client if less than one hour remain until the deadline for the winner announcement

This class is not completely thread safe because, when the cache is empty, two calls of perform method, at the same time, may cause unexpected results. Also, two calls of perform method, at the same time, for the same phase object may also cause unexpected results.

InsufficientSubmissionsRerunPossible
This phase handler will be used by the PhaseManager implementations to:
-check if the "Insufficient Submissions -ReRun Possible" phase can be started or ended
-send an email to resources associated with timeline notification for the project if the phase has started or ended and change the status of the contest to "InsufficientSubmissionsReRunPossible" if the phase has started.
This class is not completely thread safe because, when the cache is empty, two calls of perform method, at the same time, may cause unexpected results. Also, two calls of perform method, at the same time, for the same phase object may also cause unexpected results.

ExtendedPhaseHandler
This phase handler will be used by the PhaseManager implementations to:
-check if the "Extended" phase can be started or ended
-send an email to resources associated with timeline notification for the project if the phase has started or ended and change the status of the contest to "Extended" if the phase has started.
This class is not completely thread safe because, when the cache is empty, two calls of perform method, at the same time, may cause unexpected results. Also, two calls of perform method, at the same time, for the same phase object may also cause unexpected results.

RepostPhaseHandler
This phase handler will be used by the PhaseManager implementations to:
-check if the "Repost" phase can be started or ended
-send an email to resources associated with timeline notification for the project if the phase has started or ended and change the status of the contest to "Repost" if the phase has started.
This class is not completely thread safe because, when the cache is empty, two calls of perform method, at the same time, may cause unexpected results. Also, two calls of perform method, at the same time, for the same phase object may also cause unexpected results.

InsufficientSubmissionsPhaseHandler
This phase handler will be used by the PhaseManager implementations to:
-check if the "Insufficient Submissions" phase can be started or ended
-send an email to resources associated with timeline notification for the project if the phase has started or ended and change the status of the contest to "InsufficientSubmissions" if the phase has started.

This class is not completely thread safe because, when the cache is empty, two calls of perform method, at the same time, may cause unexpected results. Also, two calls of perform method, at the same time, for the same phase object may also cause unexpected results.

NoWinnerChosenPhaseHandler
This phase handler will be used by the PhaseManager implementations to:
-check if the "No Winner Chosen" phase can be started or ended
-send an email to resources associated with timeline notification for the project if the phase has started or ended and change the status of the contest to "No Winner Chosen" if the phase has started.
This class is not completely thread safe because, when the cache is empty, two calls of perform method, at the same time, may cause unexpected results. Also, two calls of perform method, at the same time, for the same phase object may also cause unexpected results.

CompletedPhaseHandler
This phase handler will be used by the PhaseManager implementations to:
-check if the "Completed" phase can be started or ended
-send an email to resources associated with timeline notification for the project if the phase has started or ended and change the status of the contest to "Completed" if the phase has started.
This class is not completely thread safe because, when the cache is empty, two calls of perform method, at the same time, may cause unexpected results. Also, two calls of perform method, at the same time, for the same phase object may also cause unexpected results.

CancelledPhaseHandler
This phase handler will be used by the PhaseManager implementations to:
-check if the "Cancelled" phase can be started or ended
-send an email to resources associated with timeline notification for the project if the phase has started or ended and change the status of the contest to "Cancelled" if the phase has started.
This class is not completely thread safe because, when the cache is empty, two calls of perform method, at the same time, may cause unexpected results. Also, two calls of perform method, at the same time, for the same phase object may also cause unexpected results.

TerminatedPhaseHandler
This phase handler will be used by the PhaseManager implementations to:
-check if the "Terminated" phase can be started or ended
-send an email to resources associated with timeline notification for the project if the phase has started or ended and change the status of the contest to "Terminated" if the phase has started.
This class is not completely thread safe because, when the cache is empty, two calls of perform method, at the same time, may cause unexpected results. Also,

two calls of perform method, at the same time, for the same phase object may also cause unexpected results.

AbandonedPhaseHandler
This phase handler will be used by the PhaseManager implementations to:
-check if the "Abandoned" phase can be started or ended
-send an email to resources associated with timeline notification for the project if the phase has started or ended and change the status of the contest to "Abandoned" if the phase has started.
This class is not completely thread safe because, when the cache is empty, two calls of perform method, at the same time, may cause unexpected results. Also, two calls of perform method, at the same time, for the same phase object may also cause unexpected results.

EmailMessageGenerator
This interface defines the contract for the implementing classes that will generate a message using a Template object and a Phase object.
Implementations should be thread safe.

