Online Review Phases 1.6.1 Component Specification

All changes performed when synchronizing documentation with the version 1.6 of the source code of this component and fixed errors in the CS are marked with **purple**.

All changes made in the version 1.6.1 are marked with **blue**.

All new items in the version 1.6.1 are marked with **red**.

The Online Review application defines a set of phase types. This component provides the plug-ins 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 provides a set of phase handler classes. They implement PhaseHandler interface to provide plug-in handlers for Phase Management component. Phase Management can load these handler classes from a configuration files. All handlers has constructor that can load settings from a configuration namespace.

Each handler provides two methods, that are canPerform () and perform(). canPerform() method determines if a phase can start or stop (and if cannot, provides an explanation message, why). While perform() executes some additional logic to start or stop a phase. These methods examine the input phase status to choose the action. If phase status is “Scheduled”, they will check for “can start” or execute additional start logic. If phase status is “Open”, they will check for “can stop” or execute additional stop logic.

The phase handlers in this component provide the logic to handle the following phases:

- Checkpoint Submission
- Checkpoint Screening
- Checkpoint Review
- Specification Submission
- Specification Review
- Registration
- Submission
- Screening
- Review
- Appeals
- Appeals Response
- Aggregation
- Aggregation Review
- Final Fix
- Final Review
- Approval
- Post-Mortem

The phase handlers can send email to a group of users associated to the timeline notification of the current project. User can configure to send email when starting the phase or when ending the phase, in both cases or not to send email at all.

This component provides some extra useful features such as:

- Adding some utility classes for getting value from lookup tables. They are phase status, phase type, resource role, submission status, submission type, project info type, upload status, upload type and notification type. These lookup tables are frequently used in phase related operations.
- Provide caching mechanism for lookup id. Because lookup id and lookup value pairs are not changed per database installation, caching them will minimize the database queries.

Version 1.1 changes:

- Ported component from ZUML to TCUML
- Updated all functionality added between versions 1.0 and 1.1 via bug fixes.
- Added a new phase “Post-Mortem” that is inserted when a registration, submission, screening, or appeals response phases result in zero applicable participants.
- Added routine in final review to insert an approval phase
- Added routine in approval to be able to launch another final-fix-final-review cycle.

Version 1.2 changes:

- Added capability to support different email template for different role (e.g. Submitter, Reviewer, Manager, etc). This includes default email templates for all roles.
- Support for more information in the email generated. The information could be specific to the phase. For instance, in appeal phase, the information about the submitters’ scores could be given.

Version 1.3 changes:

- Changes from the Contest Dependency Automation assembly and End Of Project Analysis release assembly are integrated.
- The configuration for all phase handlers is revised to be more simple and powerful
- ManagerHelper class is updated by adding a new getter method for retrieving project detail url from configuration.

Version 1.4 changes:

- Added handlers for specification submission and specification review phases: SpecificationSubmissionPhaseHandler and SpecificationReviewPhaseHandler.
- Updated RegistrationPhaseHandler to make it check project dependencies only if Registration is the first phase of the project.
- Handlers are updated not to use ContestDependencyAutomation when inserting new phases to the project.
- ApprovalPhaseHandler was updated to end Approval phase, even when there are no submitted approval scorecards.

Version 1.6 changes:

- Added handlers for checkpoint submission, checkpoint screening and checkpoint review phases: CheckpointSubmissionPhaseHandler, CheckpointScreeningPhaseHandler and CheckpointReviewPhaseHandler.
- Added link to Studio contest in mail notifications sent for Studio competitions.
- Removed EJB calls in PostMortemPhaseHandler and replaced them with library calls
- Updated ReviewPhaseHandler to populate winners for Studio competitions

Version 1.6.1 changes:

- canPerform() methods of all handlers were updated to return not only true/false value, but additionally an explanation message in case if operation cannot be performed.

1.1 Design Patterns

Strategy pattern is used in phase handler classes. They implement the PhaseHandler interface to provide plug-in handlers for the Phase Management component.

1.2 Industry Standards

JDBC, XML, SQL

1.3 Required Algorithms

1.3.1 Loading configuration property [Common]

```
This is the logic of AbstractPhaseHandler constructor
Get the value of the 'ConnectionFactoryNS' property in the given namespace –
required
Create a new instance of DBConnectionFactoryImpl using the above value and assign
to the 'factory' field.
Get the value of the 'ManagerHelperNamespace' property in the given namespace -
optional
      If the property does not exist, create the ManagerHelper field using
default constructor
      Else, create the ManagerHelper field using the configured namespace.
Get the value of the 'ConnectionName' property - optional. Assign its value to
'connectionName' field if it exists.

Load the schemes defined as child property of 'Schemes', each schemes child
property contains a list of roleName to be mapped to this schemes, regex '*' can
be used to match all the resource roles in database.
Retrieve the full resource roles from database by calling
managerHelper.getResourceManager().getAllResourceRoles().
For each scheme in schemes list:
      For xx in {start, end}.
      Create new EmailOptions object
      Read the configuration values [$scheme]/xxPhaseEmail/ into the EmailOptions object
      Add it to xxPhaseEmailOptions map for each role names related to the scheme. If the
      role name contains '*', map the created email options to every role name.

Get the value of the 'ProjectDetailsURL' property by calling
ManagerHelper#getProjectDetailsBaseUrl() method.

Throw ConfigurationException if any required field is missing or empty.
```

1.3.2 Common logic for accessing database methods [Common]

```
To satisfy the requirement of connection cannot be cached. Methods that access
database must create connection at the beginning and close connection when
finished.

With methods that update database, transaction logic should be provided as below
to ensure data consistence.
Create connection using the 'factory' instance and the configured DB connection
name (connectionName).
If connectionName is null, create the default connection.

Set the connection's auto commit to false.
Do create/update.
If error occurred, call rollback, and then throw PhaseHandlingException that wrap
the error.
Call commit if everything is fine.
Close the connection in finally block.
```

1.3.3 Email sending when changing phase [Common]

This method is used in every phase handler class, perform() method to send email to the related users.

```
Do not send any e-mails for phases whose duration is zero or its dates have not
yet been set.
```

```
First check value of 'sendStartPhaseEmail' and 'sendEndPhaseEmail'.
    If phase status is 'Scheduled' and sendStartPhaseEmail=true --> Initialize and
send start email
    If phase status is 'Open' and sendEndPhaseEmail=true --> Initialize and send
end email
    Otherwise, do nothing
```

Use DocumentGenerator

```
// instantiate the Document Generator
DocumentGenerator instance = new DocumentGenerator();
instance.setDefaultTemplateSource(new FileTemplateSource());

Project project =
managerHelper.getProjectManager().getProject(phase.getProject().getId());
```

Get email addresses to send:

```
    Use Resource Management to get all notification ids for a project. The
notification ids here are
    the external id of users that need to be notified when a phase change.
        Lookup project info type id for "Timeline Notification" use
NotificationTypeLookupUtility class. (typeId)
        long[] externalIds =
managerHelper.getResourceManager().getNotifications(phase.getProject().getId(),
typeId);

// get projectId
long projectId = phase.getProject().getId();

// the list of resource where the email is to be sent
List resources = new ArrayList();

// populate the list
ResourceRole[] roles = resourceManager.getAllResourceRoles()
For role in roles
       // find set of resources with this role
       Filter resourceRoleFilter =
             ResourceRoleFilterBuilder.createResourceRoleIdFilter(role.getId());
       Filter projectIdFilter =
             ResourceFilterBuilder.createProjectIdFilter(projectId);
        Filter fullFilter = SearchBundle.buildAndFilter(resourceRoleFilter,
                             projectIdFilter);
       Resource[] cur=getResourceManager().searchResources(fullFilter);
       For each resource in cur
             // is this resource needs notification?
             long externalId = Long.parseLong(resource.getProperty("External
                Reference ID"));
             if (externalId is included in externalIds)
                   also, compare if the resource in this role has higher priority
             than other roles
                   resources.add(resource);

// process each resource
For each res in resources
       // Use Resource Manager to get the role of the user
       Resource res = managerHelper.getResourceManager().getResource(id);
       ResourceRole role = res.getResourceRole();
       String roleName = role.getName();

       // (xx can be start or end)
       EmailOptions options = xxPhaseEmailOptions.get("default");

       EmailOptions roleOptions = xxPhaseEmailOptions.get(roleName);
```

```
If (roleOptions is not null)
       For each not-null property of roleOptions
              set the property of options object to that of roleOptions
if (roleOptions.isSend() is false or options.isSend is false)
       continue;

//The idea is copy the setting from default-options and if any fields in
//the role-options is not null, using the role-options's setting, also, if
//the role-options is null, using the default one

//to send email now
String name = options.getTemplateName();
Template template = instance.getTemplate(name);

// for each external user, set field values
TemplateFields root = instance.getFields(template);
Node[] nodes = root.getNodes();

for (int i = 0; i <  nodes.length; i++) {
       if (nodes[i] instanceof Field) {
              Field field = nodes[i];
           // Set field value using field.setValue() method,
           // base on field.getName()
       }
       Else if (nodes[i] instanceof Loop) {
         List loopItems = (List) values.get(loop.getLoopElement());
         if (loopItems == null) {
            throw new PhaseHandlingException("For loop :" +
         loop.getLoopElement() + ", the value in look up maps should not be
         null.");
         }
         for (int t = 0 ; t < loopItems.size(); t++) {
             NodeList item = loop.insertLoopItem(t);
             setNodes(item.getNodes(), user, project, phase, (Map<String,
         Object>) loopItems.get(t), bStart);
         }
       }
       Else if (nodes[i] instanceof Condition) {
              Condition condition = ((Condition) nodes[i]);
          if (values.containsKey(condition.getName())) {
condition.setValue(values.get(condition.getName()).toString());
        }
         setNodes(condition.getSubNodes().getNodes(), user, project, phase,
values, bStart);
       }

String emailContent = instance.applyTemplate(root);
```

Send email

```
    TCSEmailMessage message = new TCSEmailMessage();
    message.setSubject(options.getSubject() + ": " + projectName);
    message.setBody(emailContent);
    message.setFromAddress(options.getFrom());
    message.setToAddress(user.getEmail());
    EmailEngine.send(message);
```

The map between field name and value to set:

```
    PROJECT_NAME -> project's name
    PROJECT_VERSION --> project's version
    PROJECT_CATEGORY --> project's category
    PHASE_TYPE --> Phase.getPhaseType().getName()
    PHASE_OPERATION --> "start" or "end" depends on the input phase status.
```

```
        If status is "Scheduled" -> "start"
        If status is "Open" -> "end"
    PHASE_TIMESTAMP --> The time of performing the phase
    USER_FIRST_NAME --> ExternalUser.getFirstName()
    USER_LAST_NAME --> ExternalUser.getLastName()
    USER_HANDLE --> ExternalUser. getHandler()
    OR_LINK --> "<![CDATA[" + projectDetailsBaseURL + project.getId() + "]]>"
    STUDIO_LINK --> <![CDATA[" + studioProjectDetailsBaseURL +
project.getId()+"]]>"
    For each other value, look up values Map argument with key = field name.
```

Those field names are constants. They are and used in a template of DocumentGenerator to generate documents.

1.3.4 *Using various related manager components [Common]*

This component uses these components for searching/updating database:
Deliverable Management: UploadManager interface. Default implementation is PersistenceUploadManager class and default persistence class is SqlUploadPersistence.
Resource Management: ResourceManager interface. Default implementation is PersistenceResourceManager class and default persistence class is SqlResourcePersistence.
Review Management: ReviewManager interface. Default implementation is DefaultReviewManager class.
Scorecard Management: ScorecardManager interface. Default implementation is ScorecardManagerImpl class.
Project Management: ProjectManager and ProjectLinkManager interfaces. Default implementations are ProjectManagerImpl and ProjectLinkManagerImpl classes.

Phase Management : PhaseManager interface. Default implementation is DefaultPhaseManager class.

User Project Data Store : The UserRetrieval interface are used from this component. Default implementations used are DBUserRetrieval.

Review Score Aggregator : The ReviewScoreAggregator class is used.

These default implementations of these interfaces require initialization of DBConnectionFactory, SearchBundle and IDGenerator components. These components should be properly configured. See their component specifications for configuration information.

All of those interface provide search() method for searching.
search() method receives a parameter of type Filter, a class of Search Builder component.
Filter can be created by some utility methods provided in those management components. Filter can be created for various search conditions and can be combined using AND/OR/NOT.

Some phase handler's perform method need to update database.
In the update methods, operator parameter is required. Simply pass the "operator" parameter to it.

1.3.5 *Loading manager instance use reflection [Common]*

ManagerHelper class loads the manager from the settings in a configuration namespace. The properties are detailed in section “3.2 Configuration Parameters”. Their values are used to create manager instances use reflection. The phase handler classes keep this class as a field to use when needed.
Following is details:

- For ScorecardManager, ReviewManager, ProjectManager, ProjectLinkManager, ReviewScoreAggregator, PhaseManager, UserRetrieval:
  - If "Namespace" property does not present, default constructor will be used.
  - Else, use the namespace to pass to the constructor.
- For ResourceManager and UploadManager, the constructor parameters are SearchBundle and IDGenerator instances.
  - Create SearchBundleManager instance.
  - Use SearchBundleManager to create SearchBundle instances from the configured names.
  - Use IDGeneratorFactory to create IDGenerator instances from the configured names.
- For UserTermsOfuse and ProjectRoleTermsOfUse
  - Default constructor for passed namespace will be used to create object.

1.3.6 *Lookup values [Common]*

```
Logic for lookUpId() methods in the utility classes:
- Look at 'cachedPairs' map to see if parameter 'value' exists as a key
    If yes
        Return the cached id
    Else:
        Query database for lookup id base the given value
        Save the pair to 'cachedPairs' map. Key is "lookup value", value is
"lookup id"
        Return the id
    End If

Select commands (? will be the 'value' parameter)
    For SubmissionStatusLookupUtility
        SELECT submission_status_id FROM submission_status_lu WHERE name = ?
    For SubmissionTypeLookupUtility
        SELECT submission_type_id FROM submission_type_lu WHERE name = ?
    For ResourceRoleLookupUtility
        SELECT resource_role_id FROM resource_role_lu WHERE name = ?
    For PhaseTypeLookupUtility
        SELECT phase_type_id FROM phase_type_lu WHERE name = ?
    For PhaseStatusLookupUtility
        SELECT phase_status_id FROM phase_status_lu WHERE name = ?
    For ProjectInfoTypeLookupUtility
        SELECT project_info_type_id FROM project_info_type_lu WHERE name = ?
    For UploadStatusLookupUtility
        SELECT upload_status_id FROM upload_status_lu WHERE name = ?
    For UploadTypeLookupUtility
        SELECT upload_type_id FROM upload_type_lu WHERE name = ?
```

1.3.7 *Check if a phase start time is reached [Routine]*

Corresponds to PhasesHelper#reachedPhaseStartTime() method.

```
Call phase.calcStartDate() method to get the date time when the given phase can
start.
If phase start time is null, return true.
If current date time is later than or equal to phase start time, return true.
```

1.3.8 *Check if a phase end time is reached [Routine]*

Corresponds to PhasesHelper#reachedPhaseEndTime() method.

```
Call phase.calcEndDate() method to get the date time when the given phase can end.
If current date time is later than or equal to phase end time, return true.
```

1.3.9 *Check if all dependencies of a phase have correct status [Routine]:*

Corresponds to PhasesHelper#arePhaseDependenciesMet() method. Starting from the version 1.6.1 this routine not only returns true/false, but additionally provides an explanation message in case if at least one of dependencies has invalid status.

```
Get an array of phase's dependencies
    dependencies[]=phase.getAllDependencies()

Return OperationCheckResult.SUCCESS if dependencies.length = 0

For each dependency
    Get its phase using getDependency() method (subPhase)
    Get phase status name using subPhase.getPhaseStatus().getName()
    If phase has to be started
       If dependency.isDependencyStart() and dependency.isDependentStart()
          If the subPhase status name is not "Open",
              return new OperationCheckResult("Dependency " +
subPhase.getPhaseType().getName() + " phase is not yet started")
       If not dependency.isDependencyStart() and dependency.isDependentStart()
          If the subPhase status name is not "Closed",
              return new OperationCheckResult("Dependency " +
subPhase.getPhaseType().getName() + " phase is not yet ended")
    Else
       If dependency.isDependencyStart() and not dependency.isDependentStart()
          If the subPhase status name is not "Open",
              return new OperationCheckResult("Dependency " +
subPhase.getPhaseType().getName() + " phase is not yet started")
       If not dependency.isDependencyStart() and not dependency.isDependentStart()
          If the subPhase status name is not "Closed",
              return new OperationCheckResult("Dependency " +
subPhase.getPhaseType().getName() + " phase is not yet ended")
End For

Return OperationCheckResult.SUCCESS
```

1.3.10 *Locating phases [Routine]*

Corresponds to PhasesHelper#locatePhase() method.

```
In some methods, from the current phase we need to find a backward phase or a
forward phase. For example, from Aggregation phase, we may need to go back and
find the nearest Review phase. Or from Submission, we may need to go forward and
find the nearest Screening phase.

Get the current project
    currentPhase.getProject() (project)

Get all phases belong to the project. Note that the phases are sorted
    Using project.getAllPhases() (phases[])

Find the index of the currentPhase in phases[] (index)
    Using currentPhase.getId() and compare with instances in phases[]

To find a nearest backward phase of a type, start from index-1 and decreasing
To find a nearest forward phase of a type, start from index+1 and increasing

If cannot find the nearest backward/forward case of the given type, throw
PhaseHandlingException
```

1.3.11 *Search resource based on resource role names and phase id [Routine]*

Corresponds to PhasesHelper#searchResourcesForRoleNames() method.

```
Input:
    - resourceRoleNames(String[]): An array of resource role  names to search
    Resource is people who assigned for a phase. Resource role can be "Submitter",
"Screener", "Reviewer", "Aggregator", etc.
    - phaseId: The phase id to search for resource
Output:
    An array of Resource instance (resources[])

Lookup resource ids for the resource role names use "ResourceRoleLookupUtility"
class.

Search using Resource Management
    Create filter with resource_role_id IN [roleIds] AND phase id = phase.getId()
    Resource[] resources = ResourceManager.search(filter)
```

1.3.12 *Search all reviews for a phase based on resource roles [Routine]*

Corresponds to PhasesHelper#searchReviewsForResourceRoles() method.

```
Input:
    - phaseId(long): Id of the phase to search for reviews
    - resourceRoleNames(String[]): The name of the reviewer roles. For
specification review it is "Specification Reviewer". For screening review, the
names can be "Primary Screener" and "Screener". For review, it is "Reviewer",
"Accuracy Reviewer", "Failure Reviewer" and "Stress Reviewer". For aggregation, it
is "Aggregator". For approval, it is “Approver”. For post-mortem, it is “Post-
mortem Reviewer”
Output:
    - reviews[]: An array of reviews of some type (specification review scorecard,
screening review scorecard, main review scorecard, aggregation review scorecard,
post-mortem scorecard or approval scorecard)

Search the reviewIds using Resource Management:
    Search for resources with resource_role_name IN resourceRoleNames[] AND
phase_id=phaseId
    Resource[] reviewers = ResourceManager.search(filter)
    Create an array of reviewerIds from reviewers[] array using reviewer.getId()
    Use Review Management to search for review with reviewer id IN [reviewerIds
array]
    Review[] reviews = ReviewManager.search(filter)
```

1.3.13 *Check screening type [Routine]*

Corresponds to PhasesHelper#isScreeningManual() method.

```
Get phase attribute "Manual Screening":
    String manualScreening = phase.getAttribute("Manual Screening")
    If the return value is "Yes" screening type is manual, otherwise it is
automatic.
```

1.3.14 *Get the scorecard minimum score using a review [Routine]*

Corresponds to PhasesHelper#getScorecardMinimumScore() method.

```
Get scorecardId from one the review instance: scorecardId = review.getScorecard()
Use Scorecard Management to get the Scorecard instance:
scorecard = ScorecardManager.getScorecard(scorecardId)
    minScore = scorecard.getMinScore()
```

1.3.15 *Get the winning submitter [Routine]*

Corresponds to PhasesHelper#getWinningSubmitter() method.

```
Get project with project manager:
    project = projectManager.getProject(projectId)
Get winner ID from the project property
    winnerId = (String) project.getProperty("Winner External Reference ID")
If winnerId is null then return null
Lookup "Submitter" role ID using ResourceRoleLookupUtility (submitterRoleId)
Use ResourceManager to search for resource with
    resource_role_id = submitterRoleId
    projectId = <value of project id routine parameter>
    extension_property_name = "External Reference ID"
    extension_property_value = winnerId
Return the first found resource or null if found nothing
```

1.3.16 Get worksheet [Routine]

Corresponds to PhasesHelper#getWorksheet() method. It accepts role name and phase ID. This method is used for retrieving Aggregation and Final Review worksheets.

```
Use "Search all reviews for a phase based on resource roles" routine with the
given aggregation phase ID and role
If no reviews found, return null
If more than one review is found, throw PhaseHandlingException
Return the found review
```

1.3.17 Search all ScreeningTasks for the project [Routine]

This routine is removed in the version 1.6.1.

1.3.18 Search all active submissions with specific submission type for the project [Routine]

```
Corresponds to PhasesHelper#searchActiveSubmissions() method. Parameter
submissionTypeName:String was added to the method.
    Lookup submission status id for "Active" status using
SubmissionStatusLookupUtility class (activeStatusId)
    Lookup submission type ID by submissionTypeName using
SubmissionTypeLookupUtility:
        submissionTypeId = SubmissionTypeLookupUtility.lookUpId(connection,
submissionTypeName)
    Use Deliverable Management to search for submission with: ProjectId =
phase.getProject().getId()
    AND submission status id = activeStatusId
    AND submission type ID = submissionTypeId (use SubmissionFilterBuilder.
createSubmissionTypeIdFilter(submissionTypeId) filter)
```

1.3.19 Insert Post-Mortem Phase

```
Corresponds to PhasesHelper#insertPostMortemPhase() method.
Get current project by phase.getProject()
    (currentPrj)
Get "Post-Mortem Required" property of the project:
    currentPrj.getProperty("Post-Mortem Required") (postMortemRequired)
If postMortemRequired is not equal to "true" then return (no need to insert a
post-mortem phase)
Get current project phases
Phase[] phases = currentPrj.getAllPhases(). The phases are already sorted in the
right order
If phase with type "Post-Mortem" already exists in phases, return.
Find the current phase in the array using phase.getId() (currentPhase)
Lookup phase type with name "Post-Mortem"
Lookup phase status with name "Scheduled"
Create a Phase instance of current project with "Scheduled" status and type "Post-
Mortem" respectively. Read the length of this phase from the configuration.
Set "Reviewer Number" and "Scorecard ID" attributes of the created phase (read
these values from the configuration)
```

```
Insert the new phase into "phases" array, after the currentPhase.
Update the phases in currentPrj instance
        Using currentPrj.addPhase() methods
Update the phases into persistence using Phase Management component
        Using PhaseManager.updatePhases()
```

1.3.20 Insert Approval Phase

Corresponds to PhasesHelper#insertApprovalPhase() method.

```
Get current project by phase.getProject()
        (currentPrj)
Get current project phases
        Phase[] phases = currentPrj.getAllPhases(). The phases are already sorted
in the right order
Find the current phase in the array using phase.getId() (currentPhase)
Lookup phase type with name "Approval"
Lookup phase status with name "Scheduled"
Create a Phase instance for current project, "Scheduled" status, type "Approval"
respectively. Read the length of this phase from the configuration.
Set "Reviewer Number" and "Scorecard ID" attributes of the created phase (take
values from the previous approval phase or read from the configuration if it’s the
first approval phase in the project)
Insert the new phase into "phases" array, after the currentPhase.
Update the phases in currentPrj instance
        Using currentPrj.addPhase() method
Update the phases into persistence using Phase Management component
        Using PhaseManager.updatePhases()
```