DefaultEmailMessageGenerator
This class implements the EmailMessageGenerator interface and it is used to generate messages using a template and a phase. It can handle Long, Date and String objects as template data.
This class is thread safe because it has no state.

1.5 Component Exception Definitions

EmailSendingException
This exception will be thrown from the three protected methods that send emails if errors occur when preparing the message to be sent or when the message is sent.

PhaseHandlerConfigurationException
This exception will be thrown from the constructors of the handlers if while retrieving the configured properties or if a required property is missing.

EmailMessageGenerationException
This exception will be thrown by the EmailMessageGenerator implementations and the abstract handler if errors occur while generating the email message.

1.6 Thread Safety

The component is not completely thread safe. The handlers are not completely thread safe because, when the cache is empty, two calls of perform method, at the same time, may cause unexpected results. Also, two calls of perform method, at the same time, for the same phase object may also cause unexpected results. DefaultEmailMessageGenerator is thread safe because it has no state.

Note: To make the concrete handlers thread safe, its methods will need to be synchronized but considering the fact that thread safety is not a requirement and the fact that the handlers will most likely be used in a thread safe manner, the methods will not be synchronized.

2. Environment Requirements

2.1 Environment

- Development language: Java1.5
- Compile target: Java1.5, Java1.6

2.2 TopCoder Software Components

- **Studio Contest Manager 1.0** **-** ContestManager interface is used
- **Contest and Submission Entities 1.0 –** Contest and ContestStatus classes are used
- **Phase Management 1.0.4–** PhaseHandler interface is defined in this component.
- **Project Phases 2.0 –** Some classes from this component are used.
- **Email Engine 3.0 –** Used to send emails.
- **Document Generator 2.1 –** Used to generate the message.
- **Object Factory** 2.1 **–** Used to create an EmailMessageGenerator instance
- **Configuration Manager 2.1.5 –** used to read the configured properties
- **Base Exception 2.0** – used by PhaseHandlingException class.
- **Object Factory Config Manager Plugin 1.0 –** Used to create an EmailMessageGenerator instance

2.3 Third Party Components

JavaMail is used by Email Engine.

3. Installation and Configuration

3.1 Package Name

com.topcoder.clientcockpit.phases
com.topcoder.clientcockpit.phases.messagegenerators

3.2 Configuration Parameters

Parameter	**Description**	**Values**
bean_name	The name used with JNDI to retrieve the bean. It is **required** if the constructor without ContestManager parameter is used.	non empty string
use_cache	Indicates weather contest statuses will be cached. **Optional.** **If not present default to true.**	"true" or "false"(case insensitive).
send_start_phase_email	Indicates if an email should be sent when a phase starts. **Optional.** **If not present default to true.**	"true" or "false"(case insensitive).
start_email_template_source	Identifies the source from where the template will be retrieved. **Optional.**	non empty string
start_email_template_name	The name of the template. **Required If** send_start_phase_email **results true.**	non empty string
start_email_subject	The subject of the email. **Required If** send_start_phase_email **results true.**	non empty string
start_email_from_address	The address from which the email is sent. **Required If** send_start_phase_email **results true.**	non empty string
send_end_phase_email	Indicates if an email should be sent when a phase ends. **Optional.** **If not present default to true.**	"true" or "false"(case insensitive).
end_email_template_source	Identifies the source from where the template will be retrieved. **Optional.**	non empty string
end_email_template_name	The name of the template. **Required If** send_end_phase_email **results true.**	non empty string
end_email_subject	The subject of the email. **Required If** send_end_phase_email **results true.**	non empty string
end_email_from_address	The address from which the email is sent. **Required If** send_end_phase_email **results true.**	non empty string
send_eight_hours_phase_email	Indicates if an email should be sent when there less than eight hours before the winner announcement deadline. **Optional.** **If not present default to true.**	"true" or "false"(case insensitive).