1.3.21 Insert final fix/review cycle

Corresponds to PhasesHelper#insertFinalFixAndFinalReview() method.

```
Get current project by phase.getProject()(currentPrj)
Get current project phases: Phase[] phases = currentPrj.getAllPhases(). The phases
are already sorted in the right order
Find the current Final Review phase in the array
       use phase.getId() (currentPhase)
Lookup phase types with names "Final Fix" and "Final Review"
Lookup phase status with name "Scheduled"
Create two Phase instances of Current project, "Scheduled" status, types "Final
Fix" and "Final Review" respectively
Insert the two new phase into "phases" array, after the currentPhase.
Update the phases in currentPrj instance Using currentPrj.addPhase() method
Update the phases into persistence with Phase Management component using
PhaseManager.updatePhases()
```

1.3.22 Insert specification submission/specification review cycle

Corresponds to PhasesHelper#insertSpecSubmissionAndSpecReview() method. *(For developers: this new method is very similar to PhasesHelper#insertFinalFixAndFinalReview(). Please use it as a reference).*

```
Get current project by phase.getProject()(currentPrj)
Get current project phases: Phase[] phases = currentPrj.getAllPhases(). The phases
are already sorted in the right order
Find the current "Specification Review" phase in the array
       use phase.getId() (currentPhase)
Lookup phase types with names "Specification Submission" and "Specification
Review"
Lookup phase status with name "Scheduled"
Create two Phase instances of Current project, "Scheduled" status, types
"Specification Submission" and "Specification Review" respectively
```

```
Insert the two new phase into "phases" array, after the currentPhase.
Update the phases in currentPrj instance Using currentPrj.addPhase() method
Update the phases into persistence with Phase Management component using
PhaseManager.updatePhases()
```

1.3.23 Check if the given phase is the first phase in the project [Routine]

Corresponds to PhasesHelper#isFirstPhase() method.

```
    Input: phase:Phase – the phase to be checked
    Get all phases for the project:
        Phase[] phases = phase.getProject().getAllPhases()
    Get index of the input phase in phases array. (phaseIndex)
    While phaseIndex > 0 and phases[phaseIndex-1].calcStartDate() equals to
phases[phaseIndex].calcStartDate() do:
        phaseIndex--;
    If phaseIndex == 0 then this is the first phase in the project
```

1.3.24 Check if all parent projects are completed [Routine]

Corresponds to PhasesHelper#areParentProjectsCompleted() method.

```
    Input: projectId:long
    ProjectLink[] links=projectLinkManager.getDestProjectLinks(projectId);
    For each link with type.isAllowOverlap()==false
        Get parent project: parentProject = link.getDestProject();
        If parentProject.getProjectStatus().getId() != 1 return false (not active)
        phasesProject = phaseManager.getPhases(parentProject.getId());
        For each phase from phasesProject.getAllPhases()
            If phase status ID != 3 (not closed) return false
    return true
```

1.3.25 Extend project start time by the given time [Routine]

This routine was removed from RegistrationPhaseHandler in the version 1.4.

1.3.26 Specification Submission Phase Handler

canPerform() - Can start (checkPhaseStatus() returned true)

Specification Submission phase can start as soon as:
- The dependencies are met
- Phase start date/time is reached (if specified)
- This is NOT the first phase in the project OR all parent projects are completed

If all conditions are met this method must return OperationCheckResult.SUCCESS.
Otherwise a new OperationCheckResult instance with one of the following
explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Phase start time is not yet reached" (created in subroutine)
"Not all parent projects are completed"

Check if the dependencies are met
    Use the routine "**Check if all dependencies of a phase have correct status**".

If phase.calcStartDate() is not null
    Use the routine "Check if a phase start time is reached".

Check if the current phase is the first phase in the project
    Use "Check if the given phase is the first phase in the project" routine

Checking if all parent projects are completed:
    Use "Check if all parent projects are completed" routine

canPerform() - Can stop (checkPhaseStatus() returned false)

Specification Submission phase can stop when:
- *The dependencies are met*
- *If one active submission with "Specification Submission" type exists*

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Specification submission doesn't exist"

The dependencies are met
Use the routine **"Check if all dependencies of a phase have correct status"**.

Check if one specification submission exists
Use "Search all active submissions with specific submission type for the project" routine with submission type = "Specification Submission" to get submissions[].
If submissions array is empty, return false
If submissions array contains more than 1 element, throw PhaseHandlingException
Return true

perform() – Start (checkPhaseStatus() returned true)

Call sendEmail(phase) method of AbstractPhaseHandler

perform() – Stop (checkPhaseStatus() returned false)

Call sendEmail(phase) method of AbstractPhaseHandler

1.3.27 Specification Review Phase Handler

canPerform() - Can start

Specification Review phase can start as soon as:
- *The dependencies are met*
- *Phase start date/time is reached (if specified)*
- *If one active submission with "Specification Submission" type exists*

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Phase start time is not yet reached" (created in subroutine)
"Specification submission doesn't exist"

Check if the dependencies are met
Use the routine **"Check if all dependencies of a phase have correct status"**.

If phase.calcStartDate() is not null
Use the routine "Check if a phase start time is reached".

Check if one specification submission exists
Use the approach described for "**Can stop**" in the previous section of CS

canPerform() - Can stop

Specification Review phase can stop when:

- *The dependencies are met*
- *The specification submission exists*
- *The specification review is done by Specification Reviewer*

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Specification submission doesn't exist"
"Specification review is not yet committed"

The dependencies are met
    Use the routine "**Check if all dependencies of a phase have correct status**".

Check the specification review is done by Specification Reviewer
    Use "**Search all active submissions with specific submission type for the project**" routine with submission type = "Specification Submission" to get submissions[].
    If submissions array is empty, return false
    If submissions array contains more than 1 element, throw PhaseHandlingException
    submission = submissions[0];
    Search the Specification Review worksheet using the "Search all reviews for a phase based on resource roles" routine with phase_id = phase.getId(), resource_role_name = "Specification Reviewer" and submission_id = submission.getId(). (reviews[])
    If reviews array is empty, return false
    If reviews array contains more than 1 element, throw PhaseHandlingException
    review = reviews[0];
    If review.isCommitted(), the specification review is done

perform() – Start

Call sendEmail(phase,map) method of AbstractPhaseHandler
- supply the map with the following:
- N_SPECIFICATION_REVIEWERS: the number of specification reviewers assigned

To get N_SPECIFICATION_REVIEWERS:
Use "Search resource based on resource role name and phase id" with ResourceRoleName: "Specification Reviewer" and phaseId:(phase.getId())

perform() – Stop

When Specification Review phase is stopping, if the specification review is rejected, another specification submission/specification review cycle is inserted and status of the specification submission is changed to "Failed Review".

Find out if the specification review is rejected
    Use "**Search all active submissions with specific submission type for the project**" routine with submission type = "Specification Submission" to get submissions[].
    If submissions.length != 1, throw PhaseHandlingException
    submission = submissions[0];
    Search the Specification Review worksheet using the "Search all reviews for a phase based on resource roles" routine with phase_id = phase.getId(), resource_role_name = "Specification Reviewer" and submission_id = submission.getId(). (reviews[])
    If reviews.length != 1, throw PhaseHandlingException
    review = reviews[0];
    If not review.isCommitted(), throw PhaseHandlingException
    Get all review comments
        review.getAllComments() (comments[])
    Search the result array and find the comment with

```
        comment.getCommentType().getName()="Specification Review Comment"
(srComment)
    Check if the comment is approve or reject
        If srComment.getExtraInfo()="Rejected" then the specification review is
rejected.
        Else if it's not "Approved" or "Accepted", throw PhaseHandlingException
```

If specification review was rejected, update specification submission status

```
    Lookup submission status id for "Failed Review" status using
SubmissionStatusLookupUtility class (failedReviewStatusId)
    Set "Failed Review" status to submission:
        submission.setSubmissionStatus(failedReviewStatusId)
    Update submission in persistence:
        getManagerHelper().getUploadManager().updateSubmission(submission,
operator)
```

Insert another specification submission/specification review cycle *(For developers: FinalReviewPhaseHandler#checkFinalReview method performs very similar operations, please use it as a reference, but don't use* ContestDependencyAutomation*)*

```
    Use the routine "Insert specification submission/specification review cycle"
    Search for old "Specification Reviewer" resource (associated with the current
phase). Clone it and associate a cloned resource with the new "Specification
Review" phase. Please use PhasesHelper#createAggregatorOrFinalReviewer as a
reference to implement this step.
    Persist cloned resource with ResourceManager#updateResource().
```

Call sendEmail(phase,map) method of AbstractPhaseHandler
- supply the map with the following:
- RESULT: The result of specification review: rejected/approved

To get RESULT: See "**Find out if the specification review is rejected**" step above.

1.3.28 Registration Phase Handler

canPerform() - Can start

Registration phase can start as soon as:
- The dependencies are met
- Phase start date/time is reached (if specified)
- This is NOT the first phase in the project OR all parent projects are completed

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Phase start time is not yet reached" (created in subroutine)
"Not all parent projects are completed"

Check if the dependencies are met
    Use the routine "**Check if all dependencies of a phase have correct status**".

If phase.calcStartDate() is not null
    Use the routine "Check if a phase start time is reached".

Check if the current phase is the first phase in the project
    Use "Check if the given phase is the first phase in the project" routine

Checking if all parent projects are completed:
    Use "Check if all parent projects are completed" routine

canPerform() - Can stop

Registration phase can stop when all these conditions met:
    - The dependencies are met
    - The period has passed
    - The number of registrations meets the required number or there are no registrations.

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Phase end time is not yet reached"
"Not enough registrants"

The dependencies are met
    Use the routine "**Check if all dependencies of a phase have correct status**".

The period has passed
    Use the "Check if a phase end time is reached" routine.

For "The number of registrations meets the required number or there are no registrations":
    Get the number of required registrations using the "Registration Number" phase attribute:
        String regNumber = phase.getAttribute("Registration Number")
    Use Resource Management to search for resources with
        Role equals "Submitter"
        Project id equals phase.getProject().getId()
    Compare the number of resource returned from the search with "regNumber" to see if the condition met. Condition is also met when the number of registrations is equal to 0.

perform() - Start

Call sendEmail(phase,map) method of AbstractPhaseHandler

perform() - Stop

If there are no registrants, use the routine "**Insert Post-Mortem phase**".
Call sendEmail(phase,map) method of AbstractPhaseHandler
- supply the map with the following:
- N_REGISTRANTS: the number of registrants
- REGISTRANT: a list containing 1 element for each registrant. Each registrant is represented as a map with the following keys: REGISTRANT_HANDLE, REGISTRANT_RELIABILITY, REGISTRANT_RATING

To get N_REGISTRANTS and REGISTRANT:
    Use Resource Management to search for resources with
        Role equals "Submitter"
        Project id equals phase.getProject().getId()

1.3.29 Submission Phase Handler

canPerform() - Can start

- The dependencies are met
- Check phase start date time if exist

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)

"Phase start time is not yet reached" (created in subroutine)

Check the dependencies are met
    Use the routine "Check if a phase start time is reached".

If phase.calcStartDate() is not null
    Submission phase can start as soon as the start time is reached

canPerform() - Can stop

Submission phase can stop when all of these conditions met:
- *The dependencies are met*
- *The phase end time is reached*

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Phase end time is not yet reached"

The dependencies are met
    Use the routine **"Check if all dependencies of a phase have correct status"**.

The phase end time is reached
    Use the "Check if a phase end time is reached" routine.

perform() - Start

Call sendEmail(phase,map) method of AbstractPhaseHandler

perform() - Stop

    If there are no submissions with "Contest Submission" type, use the routine **"Insert Post-Mortem phase"**.
Call sendEmail(phase,map) method of AbstractPhaseHandler
- supply the map with the following:
- N_SUBMITTERS : the number of submitters
- SUBMITTER: a list containing 1 element for each submitter. Each submitter is represented as a map with the following keys: SUBMITTER_HANDLE, SUBMITTER_RELIABILITY, SUBMITTER_RATING

To get SUBMITTER array:
- Use Deliverable Management to search for submission with: ProjectId = phase.getProject().getId() AND submission type ID = looked up "Contest Submission" type ID
- For each Submission returned, get the owner: submission.getUpload().getOwner()
- Use Resource Management to get submitter (submitter is a resource):
            submitter = ResourceManager.getResource(submitterId)

1.3.30 Screening Phase Handler

canPerform() - Can start

Screening can start as soon as the dependencies are met.
    Use the routine **"Check if all dependencies of a phase have correct status"**.
If phase.calcStartDate() is not null
    Use the routine "Check if a phase start time is reached".
    Also the number of active submissions with "Contest Submission" type must be greater than 0.

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Phase start time is not yet reached" (created in subroutine)
"Contest has no submissions"

canPerform() - Can stop

Screening can stop when:
- The dependencies are met
- If it's primary screening mode, all submissions that passed auto-screening have one screening scorecard committed.
- If it's individual screening mode, the submission that passed auto-screening has one screening scorecard committed.

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Not all primary screening scorecards are committed (see submission with ID=XXX)"
"Individual screening scorecard is not committed"
"Unknown screening mode."

The dependencies are met
Use the routine "**Check if all dependencies of a phase have correct status**".

Determine the screening mode (Primary or individual):
Lookup resource role id for "**Primary Screener**" and "**Screener**" using ResourceRoleLookupUtility (primaryScreenerId and screenerId)
Use Resource Management to search for resource with the resource role id = "primaryScreenerId" and "screenerId"
If search for "Primary Screener" returns at least one row, the screening mode is primary
If search for "Screener" returns at least one row, the screening mode is individual.

If it's primary screening mode, all submissions that passed auto-screening have one screening scorecard committed.

```
    Search all screening scorecard for the current phase:
    Use the "Search all reviews for a phase base on resource roles" routine.
        Input:
            - phaseId=phase.getId() (current phase id)
            - resourceRoleName="Primary Screener"
        Output:
            - screenReviews[]
```

Search all active submissions with "Contest Submission" type for current project using "Search all active submissions with specific submission type for the project" routine (submissions[])

```
    Number of reviews must be equal to the number of submissions.
    Check if every submission has an associated screening scorecard
        For each submission
            Look for its matching scorecard using submission.getId()=screenReview.getSubmission()
        End For

    Check if all of the screening scorecards are committed
        Walk the screenReviews[] and check screenReview.isCommitted()
```

```
If it's individual screening mode, the submission that passed auto-screening has
one screening scorecard committed.
    Search the submission for the current individual screening phase
    Search the submitter for this phase
        Use the "Search resource based on resource role names and phase id"
routine.
        Input:
            - resourceRoleNames: "Submitter"
            - phaseId: current phase id (phase.getId())
        Output:
            - resources[]
        If resources[].length <> 1, throw PhaseHandlingException for inconsistence
data.
        submitter = resource[0]

    Search the submission base on the submitter (submissions[])
        Use Deliverable Management to search for submission with: ProjectId =
phase.getProject().getId()
        AND resource Id = submitter.getId()
        AND submission type ID = <looked up ID for "Contest Submission">
        If submissions[].length <> 1, throw PhaseHandlingException for
inconsistence data.
        submission=submissions[0]

    Use the "Search all reviews for a phase base on resource roles" routine.
        Input:
            - phaseId=phase.getId() (current phase id)
            - resourceRoleName="Screener"
        Output:
            - screenReviews[]
        For individual screening mode, each phase will be associated with one
submission. So if screenReviews[].length <> 1,
        throw PhaseHandlingException for inconsistence data.
        screenReview=screenReviews[0]

    Check if the screening review is associated with the submission
        Use submission.getId()=screenReview.getSubmission()

    Check if the screening scorecards are committed
        Check screenReview.isCommitted()
```

perform() - Start

Call sendEmail(phase,map) method of AbstractPhaseHandler
- supply the map with the following:
- NEED_PRIMARY_SCREENER : set to 1 if primary screener is not available, 0 otherwise
- SUBMITTER: a list containing 1 element for each submitter. Each submitter is represented as a map with following key SUBMITTER_HANDLE, SUBMITTER_RELIABILITY, SUBMITTER_RATING

To get NEED_PRIMARY_SCREENER:
- Use Resource Management to search for resource with the resource role id = "primaryScreenerId" and "screenerId"
- If this is null then NEED_PRIMARY_SCREENER is set to 1
-

To get SUBMITTER array:
- Use Deliverable Management to search for submission with: ProjectId = phase.getProject().getId() AND submission type = "Contest Submission"
- - For each Submission returned, get the owner: submission.getUpload().getOwner()

```
- -   Use Resource Management to get submitter (submitter is a resource):
           submitter = ResourceManager.getResource(submitterId)
```

perform() – Stop

When screening is stopping:
- All submissions with failed screening scorecard scores should be set to the status "Failed Screening"
- Screening score for the all submissions will be saved to the submitters' resource properties named "Screening Score".

```
    Search all submissions with "Contest Submission" type for current project
using "Search all active submissions with specific submission type for the
project" routine (submissions[]):

    Search all screening scorecard for the current phase:
        Use the "Search all reviews for a phase base on resource roles" routine.
        Input:
            - phaseId=phase.getId() (current phase id)
            - resourceRoleNames="PrimaryScreener", "Screener"
        Output:
            - screenReviews[]

    Throw PhaseHandlingException if the number of submissions is not equal to the
number of screening scorecards.

    Get the screening minimum score:
        Use the "Get the scorecard minimum score using a review" routine.
        (With the first instance in screenReviews[])

    For each submission
        Find the matching screening review using submission.getId() =
screeningReview.getSubmission()
        Get submission's screening score: screeningScore =
screeningReview.getScore()

        Store the screening score for the submission
        Set screening score:
submission.setScreeningScore(Double.valueOf(String.valueOf(screeningScore)));

        If screeningScore<screening minimum score
            Set submission status to "Failed Screening"
                Lookup submission status with name "Failed Screening" using
UploadManager (failedScreeningStatusId)
                Set the status for the submission using
submission.setSubmissionStatus()
                Update submission using Deliverable Management:
UploadManager.updateSubmission()
        End If
    End for
```

If there are no passing screenings, use the routine "**Insert Post-Mortem phase**".

Call sendEmail(phase,map) method of AbstractPhaseHandler
- supply the map with the following:
- SUBMITTER: a list containing 1 element for each submitter. Each submitter is represented as a map with following key SUBMITTER_HANDLE, SUBMITTER_SCORE, SUBMITTER_RESULT
- NO_SCREENING_PASS: 0 if no submissions passed screening, 1 otherwise

To get SUBMITTER array:
- Use Deliverable Management to search for submission with: ProjectId = phase.getProject().getId() AND submission type = "Contest Submission"

```
- For each Submission returned, get the owner: submission.getUpload().getOwner()
- Use Resource Management to get submitter (submitter is a resource):
       submitter = ResourceManager.getResource(submitterId)
- To get the submitter score: get from submission.getScreeningScore()
```

1.3.31 Review Phase Handler

canPerform() - Can start

Review can start as soon as the dependency phases have ended

    Use the routine "Check if all dependencies of a phase have correct status".

If phase.calcStartDate() is not null

    Use the routine "Check if a phase start time is reached".
    Check if count of active submissions with "Contest Submission" type is above 0 using "Search all active submissions with specific submission type for the project" routine

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Phase start time is not yet reached" (created in subroutine)
"No submissions that passed screening"

canPerform() - Can stop

- All dependencies are met
- All active submissions with "Contest Submission" type *have one review scorecard from each reviewer for the phase*
- All test case reviewers have one test case upload.

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Not all review scorecards are committed"
"Not all test cases are uploaded (see reviewer with XXX role)"

All active submissions have one review scorecard from each reviewer for the phase

```
    Search all "Active" submissions with "Contest Submission" type for current project using "Search all active submissions with specific submission type for the project"
    Search for reviewers with resource_role_name = "Reviewer" AND phase id = phase.getId() using "Search resource based on resource role names and phase id" routine
    Search all review scorecards for the current phase with resource_role_name = "Reviewer" AND phase id = phase.getId() using "Search all reviews for a phase based on resource roles" routine
    If number of reviewers == 0 return false.
    Get "Reviewer Number" attribute from the phase if exists. If value of this attribute > actual number of reviewers, return false.
    For each submission
        Match the submission with its reviews by using submission.getId() and review.getSubmission()
        Each submission should have the number of review scorecard = the number of reviewers.
        Match the submission's reviews with the reviewer for consistence check by using
            review.getAuthor() = reviewer.getId()
        All reviews must be committed.
    End For
```

All test case reviewers have one test case upload.

```
    Find all test case reviewer IDs using InFilter search of Resource Management:
        Lookup resource role id using ResourceRoleLookupUtility
            "Accuracy Reviewer" - accuracyReviewerId
            "Failure Reviewer" - failureReviewerId
            "Stress Reviewer" - stressReviewerId
        Search for resource with: resource_role_id IN [accuracyReviewerId,
failureReviewerId, stressReviewerId] AND
        phase id = phase.getId() (reviewerIds)
    If number of reviewers = 0, the condition is met.
    Use UploadManager to search for uploads with the resourceId IN [reviewerIds
array]
    For each reviewerId
        To see if each reviewer has one test case upload
            Match reviewer and his upload by reviewerId = upload.getOwner()
    End For
```

perform() – Start

Call sendEmail(phase,map) method of AbstractPhaseHandler

- supply the map with the following:
- N_REVIEWERS: the number of available reviewers
- N_REQUIRED_REVIEWERS: the number of reviewers required
- NEED_REVIEWER: if N_REVIEWERS is less than REQUIRED_REVIEWERS
- SUBMITTER: a list containing 1 element for each submitter. Each submitter is represented as a map with following key SUBMITTER_HANDLE, SUBMITTER_RELIABILITY, SUBMITTER_RATING

To get SUBMITTER array:

- Use Deliverable Management to search for submission with: ProjectId = phase.getProject().getId() AND submission type = "Contest Submission"
- For each Submission returned, get the owner: submission.getUpload().getOwner()
- Use Resource Management to get submitter (submitter is a resource):

```
            submitter = ResourceManager.getResource(submitterId)
To get N_REVIEWERS:
Use the "Search all reviews for a phase base on resource roles" routine
        Input:
            - resourceRoleName: "Reviewer"
            - phaseId: reviewPhase.getId()
        Output:
            - reviews[]
```

perform() – Stop

Initial score for the all passed screening submissions will be calculated and saved to the submitters' resource properties named "Initial Score".
Submissions that passed screening will have status "Active" (instead of "Failed Screening"). For Studio competitions we need to perform the same operations as for AppealsResponse phase - populate places and update project with placement details.

```
    Search all "Active" submissions with "Contest Submission" type for the current
project using "Search all active submissions with specific submission type for the
project" routine
    Search the reviewers with resource_role_name = "Reviewer" AND phase id =
phase.getId() using "Search resource based on resource role names and phase id"
routine
    Search all review scorecard for the current phase with resource_role_name =
"Reviewer" AND phase id = phase.getId() using "Search all reviews for a phase
based on resource roles" routine
    For each submission
        Match the submission with its reviews by using submission.getId() and
review.getSubmission()
```

```
        Each submission should have the number of review scorecard = the number of
reviewers.
        Get the scores from the reviews scorecard using review.getScore()
        Use Review Score Aggregator to calculate aggregation of initial score
(aggScore)
        Set initial score to the submission using Submission#setInitialScore()
        Update submission in persistence using UploadManager#updateSubmission()
    End For

For Studio Competitions:
        Get all prizes with type "Submission Prize" for the project, order prizes
   by amount.
For each submission
        Match the submission with its reviews by using submission.getId() and
review.getSubmission()
        Each submission should has the number of review scorecard = the number of
reviewers.
        Get the scores from the reviews scorecard using review.getScore()

        Get submitterId: submitterId = submission.getUpload().getOwner()
        Use Resource Management to get submitter (submitter is a resource):
            submitter = ResourceManager.getResource(submitterId)
        Set "Final Score" to submission:
submission.setFinalScore(Double.valueOf(aggScore + ""))
        Break ties if exist: Submission with earlier upload date gets a higher
placement.
        Set "Placement" to submission: submission.setPlacement(new
Long(placement))
        Set prize for submission if placement is less then number of prizes.


        If reviewScore < minScore and submission status is "Active"
            Set the status for the submission to failedReviewStatus using
submission.setSubmissionStatus()
        Else
            If placement <> 1
                Set the status for the submission to noWinStatus using
submission.setSubmissionStatus()
            End If
        End If

        Update submission using Deliverable Management:
UploadManager.updateSubmission()

        Store the winning submitter in winningSubmitter
        Store the runner up submitter in runnerUpSubmitter

    End For

Set project properties to store the winner and the runner up:
    Get projectId
        projectId = phase.getProject().getId()
    Gets the project instance using Project Management
        project = ProjectManager.getProject(projectId)

    Get the external reference id of the winner and the runner up, store in the
resource's property
    named "External Reference ID"
        winnerExtId = winningSubmitter.getProperty("External Reference ID")
        runnerExtId = runnerUpSubmitter.getProperty("External Reference ID")

    Set the winner for the project in the project property named "Winner External
Reference ID"
```

```
        project.setProperty("Winner External Reference ID", winnerExtId)
    Set the runner up for the project in the project property named "Runner-up
External Reference ID"
        project.setProperty("Runner-up External Reference ID", runnerExtId)
```

Call sendEmail(phase,map) method of AbstractPhaseHandler

- supply the map with the following:
- SUBMITTER: a list containing 1 element for each submitter. Each submitter is represented as a map with following key SUBMITTER_HANDLE, SUBMITTER_SCORE

```
To get SUBMITTER array:
- Use Deliverable Management to search for submission with: ProjectId =
phase.getProject().getId()
- For each Submission returned, get the owner: submission.getUpload().getOwner()
-   Use Resource Management to get submitter (submitter is a resource):
            submitter = ResourceManager.getResource(submitterId)
- To get the SUBMITTER_SCORE use submission.getInitialScore();
```

1.3.32 Appeals Phase Handler

canPerform() - Can start

```
- Appeals can start as soon as the dependency phases have ended
    Use the routine "Check if all dependencies of a phase have correct status".
If phase.calcStartDate() is not null
    Use the routine "Check if a phase start time is reached".
```

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Phase start time is not yet reached" (created in subroutine)

canPerform() - Can stop

```
Appeals can start when:
- All dependencies are met.
- The period has passed.
    Use the "Check if a phase end time is reached" routine.
    Check if all submitters have elected to end appeals
      Get all submitters active in this project
- OR Appeals can be closed early (see PhasesHelper#canCloseAppealsEarly()):
    Lookup ID of "Submitter" role using ResourceRoleLookupUtility
(submitterRoleId)
    Use ResourceManager to find resources (earlyAppealCompletionsSubmitters) with
        resource_role_id = submitterRoleId
        AND project_id = phase.getProject().getId()
        AND extension_property_name = "Appeals Completed Early"
        AND extension_property_value = "Yes"
    Search all "Active" submissions with "Contest Submission" type for the current
project using "Search all active submissions with specific submission type for the
project" routine
    For each found submission
        If earlyAppealCompletionsSubmitters doesn't contain a resource with ID
equal to submission.getUpload().getOwner()
            The condition is not met (appeals cannot close early).
        End If
    End For
```

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Phase end time is not yet reached and appeals cannot be closed early"

perform() – Start and Stop

Call sendEmail(phase,map) method of AbstractPhaseHandler
- supply the map with the following:
- SUBMITTER: a list containing 1 element for each submitter. Each submitter is represented as a map with following key SUBMITTER_HANDLE, SUBMITTER_SCORE
- To get SUBMITTER array:
- Use Deliverable Management to search for submission with: ProjectId = phase.getProject().getId() AND submission type = "Contest Submission"
- For each Submission returned, get the owner: submission.getUpload().getOwner()
- Use Resource Management to get submitter (submitter is a resource): submitter = ResourceManager.getResource(submitterId)
- To get the SUBMITTER_SCORE, submission.getInitialScore();

1.3.33 *Appeals Response Phase Handler*

canPerform() - Can start

```
*Appeals Response can start as soon as the dependency phases have ended*
    Use the routine "Check if all dependencies of a phase have correct status".
*If phase.calcStartDate() is not null*
    Use the routine "Check if a phase start time is reached".
```

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Phase start time is not yet reached" (created in subroutine)

canPerform() - Can stop

Appeals Response can end when:
The dependencies are met
All appeals are resolved.

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Not all appeals are resolved"

```
The dependencies are met
    Use the routine "Check if all dependencies of a phase have correct status".

All appeals are resolved
    Find appeals:
        Go back to the nearest Review phase using "Locating phases" routine
    Get all reviews:
        Use "Search resource based on resource role names and phase id" routine
with current phase id and role IN ("Reviewer", "Accuracy Reviewer", "Failure
Reviewer", "Stress Reviewer") to get reviewers
    For each review from reviews
        Use review.getAllComments() to get its comments
        Use comment.getCommentType().getName() to find comments with type "Appeal"
and "Appeal Response"
    End For
    If the numbers of Appeal and Appeal Response comments are equal, then all
appeals are resolved.
```

perform() – Start

Call sendEmail(phase,map) method of AbstractPhaseHandler
- supply the map with the following:
- SUBMITTER: a list containing 1 element for each submitter. Each submitter is represented as a map with following key SUBMITTER_HANDLE, SUBMITTER_PRE_APPEALS_SCORE

See map values retrieval details below.

perform() – Stop

When Appeals Response is stopping, all submissions with failed review scores should be set to the status Failed Review.
Overall score for the passing submissions should be calculated and saved to the submitters' resource properties together with their placements. The winner and runner-up should be populated in the project properties.
Submissions that do not win should be set to the status Completed Without Winning.

Update failed review submission status to "Failed Review":

Search all "Active" submissions with "Contest Submission" type for the current project using "**Search all active submissions with specific submission type for the project**" routine (submissions[])

Locate the nearest previous Review Phase (reviewPhase)
Using "Locating phases" routine.

Search all review scorecards for the review phase
Use the "Search all reviews for a phase base on resource roles" routine
Input:
- resourceRoleName: ("Reviewer", "Accuracy Reviewer", "Failure Reviewer", "Stress Reviewer")
- phaseId: reviewPhase.getId()

Output:
- reviews[]

Get minimum review score (minScore)
Use the "Get the scorecard minimum score using a review"
(with the first instance in reviews[] array)

Use Review Score Aggregator to calculate aggregation of final score (aggScore) and placement (placement)

Lookup submission status for "Failed Review" using UploadManager class. (failedReviewStatus)
Lookup submission status for "Completed Without Win" status using UploadManager class. (noWinStatus)

For each submission
Match the submission with its reviews by using submission.getId() and review.getSubmission()
Each submission should has the number of review scorecard = the number of reviewers.
Get the scores from the reviews scorecard using review.getScore()

Get submitterId: submitterId = submission.getUpload().getOwner()
Use Resource Management to get submitter (submitter is a resource):
submitter = ResourceManager.getResource(submitterId)
Set "**Final Score**" to submission: submission.setFinalScore(Double.valueOf(aggScore + ""))
Break ties if exist: Submission with earlier upload date gets a higher placement.
Set "**Placement**" to submission: submission.setPlacement(new Long(placement))

```
        If aggScore < minScore and submission status is "Active"
            Set the status for the submission to failedReviewStatus using
submission.setSubmissionStatus()
        Else
            If placement <> 1
                Set the status for the submission to noWinStatus using
submission.setSubmissionStatus()
            End If
        End If

        Update submission using Deliverable Management:
UploadManager.updateSubmission()

        Store the winning submitter in winningSubmitter
        Store the runner up submitter in runnerUpSubmitter

    End For

Set project properties to store the winner and the runner up:
    Get projectId
        projectId = phase.getProject().getId()
    Gets the project instance using Project Management
        project = ProjectManager.getProject(projectId)

    Get the external reference id of the winner and the runner up, store in the
resource's property
    named "External Reference ID"
        winnerExtId = winningSubmitter.getProperty("External Reference ID")
        runnerExtId = runnerUpSubmitter.getProperty("External Reference ID")

    Set the winner for the project in the project property named "Winner External
Reference ID"
        project.setProperty("Winner External Reference ID", winnerExtId)
    Set the runner up for the project in the project property named "Runner-up
External Reference ID"
        project.setProperty("Runner-up External Reference ID", runnerExtId)

    Update the project
        Use ProjectManager.updateProject() method with updateReason="Update the
winner and runner up."

If there are no passes, use the routine "Insert Post-Mortem phase".

Call sendEmail(phase,map) method of AbstractPhaseHandler
-  supply the map with the following:
-  SUBMITTER: a list containing 1 element for each submitter. Each submitter is
   represented as a map with following key SUBMITTER_HANDLE,
   SUBMITTER_PRE_APPEALS_SCORE, SUBMITTER_POST_APPEALS_SCORE, SUBMITTER_RESULT
-  To get SUBMITTER array:
-  Use Deliverable Management to search for submission with: ProjectId =
phase.getProject().getId() AND submission type = "Contest Submission"
-  For each Submission returned, get the owner: submission.getUpload().getOwner()
-    Use Resource Management to get submitter (submitter is a resource):
            submitter = ResourceManager.getResource(submitterId)
-  To get the SUBMITTER_PRE_APPEALS_SCORE, submission.getInitialScore();
-  To get the SUBMITTER_POST_APPEALS_SCORE, submission.getFinalScore();
```

1.3.34 Aggregation Phase Handler

`canPerform() - Can start`

```
Aggregation can start as soon as:
    The dependencies are met:
        Use the routine "Check if all dependencies of a phase have correct
status".
    If phase.calcStartDate() is not null
        Use the routine "Check if a phase start time is reached".
    There is a winner:
        Use "Get the winning submitter" routine.
    There is an aggregator:
        Use "Search resource based on resource role names and phase id" routine
with "Aggregator" role. The number of found resources must be equal to 1.
```

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Phase start time is not yet reached" (created in subroutine)
"There is no winner for the project"
"There is no aggregator for the project"

canPerform() - Can stop

```
Aggregation can end when:
      The dependencies are met;
      The winning submission must have one aggregated review scorecard committed.
```