eight_hours_email_template_source	Identifies the source from where the template will be retrieved. **Optional.**	non empty string
eight_hours_email_template_name	The name of the template. **Required If** send_eight_hours_phase_email **results true.**	non empty string
eight_hours_email_subject	The subject of the email. **Required If** send_eight_hours_phase_email **results true.**	non empty string
eight_hours_email_from_address	The address from which the email is sent. **Required If** send_eight_hours_phase_email **results true.**	non empty string
send_one_hour_phase_email	Indicates if an email should be sent when there less than one hour before the winner announcement deadline. **Optional. If not present default to true.**	"true" or "false"(case insensitive).
one_hour_email_template_source	Identifies the source from where the template will be retrieved. **Optional.**	non empty string
one_hour_email_template_name	The name of the template. **Required If** send_one_hour_phase_email **results true.**	non empty string
one_hour_email_subject	The subject of the email. **Required If** send_one_hour_phase_email **results true.**	non empty string
one_hour_email_from_address	The address from which the email is sent. **Required If** send_one_hour_phase_email **results true.**	non empty string
message_generator_key	The key used with Object Factory to create an EmailMessageGenerator object. **Optional.**	non empty string.
specification_factory_namespace	The namespace used to create a ConfigManagerSpecificationFactory instance. **Optional.**	non empty string

A sample is provided in config.xml in /docs directory.

3.3 Dependencies Configuration

None.

4. Usage Notes

4.1 Required steps to test the component

- Extract the component distribution.
- Follow Dependencies Configuration. Follow build script to ensure all dependency libraries are present in class path.
- Execute 'ant test' within the directory that the distribution was extracted to.

4.2 Required steps to use the component

See demo.

4.3 Demo

4.3.1 How the handlers defined in this component work with a PhaseManager implementation from Phase Management component

Not all the handlers defined in this component will be added to the PhaseManager. The scenario that will be presented is enough to understand how the handlers work with a PhaseManager implementation and how the contest status will change from “Draft” phase to “Completed” phase.

4.3.1.1 Create the handlers and add them to a PhaseManager implementation

//create a DefaultPhaseManager instance
```
PhaseManager manager = new DefaultPhaseManager(someNamespace);
```

//create and add handlers to the manager

//create a handler for the “Draft” phase
```
PhaseHandler draftHandler = new DraftPhaseHandler(namespace);
```
//add this handler to the manager and make this handler handle the starting and //ending of the “Draft” phase
```
PhaseType draftPhaseType = new PhaseType(id,”Draft”);
manager.registerHandler(draftHandler, draftPhaseType,
PhaseOperationEnum.START);
manager.registerHandler(draftHandler, draftPhaseType,
PhaseOperationEnum.END);
```

//create a handler for the “Scheduled” phase
```
PhaseHandler scheduledHandler = new ScheduledPhaseHandler(namespace);
```
//add this handler to the manager and make this handler handle the starting and //ending of the “Scheduled” phase
```
PhaseType scheduledPhaseType = new PhaseType(id,”Scheduled”);
manager.registerHandler(scheduledHandler, scheduledPhaseType,
PhaseOperationEnum.START);
manager.registerHandler(scheduledHandler, scheduledPhaseType,
PhaseOperationEnum.END);
```

//create a handler for the “Active” phase
```
PhaseHandler activeHandler = new ActivePhaseHandler(namespace);
```
//add this handler to the manager and make this handler handle the starting and //ending of the “Active” phase
```
PhaseType activePhaseType = new PhaseType(id,”Active”);
manager.registerHandler(activeHandler, activePhaseType,
PhaseOperationEnum.START);
manager.registerHandler(activeHandler, activePhaseType,
PhaseOperationEnum.END);
```

//create a handler for the “Action Required” phase
```
PhaseHandler actionRequiredHandler = new
     ActionRequiredPhaseHandler(namespace);
```

//add this handler to the manager and make this handler handle the starting and
//ending of the “Action Required” phase

```
PhaseType actionRequiredPhaseType = new PhaseType(id,”Action Required”);
manager.registerHandler(actionRequiredHandler, actionRequiredPhaseType,
PhaseOperationEnum.START);
manager.registerHandler(actionRequiredHandler, actionRequiredPhaseType,
PhaseOperationEnum.END);
```

//create a handler for the “Completed” phase

```
PhaseHandler completedHandler = new CompletedPhaseHandler(namespace);
```

//add this handler to the manager and make this handler handle the starting of the
//“Completed” phase (this phase is an end phase)

```
PhaseType completedPhaseType = new PhaseType(id,”Completed”);
manager.registerHandler(completedHandler, completedPhaseType,
PhaseOperationEnum.START);
```

4.3.1.2 Create a Project instance and add a Contest instance and phases to the project

```
All the phase statuses are initial to be PhaseStatus.SCHEDULED
```

//create a Contest instance and set its fields…

```
Contest contest = new Contest();
```

//create a Project instance

```
Project project = new Project (startDate, workdays);
```