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Aggregation review scorecard is not yet committed"

```
The dependencies are met
    Use the routine "Check if all dependencies of a phase have correct status".

The winning submission must have one aggregated review scorecard committed.
    Search the aggregated review scorecard using the "Get worksheet" routine with
"Aggregator" role. If found and review.isCommitted(), the condition is met.
```

perform() – Start

```
When Aggregation is starting and Aggregation worksheet is not created, it should
be created; otherwise it should be marked uncommitted, as well as the aggregation
review comments:

    Find the previous "Aggregation" phase using "Locating phases" routine
(previousAggregationPhase)
    Use "Search resource based on resource role names and phase id" routine to
find aggregator with "Aggregator" role and phase_id = phase.getId() (aggregator)
    Throw PhaseHandlingException if aggregator cannot be found
    Get aggregator user ID:
        aggregatorUserId = aggregator.getProperty("External Reference ID")
    If previousAggregationPhase != null
        Search the aggregated review scorecard using the "Get worksheet" routine
with "Aggregator" role. (aggWorksheet)
    End If

    If aggWorksheet == null
        Create aggWorksheet:Review instance
        Set author to aggWorksheet
            aggWorksheet.setAuthor(aggregator.getId())
        Locate the last "Review" phase using "Locating phases" routine
(reviewPhase)
        Use "Search resource based on resource role names and phase id" routine to
find reviews with role IN ("Reviewer", "Accuracy Reviewer", "Failure Reviewer",
"Stress Reviewer") and phase_id = phase.getId() (reviewers)
        Use "Get the winning submitter" routine to find a winning submitter
(winningSubmitter). Throw PhaseHandlingException if not found.

        Use UploadManager to find a submissions with resource_id =
winningSubmitter.getId() AND submission type ID = <looked up ID for "Contest
Submission"> type (submissions)
        If no submissions found, throw PhaseHandlingException
        If number of found submissions > 1
            Iterate over all submissions to find one with
submission.getPlacement() == 1 (set it’s ID to winningSubmissionId)
        End If
        Else winningSubmissionId = submissions[0].getId()

        Use "Search all reviews for a phase based on resource roles" routine to
find all reviews with role IN ("Reviewer", "Accuracy Reviewer", "Failure
Reviewer", "Stress Reviewer"), phase_id = reviewPhase.getId() and submission_id =
winningSubmissionId (reviews)

        For each review from reviews
            Set scorecard from review to aggWorksheet
            Set submission from review to aggWorksheet
            Copy comments and review items with types ("Comment", "Required",
"Recommended", "Appeal", "Appeal Response", "Aggregation Comment", "Aggregation
Review Comment", "Submitter Comment", "Manager Comment") from review to
aggWorksheet
        End For

        Lookup comment type for "Aggregation Review Comment" using ReviewManager
(aggregationReviewCommentType)
        Lookup comment type for "Submitter Comment" using ReviewManager
(submitterCommentType)
        For each reviewer from reviewers
            If review is aggregator ("External Reference ID" property equals to
aggregatorUserId)
                Create empty comment with aggregationReviewCommentType and
reviewer.getId() as author. Add it to aggWorksheet.
        End For
        Create empty comment with submitterCommentType and
winningSubmitter.getId() as author. Add it to aggWorksheet.

        Use ReviewManager to create aggWorksheet in persistence (using
createReview() method)

    Else // aggregation scorecard already exists
        Set author of aggWorksheet to aggregator.getId()
        Set scorecard committed flag to false
        Get all comments from the scorecard
            comments = aggWorksheet.getAllComments()
        For each comment
            Get comment extra info:
                extraInfo = comment.getExtraInfo()
            If extraInfo is "Approved" or "Accepted"
                comment.setExtraInfo("Approving")
            Else
                Clear comment extra info: comment.setExtraInfo(null);
            End If
        End For
        Clone the aggWorksheet and persist it using ReviewManager#createReview()
    End If

Call sendEmail(phase,map) method of AbstractPhaseHandler
-  supply the map with the following:
-  N_AGGREGATOR: the number of aggregator currently assigned

To find N_AGGREGATOR
Use "Search resource based on resource role names and phase id" with resource role
name = "Aggregator" and phase_id = phase.getId()
```

perform() – Stop

```
Call sendEmail(phase,map) method of AbstractPhaseHandler
```

1.3.35 Aggregation Review Phase Handler

canPerform() - Can start

Aggregation can start as soon as the dependencies are met

```
    Use the routine "Check if all dependencies of a phase have correct status".
```

If phase.calcStartDate() is not null

```
    Use the routine "Check if a phase start time is reached".
```

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Phase start time is not yet reached" (created in subroutine)

canPerform() - Can stop

The dependencies are met;
AND The time allotted for this review has expired
OR The aggregation review is performed by two reviewers other than the aggregator, and the winning submitter

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Phase end time is not yet reached and not all aggregation reviews are complete"

```
    Locate the nearest backward Aggregation phase (aggPhase)
        Use the "Locating phases" routine.

    Search the aggregated review scorecard using the "Get worksheet" routine with
"Aggregator" role. (aggWorksheet)

    Get all comments: aggWorksheet.getAllComments() (comments[])

    Locate the nearest Review phase (reviewPhase)
        Use the "Locating phases" routine.
    Locate the nearest Aggregation phase (aggregationPhase)
        Use the "Locating phases" routine.

    Use "Search resource based on resource role names and phase id"
    Input:
        - resourceRoleNames: "Reviewer", "Accuracy Reviewer", "Failure Reviewer",
"Stress Reviewer"
        - phaseId: reviewPhase.getId()
    Output: reviewers[]

    Use "Search resource based on resource role names and phase id"
    Input:
        - resourceRoleNames: "Aggregator"
        - phaseId: aggregationPhase.getId()
    Output: aggregators[]
    If none found, throw PhaseHandlingException

    Get aggregator user ID:
        aggregatorUserId = aggregators[0].getProperty("External Reference ID")

    Locate the winning submitter (winningSubmitter) using "Get the winning submitter" routine

    For each reviewer from reviewers
        If "External Reference ID" property of reviewer equals to aggregatorUserId, then continue
        If comments doesn't contain any comment with author = reviewer.getId(), comment type = "Aggregation Review Comment" and extra info IN ("Approved", "Accepted", "Rejected"), then go to "phase end time is reached" check
    End For

    If comments contains any comment with author = winningSubmitter.getId(), comment type = "Submitter Comment" and extra info IN ("Approved", "Accepted", "Rejected"), then return true

    Use the "Check if a phase end time is reached" routine.
    If phase end time is reached
        Approve pending aggregation review
            Iterate over all comments from aggWorksheet:
                If (comment type is "Submitter Comment" or "Aggregation Review Comment") and extra info is not IN ("Approved", "Rejected")
                    Make comment approved: comment.setExtraInfo("Approved")
        Use ReviewManager#updateReview() to update aggWorksheet in persistence
    End If
```

perform() – Start

Call sendEmail(phase) method of AbstractPhaseHandler

perform() – Stop

When Aggregation Review phase is stopping, if the aggregation is rejected by anyone, another aggregation/aggregation review cycle is inserted.

```
Use similar logic in "canPerform() - Can stop" to get resources[] and comments[]
    Check comment.getExtraInfo() for each comment. Value must be "Approved"/"Accepted" or "Rejected". Otherwise, throws PhaseHandlingException
    If all comments are "Approved" or "Accepted"
        return
    Else (there is at least one reject)
        Insert new aggregation/aggregation review cycle:
        Get current project by phase.getProject()
            (currentPrj)
        Get current project phases
            Phase[] phases = currentPrj.getAllPhases(). The phases is already sorted in the right order
            Find the current phase in the array use phase.getId() (currentPhase)
        Lookup phase types for "Aggregation" and "Aggregation Review" using PhaseManager
        Lookup phase status for "Scheduled" using PhaseManager
        Create two Phase instances of
            Current project, "Scheduled" status, type "Aggregation" and
"Aggregation Review" respectively
        Insert the two new phase into "phases" array, after the currentPhase.
        Update the phases in currentPrj instance
            Using currentPrj.addPhase() methods
        Update the phases into persistence using Phase Management component
            Using PhaseManager.updatePhases()

        Search for old "Aggregator" resource (associated with the previous
aggregation phase). Clone it and associate a cloned resource with the new
aggregation phase.
        Persist cloned resource with ResourceManager#updateResource().
    End If

Call sendEmail(phase) method of AbstractPhaseHandler
```

1.3.36 *Final Fix Phase Handler*

canPerform() - Can start

```
Final Fix can start as soon as:
    The dependencies are met
        Use the routine "Check if all dependencies of a phase have correct
status".
    If phase.calcStartDate() is not null
        Use the routine "Check if a phase start time is reached".
    There is a final reviewer available for the next final review:
        Locate the next "Final Review" phase using "Locating phases" routine
(finalReviewPhase)
        Use the "Search resource based on resource role names and phase id"
routine to find all resource with "Final Reviewer" role and phase_id =
finalReviewPhase.getId().
        If found one reviewer, the condition is met
```

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Phase start time is not yet reached" (created in subroutine)
"Final Review phase cannot be located"
"There is no Final Reviewer assigned for the next Final Review phase"

canPerform() - Can stop

```
       The dependencies are met;
       The final fix has been uploaded;
```

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Final fix is not yet uploaded"

```
The dependencies are met;
    Use the routine "Check if all dependencies of a phase have correct status".

Check if the final fix has been uploaded
    Lookup upload type ID for "Final Fix" using UploadTypeLookupUtility
(uploadTypeId)
```

```
    Lookup upload status ID for "Active" using UploadStatusLookupUtility
(uploadStatusId)
    Use "Get the winning submitter" routine to get the winning submitter
(winningSubmitter)
    Use UploadManager to search for upload with: "Final Fix" type, "Active" status
and resource_id = winningSubmitter.getId()
    The result array must contain zero or one upload, throw PhaseHandlingException
otherwise.
    If result contains one upload, then the final fix is uploaded
```

perform() – Start

When Final Fix is starting and Final Review worksheet is not created, it should be created; otherwise it should be marked uncommitted. Previous final fix upload will be deleted.

```
    Find the forward and previous "Final Review" phases using "Locating phases"
routine (finalReviewPhase, previousFinalReviewPhase)
    Find the previous "Approval" phase using "Locating phases" routine
(previousApprovalPhase)

    Search for Final Reviewer using
        "Search resource based on resource role name and phase id"
        Input:
            - ResourceRoleName: "Final Reviewer"
            - phaseId: Current phase id (finalReviewPhase.getId())
        Output:
        resources[]

    If previousApprovalPhase != null
        Get the Approval worksheets:
            Search the Approval worksheets using the "Search all reviews for a
project based on resource roles" routine.
            Input:
                - projectId=phase.getProject().getId()
                - resourceRoleName="Approver"
            Output: reviews[]
        Get all project phases (phases)
        int count = 0;
        Create a list for approval worksheets (approvalWorksheets)
        For each phase from phases:
            If phase type is "Approval"
                Get "Reviewer Number" property for this phase (reviewerNumber)
                If phase is previousAprovalPhase
                    count += reviewerNumber;
                Else
                    Add from reviews[] to approvalWorksheets elements with indices
(count .. count + reviewerNumber – 1).
        End For

        finalWorksheet = new Review();
        For each approvalWorksheet from approvalWorksheets:
            Copy review items from approvalWorksheet to finalWorksheet.

    Else if previousFinalReviewPhase != null
        Get the Final Review worksheet:
            Get the Final Review worksheet using the "Get worksheet" routine with
"Final Reviewer" role (finalWorksheet).

    If finalWorksheet == null
        finalWorksheet = new Review()
        Set author: finalWorksheet.setAuthor(resources[0].getId())
        Make not committed: finalWorksheet.setCommitted(false)
        Find the "Aggregation" phase using "Locating phases" routine (aggPhase)
        If aggPhase != null
            Use the "Get worksheet" routine with "Aggregator" role (aggWorksheet)
            Copy comments and final review items with types ("Comment",
"Required", "Recommended", "Appeal", "Appeal Response", "Aggregation Comment",
"Aggregation Review Comment", "Submitter Comment", "Manager Comment") from
aggWorksheet to finalWorksheet
            Copy final review items from aggWorksheet to finalWorksheet
            finalWorksheet.setScorecard(aggWorksheet.getScorecard())
            finalWorksheet.setSubmission(aggWorksheet.getSubmission())
        Else
            Find the "Review" phase using "Locating phases" routine (reviewPhase)
            Use "Get winning submitter" routine (winningSubmitter). If not found,
throw PhaseHandlingException
            Use UploadManager to find submissions with resource_id =
winningSubmitter.getId() AND submission type = "Contest Submission" AND submission
status = "Active" (submissions)
            Throw PhaseHandlingException if number of found submissions != 1
            Get winning submission ID:
                winningSubmissionId = submissions[0].getId()

            Use "Search all reviews for a phase based on resource roles" routine
to find reviews for ("Reviewer", "Accuracy Reviewer", "Failure Reviewer", "Stress
Reviewer") roles, submission_id = winningSubmissionId and phase_id =
reviewPhase.getId(). (reviews)
            For each review from reviews
                finalWorksheet.setScorecard(review.getScorecard())
                finalWorksheet.setSubmission(review.getSubmission())
                Copy comments and review items with types ("Comment", "Required",
"Recommended", "Appeal", "Appeal Response", "Aggregation Comment", "Aggregation
Review Comment", "Submitter Comment", "Manager Comment") from review to
finalWorksheet
            End For
        End If

        Use ReviewManager#createReview() to persist finalWorksheet
    Else
        finalWorksheet.setAuthor(resources[0].getId())
        finalWorksheet.setCommitted(false)
        Clone finalWorksheet
        Use ReviewManager#createReview() to persist the cloned finalWorksheet
    End If
```

Delete the previous final fix

```
    Use similar logic as in "Check if the final fix has been uploaded" to get the
finalFix
    Set the finalFix upload status to "Deleted" use finalFix.setUploadStatus()
    Update the finalFix using UploadManager.updateUpload()
```

Call sendEmail(phase) method of AbstractPhaseHandler

perform() – Stop

Call sendEmail(phase) method of AbstractPhaseHandler

1.3.37 Final Review Phase Handler

canPerform() - Can start

Final Review can start as soon as the dependencies are met

```
    Use the routine "Check if all dependencies of a phase have correct status".
```

If phase.calcStartDate() is not null
    Use the routine "Check if a phase start time is reached".

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Phase start time is not yet reached" (created in subroutine)

canPerform() - Can stop

Final Review can stop when:
The dependencies are met;
The final review is committed by the final reviewer.

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Final review scorecard is not yet committed"

The dependencies are met:
    Use the routine "Check if all dependencies of a phase have correct status".

Check if final review is committed by the final reviewer
    Use the "Get worksheet" routine with "Final Reviewer" role to get the final review worksheet. (finalReview)
    Check committed value
        Using finalReview.isCommitted()

perform() – Start

Call sendEmail(phase,map) method of AbstractPhaseHandler
- supply the map with the following:
- N_FINAL_REVIEWERS: the number of final reviewers assigned

To get N_FINAL_REVIEWERS:
- "Search resource based on resource role name and phase id" with ResourceRoleName: "Final Reviewer" and phaseId:(phase.getId())

perform() – Stop

When Final Review phase is stopping, if the final review is rejected, another final fix/review cycle is inserted.

Find out if the final review is rejected
    Use the "Get worksheet" routine with "Final Reviewer" role to get the final review worksheet. (finalReview)
    Get all comments
        finalReview.getAllComments() (comments[])
    Search the result array and find the comment with
        comment.getCommentType().getName()="Final Review Comment" (frComment)
    Check if the comment is approve or reject
        If frComment.getExtraInfo()="Rejected" then the final review is rejected.
        Else if it's not "Approved" or "Accepted", throw PhaseHandlingException

Insert another final fix/review cycle
    Use the routine "Insert final fix/review cycle"

    Search for old "Final Reviewer" resource (associated with the current phase). Clone it and associate a cloned resource with the new "Final Review" phase.

    Persist cloned resource with ResourceManager#updateResource().

Else if approval phase doesn't already exist and "Approval Required" property of the project is "true", then insert approval phase if final review is approved

    Use the routine "Insert Approval Phase"

Call sendEmail(phase,map) method of AbstractPhaseHandler

- supply the map with the following:
- RESULT: The result of final fix: rejected/approved

To get RESULT: See **"Find out if the final review is rejected"** step above.

1.3.38 Approval Phase Handler

Change in 1.4: approval payment is now not set ("N/A") instead of setting it to 0.

canPerform() - Can start

```
Approval can start as soon as:
    The dependencies are met:
        Use the routine "Check if all dependencies of a phase have correct
status".

    If phase.calcStartDate() is not null
        Use the routine "Check if a phase start time is reached".

    There are enough approvers:
        Use ResourceManager to find resources with resource_role = "Approver" and
project_id = phase.getProject().getId(). Set the number of found resources to
projectApproversCount.
        Get "Reviewer Number" attribute from the phase (approverNum)
        The condition is met when projectApproversCount >= approverNum
```

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Phase start time is not yet reached" (created in subroutine)
"There are not enough approvers assigned for the project"

canPerform() - Can stop

```
Approval can end when:
       The dependencies are met;
       AND The approval scorecards are committed:
              At least the required number of Approver resources has been filled
              in a scorecard;
              OR The time allotted for this approval phase has expired (see
              PhasesHelper.reachedPhaseEndTime() routine);
```

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Phase end time is not yet reached and not enough approval scorecards are committed"

```
The dependencies are met:
    Use the routine "Check if all dependencies of a phase have correct status".

Check approval scorecards are committed
    Use ReviewManager to find all reviews with resource role = "Approver" and
project_id = phase.getProject().getId(). (approveReviews)
```

```
    Remove all reviews that are not associated with the current phase from
approveReviews. Also remove reviews that exceed "Reviewer Number" attribute of the
phase.
    If approveReviews is empty, return false
```
Check if the approval scorecards are committed
```
        Walk the approveReviews[] array and check committed using
approveReview.isCommitted().
```
Check enough approval scorecards are committed

If the committed count is equal to the number found in the phase's attribute "Reviewer Number" (use 1 if not specified), then all scorecards are committed. Otherwise condition fails due to some not being committed yet.

perform() – Start

Call sendEmail(phase,map) method of AbstractPhaseHandler
- supply the map with the following:
- N_APPROVERS: the number of available approvers
- N_REQUIRED_APPROVERS: the number of approvers required
- NEED_APPROVER: 1 if N_APPROVERS < N_REQUIRED_APPROVERS,0 otherwise.

To get N_APPROVERS:
- See "**There are enough approvers**" above

perform() – Stop

When Approval phase is stopping, if the approval is rejected, another final fix/review cycle is inserted.

Check if approval rejected
```
    Get approveReviews as described in "Check approval scorecards are committed"
above
    Iterate over review.getAllComments() of each review from approveReviews:
        If comment.getCommentType().getName()="Approval Review Comment"
            If comments.getExtraInfo()="Rejected", the approval is rejected
            Else if comments.getExtraInfo() is not "Approved" or "Accepted", throw
PhaseHandlingException
```

If approval rejected
```
    Use the routine "Insert final fix/review cycle"
    Search for old "Final Reviewer" resource (associated with the current phase).
Clone it and associate a cloned resource with the new "Final Review" phase.
    Persist cloned resource with ResourceManager#updateResource().
```
Else
```
    Iterate over review.getAllComments() of each review from approveReviews:
        If comment.getCommentType().getName()="Approval Review Comment - Other
Fixes" and comments.getExtraInfo()="Required" then other fixes are required
    If other fixes are required
        Get project with ProjectManager#getProject(phase.getProject().getId())
(project)
        project.setProperty("Requires Other Fixes", "true")
        Update project in persistence using ProjectManager#updateProject()
```

Call sendEmail(phase,map) method of AbstractPhaseHandler
- supply the map with the following:
- RESULT: the approval result: approved/rejected

To get RESULT: See "**Check if approval rejected**" above

1.3.39 Post-Mortem Phase Handler

canPerform() - Can start

```
*Post-mortem can start as soon as the dependencies are met*
    Use the routine "Check if all dependencies of a phase have correct status".

*If phase.calcStartDate() is not null*
    Use the routine "Check if a phase start time is reached".
```

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Phase start time is not yet reached" (created in subroutine)

canPerform() - Can stop

```
*Post-mortem can start when*
       *The dependencies are met;*
       *The period has passed.*
           Use the "Check if a phase end time is reached" routine.
       *The post-mortem scorecards are committed;*
       *At least the required number of Post-Mortem Reviewer resources have filled in a scorecard;*
```

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Phase end time is not yet reached"
"Not enough post-mortem scorecards are committed"

```
*Check the dependencies are met:*
    Use the routine "Check if all dependencies of a phase have correct status".

*Check post-mortem scorecards are committed:*
    Use ResourceManager to find resources with resource_role = "Post-Mortem Reviewer" and project_id = phase.getProject().getId(). (postMortemReviews)
    **Check if the post-mortem scorecards are committed**
        Walk the postMortemReviews[] array and check committed using postMortemReview.isCommitted()
    **Check enough post-mortem scorecards are committed**
       If the committed count is >= than the number found in the phase's attribute "Reviewer Number", then the condition is met. Otherwise condition fails due to some not being committed yet.
```

perform() – Start

```
Get all resource roles with ResourceManager.getAllResourceRoles() call (resourceRoles)
Find a role with name equal to "Post-Mortem Reviewer" in resourceRoles (postMortemReviewerRole)
Search for all existing post-mortem resources for the current project (existingPostMortemResources:Resource[])
Search for resources with "Submitter", "Reviewer", "Accuracy Reviewer", "Failure Reviewer", "Stress Reviewer" and "Copilot" role names for the current project (candidatePostMortemResources:Resource[])
For each resource in candidatePostMortemResources
    valid = false
    If resource role name for resource is "Submitter"
        If resource.getSubmissions() is not empty, valid = true
    Else
        valid = true
    End If

    If valid
        Get "External Reference ID" property from resource (candidateId)
        If existingPostMortemReviewer doesn't contain a resource with "External
Reference ID" property equal to candidateId
            Check if user has pending terms
                Get terms-specific services
                    ManagerHelper.getProjectTermsOfUse()
                    ManagerHelper.getUserTermsOfUse()
                Get necessary terms of use for resource's role using
projectRoleTermsOfUse.getTermsOfUse() (necessaryTerms)
                For each necessaryTerm from necessaryTerms
                    If userTermsOfUse.hasTermsOfUse() returns true for specific
termsId and userId, the user has pending terms
                End For
            If user doesn't have pending terms
                Create new Resource instance for postMortemReviewerRole. Copy to
it properties of resource: "External Reference ID", "Handle", "Email" and
"Rating".
                Set "Registration Date" property of this Resource instance to the
current timestamp.
                Set "Payment Status" property of this Resource instance to "N/A".
                Persist this new resource using resourceManager.updateResource()
method
            End If
        End If
    End If
End For
```

Call sendEmail(phase,map) method of AbstractPhaseHandler
- supply the map with the following:
- NEED_POST_MORTEM_REVIEWERS: if N_POST_MORTEM_REVIEWERS < N_REQUIRED_POST_MORTEM_REVIEWERS, set to 1, otherwise set to 0.
- N_POST_MORTEM_REVIEWERS: the number of post mortems reviewers available
- N_REQUIRED_POST_MORTEM_REVIEWERS: the number of required post mortems reviewers

To get N_POST_MORTEM_REVIEWERS:
- See "Check post-mortem scorecards are committed" above

perform() – Stop

Call sendEmail(phase,map) method of AbstractPhaseHandler

1.3.40 *Checkpoint Submission Phase Handler*

canPerform() - Can start

Checkpoint Submission phase can start as soon as:
- The dependencies are met
- Phase start date/time is reached (if specified)

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Phase start time is not yet reached" (created in subroutine)

Check the dependencies are met
    Use the routine "**Check if all dependencies of a phase have correct status**".

If phase.calcStartDate() is not null
    Use the routine "**Check if a phase start time is reached**".

PhaseHelper.canPhaseStart() method can be reused.

canPerform() - Can stop

Checkpoint Submission phase can stop when all of these conditions met:
- *The dependencies are met*
- *The period has passed*
- *If there are no checkpoint submissions OR:*

--- If manual checkpoint screening is required, the number of checkpoint submissions that have passed manual checkpoint screening meets the required number.

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Phase end time is not yet reached"
"Not enough checkpoint submissions for the project"

The dependencies are met
Use the routine **"Check if all dependencies of a phase have correct status"**.

The period has passed
Use the **"Check if a phase end time is reached"** routine.

Check if there are any submissions (see SubmissionPhaseHandler#hasAnySubmission() for reference):
Search for all active submissions with **"Checkpoint Submission"** type using **"Search all active submissions with specific submission type for the project"** routine.