//add the contest instance to the project

```
project.setAttribute(“contest”,contest);
project.setAttribute("MinimumSubmissions", 1);
project.setAttribute("ResourceEmails", (ArrayList) toAddresses);
```

//create a “Draft” phase

```
Phase draftPhase = new Phase(project,1);
draftPhase.setPhaseType(draftPhaseType);
draftPhase.setPhaseStatus(PhaseStatus.SCHEDULED);
```

//set other fields of the phase…
//add the phase to the project

```
project.addPhase(draftPhase);
```

//create a “Scheduled” phase

```
Phase scheduledPhase = new Phase(project,1);
scheduledPhase.setPhaseType(scheduledPhaseType);
scheduledPhase.setPhaseType(CockpitPhaseType.SCHEDULED);
```

//set other fields of the phase…
//add the phase to the project

```
project.addPhase(scheduledPhase);
```

//create a “Active” phase

```
Phase activePhase = new Phase(project,1);
activePhase.setPhaseType(activePhaseType);
activePhase.setPhaseStatus(PhaseStatus.SCHEDULED);
```

//set other fields of the phase…
//add the phase to the project

```
project.addPhase(activePhase);
```

//create a “Action Required” phase

```
Phase actionRequiredPhase = new Phase(project,1);
actionRequiredPhase.setPhaseType(actionRequiredPhaseType);
actionRequiredPhase.setPhaseStatus(PhaseStatus.SCHEDULED);
//set other fields of the phase…
//add the phase to the project
project.addPhase(actionRequiredPhase);

//create a "Completed" phase
Phase completedPhase = new Phase(project,1);
completedPhase.setPhaseType(completedPhaseType);
completedPhase.setPhaseStatus(PhaseStatus.SCHEDULED);
//set other fields of the phase…
//add the phase to the project
project.addPhase(completedPhase);
```

4.3.1.3 Use PhaseManager to start project phases and end project phases

Observation:

-`manager.canStart(phase)` call will call the corresponding `handler.canPerform(phase)` method (if phase is "Draft" then `draftHandler.canPerform(phase)` method will be called)

-`manager.canEnd(phase)` call will call the corresponding `handler.canPerform` method

-`manager.start(phase,operator)` will call the corresponding `handler.perform(phase,operator)`
After phase started, its phaseStatus is changed to be PhaseStatus.OPEN
-`manager.end(phase,operator)` will call the corresponding `handler.perform(phase,operator)`
After phase ended, its phaseStatus is changed to be PhaseStatus.CLOSED

```
//start "Draft" phase; when the manager.start call is made the status of the contest
//will be moved to "Draft" and a notification email will be sent
if (manager.canStart(draftPhase))
    manager.start(draftPhase,"");

//end "Draft" phase; when the manager.end call is made a notification email will
//be sent
if (manager.canEnd(draftPhase))
    manager.end(draftPhase,"");

//start "Scheduled" phase; when the manager.start call is made the status of the
//contest will be moved to "Scheduled" and a notification email will be sent
if (manager.canStart(scheduledPhase))
    manager.start(scheduledPhase,"");

//The ScheduledPhaseHandler will not update contest status, it's updated
//externally
contest.setStatus(CockpitContestStatus.SCHEDULED);
```

//end "Scheduled" phase; when the `manager.end` call is made a notification email
//will be sent

```
if (manager.canEnd(scheduledPhase))
    manager.end(scheduledPhase,"");
```

//start "Active" phase; when the `manager.start` call is made the status of the
//contest will be moved to "Active" and a notification email will be sent

```
if (manager.canStart(activePhase))
    manager.start(activePhase,"");
```

//Within "Active" phase, the developer submits to contest

//end "Active" phase; when the `manager.end` call is made a notification email will
//be sent

```
if (manager.canEnd(activePhase))
    manager.end(activePhase,"");
```

//start "Action Required" phase; when the `manager.start` call is made the status of
//the contest will be moved to "ActionRequired" and a notification email will be
//sent

```
if (manager.canStart(actionRequiredPhase))
    manager.start(actionRequiredPhase,"");
```

//Within "Action Required" phase, the client takes action to choose a winner

//end "Action Required" phase; when the `manager.end` call is made a notification
//email will be sent

```
if (manager.canEnd(actionRequiredPhase))
    manager.end(actionRequiredPhase,"");
```

//start "Completed" phase; when the `manager.start` call is made the status of the
//contest will be moved to "Completed" and a notification email will be sent

```
if (manager.canStart(completedPhase))
    manager.start(completedPhase,"");
```

5. Future Enhancements

New implementations of EmailMessageGenerator could be provided.