If the number of found submissions == 0, return true.

Else if manual checkpoint screening is required, the number of checkpoint submissions that have passed manual checkpoint screening should meet the required number. (See below)

Get the required number of submission that passed checkpoint screening, if the attribute is null (missing), return true(it means we don't need to check the condition that the required number of submission that passed checkpoint screening).

```
    Get the "Submission Number" phase attribute
        String subNumber = phase.getAttribute("Submission Number")
    If (subNumber == null) return null.
```

The number of checkpoint submissions that have passed manual checkpoint screening meets the required number:
Search all checkpoint screening scorecards for the current phase:
Locate the next checkpoint screening phase using **"Locating phases"** routine.
Get the checkpointScreeningPhaseId, if checkpoint screening phase does not exist, simply return true.
Use the **"Search all reviews for a phase base on resource roles"** routine.
Input:
- phaseId=checkpointScreeningPhaseId
- resourceRoleNames=**"Checkpoint Screener"**

Output:
- checkpointScreenReviews[]

Get the checkpoint screening minimum score:
Use the **"Get the scorecard minimum score using a review"** routine:
(With the first instance in checkpointScreenReviews[])

```
    Count the number submissions that pass screening
        passedNum = 0
        For each checkpointScreenReviews
            If milestoneScreenReviews.getScore() >= minScore
                passedNum++
            End If
        End For
    Check if passedNum >= subNumber
```

perform() - Start

Call sendEmail(phase,map) method of AbstractPhaseHandler

perform() - Stop

Call sendEmail(phase,map) method of AbstractPhaseHandler
Supply the map with the following:

- N_SUBMITTERS : the number of checkpoint submitters
- SUBMITTER: a list containing 1 element for each submitter. Each submitter is represented as a map with the following keys: SUBMITTER_HANDLE, SUBMITTER_RELIABILITY, SUBMITTER_RATING

To get SUBMITTER array:
- Use **Deliverable Management** to search for submission with: ProjectId = phase.getProject().getId() AND submission type ID = looked up "**Checkpoint Submission**" type ID
- For each Submission returned, get the owner: submission.getUpload().getOwner()
- Use **Resource Management** to get submitter (submitter is a resource):

```
            submitter = ResourceManager.getResource(submitterId)
```

1.3.41 Checkpoint Screening Phase Handler

canPerform() - Can start

Checkpoint Screening phase can start as soon as:
- The dependencies are met
- Phase start date/time is reached (if specified)
- There are submissions with "**Checkpoint Submission**" type.

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:
"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Phase start time is not yet reached" (created in subroutine)
"There are no checkpoint submissions for the project"

Check the dependencies are met
    Use the routine "**Check if all dependencies of a phase have correct status**".

If phase.calcStartDate() is not null
    Use the routine "**Check if a phase start time is reached**".

There are submissions with "**Checkpoint Submission**" type.
    Use "**Search all active submissions with specific submission type for the project**" routine.

canPerform() - Can stop

Checkpoint Screening phase can stop when:
- The dependencies are met
- All checkpoint submissions have one checkpoint screening scorecard committed.

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Not all checkpoint screening scorecards are committed (see submission with ID=XXX)"

```
The dependencies are met
    Use the routine "Check if all dependencies of a phase have correct status".

All checkpoint submissions have one checkpoint screening scorecard committed.
    Search all checkpoint screening scorecards for the current phase:
    Use the "Search all reviews for a phase base on resource roles" routine.
        Input:
            - phaseId=phase.getId() (current phase id)
            - resourceRoleName="Checkpoint Screener"
        Output:
            - checkpointScreenReviews[]

    Search all active checkpoint submissions with "Checkpoint Submission" type for
the current project using "Search all active submissions with specific submission
type for the project" routine (submissions[])

    Number of reviews must be equal to the number of submissions.

    Check if every checkpoint submission has an associated screening scorecard
       For each checkpoint submission
           Look for its matching scorecard using
    submission.getId()=screenReview.getSubmission()
       End For

    Check if all of the screening scorecards are committed
       Walk the checkpointScreenReviews[] and check
checkpointScreenReview.isCommitted()
```

perform() - Start

Call sendEmail(phase,map) method of AbstractPhaseHandler
- supply the map with the following:
- NEED_CHECKPOINT_SCREENER : set to 1 if checkpoint screener is not available, 0 otherwise
- SUBMITTER: a list containing 1 element for each submitter. Each submitter is represented as a map with following key SUBMITTER_HANDLE, SUBMITTER_RELIABILITY, SUBMITTER_RATING

To get NEED_CHECKPOINT_SCREENER:
- Lookup resource role id for "Checkpoint Screener" using ResourceRoleLookupUtility (milestoneScreenerId)
- Use **Resource Management** to search for resource with the resource role id = "milestoneScreenerId"
- If this is null then NEED_CHECKPOINT_SCREENER is set to 1

To get SUBMITTER array:
- Use Deliverable Management to search for submission with: ProjectId = phase.getProject().getId() AND submission type = **"Checkpoint Submission"**
- For each Submission returned, get the owner: submission.getUpload().getOwner()
- Use **Resource Management** to get submitter (submitter is a resource):

  ```
  submitter = ResourceManager.getResource(submitterId)
  ```

perform() – Stop

When screening is stopping:
- All submissions with failed screening scorecard scores should be set to the status "Failed Checkpoint Screening"
- Screening score for the all submissions will be saved to the submitters' resource properties named "Checkpoint Screening Score".

```
    Search all submissions with "Checkpoint Submission" type for current project
using "Search all active submissions with specific submission type for the
project" routine (submissions[]).

    Search all screening scorecard for the current phase:
        Use the "Search all reviews for a phase base on resource roles" routine.
        Input:
            - phaseId=phase.getId() (current phase id)
            - resourceRoleNames="Checkpoint Screener"
        Output:
            - screenReviews[]

    Throw PhaseHandlingException if the number of submissions is not equal to the
number of screening scorecards.

    Get the screening minimum score:
        Use the "Get the scorecard minimum score using a review" routine.
        (With the first instance in screenReviews[])

    For each submission
        Find the matching screening review using submission.getId() =
screeningReview.getSubmission()
        Get submission's screening score: screeningScore =
screeningReview.getScore()

        Store the screening score for the submission
        Set screening score:
submission.setScreeningScore(Double.valueOf(String.valueOf(screeningScore)));

        If screeningScore < screening minimum score
            Set submission status to "Failed Checkpoint Screening", this
submission will no longer be active, which will be eliminated from being reviewed.
                Lookup submission status with name "Failed Checkpoint Screening"
using UploadManager (failedScreeningStatusId) (See PhaseHelper.
getSubmissionStatus() for reference)
                Set the status for the submission using
submission.setSubmissionStatus()
                Update submission using Deliverable Management:
UploadManager.updateSubmission()
        End If
    End for
```

Call sendEmail(phase,map) method of AbstractPhaseHandler

- supply the map with the following:
- SUBMITTER: a list containing 1 element for each submitter. Each submitter is represented as a map with following key SUBMITTER_HANDLE, SUBMITTER_SCORE, SUBMITTER_RESULT
- NO_SCREENING_PASS: 0 if no submissions passed screening, 1 otherwise

To get SUBMITTER array:
- Use **Deliverable Management** to search for submission with: ProjectId = phase.getProject().getId() AND submission type = "**Checkpoint Submission**"
- For each Submission returned, get the owner: submission.getUpload().getOwner()
- Use **Resource Management** to get submitter (submitter is a resource):
      submitter = ResourceManager.getResource(submitterId)
- To get the submitter score: get from submission.getScreeningScore()

1.3.42 *Checkpoint Review Phase Handler*

canPerform() - Can start

Review can start as soon as:
- The dependencies are met
- Phase start date/time is reached (if specified)
- There is at least one active submission with **"Checkpoint Submission".**

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Phase start time is not yet reached" (created in subroutine)
"There are no checkpoint submissions that passed screening for the project"

Check the dependencies are met
Use the routine **"Check if all dependencies of a phase have correct status".**

If phase.calcStartDate() is not null
Use the routine **"Check if a phase start time is reached".**

Check *there is at least one active submission with* **"Checkpoint Submission"** type.
Use **"Search all active submissions with specific submission type for the project"** routine

canPerform() - Can stop

- All dependencies are met
- Checkpoint reviewer is assigned for the project
- All active submissions with **"Checkpoint Submission"** type *have one review scorecard for the phase*

If all conditions are met this method must return OperationCheckResult.SUCCESS. Otherwise a new OperationCheckResult instance with one of the following explanation messages must be constructed and returned:

"Dependency XXX phase is not yet started/ended" (created in subroutine)
"Checkpoint reviewer is not assigned for the project"
"Not all checkpoint review scorecards are committed (see submission with ID=XXX)"

All active submissions have one review scorecard for the phase
Search all active submissions with **"Checkpoint Submission"** type for current project using **"Search all active submissions with specific submission type for the project"** routine (submissions[])

Search for checkpoint reviewer with resource_role_name = **"Checkpoint Reviewer"** AND phase id = phase.getId() using **"Search resource based on resource role names and phase id"** routine (reviewers[])

If numbers of reviewers == 0 return false;
If numbers of reviewers != 1 throw PhaseHandlingException.

Search all review scorecards for the current phase with resource_role_name = **"Checkpoint Reviewer"** AND phase id = phase.getId() using **"Search all reviews for a phase based on resource roles"** routine (reviews[])

If submissions.length != reviews.length, throw PhaseHandlingException.

For each submission
Match the submission with its review by using submission.getId() and review.getSubmission()
Each submission should have one review scorecard from checkpoint reviewer (reviewers[0]) and the review must be committed.

```
    End For
```

perform() – Start

```
Call sendEmail(phase,map) method of AbstractPhaseHandler
-  supply the map with the following:
-  NEED_CHECKPOINT_REVIEWER : set to 1 if checkpoint reviewer is not available, 0
   otherwise
-  SUBMITTER: a list containing 1 element for each submitter. Each submitter is
   represented as a map with following key SUBMITTER_HANDLE,
   SUBMITTER_RELIABILITY, SUBMITTER_RATING

To get NEED_CHECKPOINT_REVIEWER:
-  Lookup resource role id for "Milestone REVIEWER" using
ResourceRoleLookupUtility (milestoneReviewerId)
-  Use Resource Management to search for resource with the resource role id =
"milestoneReviewerId"
-  If this is null then NEED_CHECKPOINT_REVIEWER is set to 1

To get SUBMITTER array:
-  Use Deliverable Management to search for submission with: ProjectId =
phase.getProject().getId() AND submission type = "Checkpoint Submission"
-  For each Submission returned, get the owner: submission.getUpload().getOwner()
-  Use Resource Management to get submitter (submitter is a resource):
            submitter = ResourceManager.getResource(submitterId)
```

perform() – Stop

Initial score for the all passed screening submissions will be calculated and saved to the submitters’ resource properties named "Initial Score". Submissions that passed screening will have status "Active" (instead of "Failed Checkpoint Screening")

```
    Search all active submissions with "Checkpoint Submission" type for the
current project using "Search all active submissions with specific submission type
for the project" routine (submissions[])

    Search the reviewers with resource_role_name = "Checkpoint Reviewer" AND phase
id = phase.getId() using "Search resource based on resource role names and phase
id" routine (reviewers[])

    Search all review scorecards for the current phase with resource_role_name =
"Checkpoint Reviewer" AND phase id = phase.getId() using "Search all reviews for a
phase based on resource roles" routine (reviews[])

    submissionScores = new
com.topcoder.management.review.scoreaggregator.Submission[submissions.length]

    For each submission
        Match the submission with its reviews by using submission.getId() and
review.getSubmission()
        Each submission should have exactly one review scorecard.
        Get the score from the review scorecard using review.getScore()
        Set initial score to the submission using Submission#setInitialScore()
        Set final score to the submission using Submission#setFinalScore()
        Put(submission ID, score) pair to submissionScores array
    End For

    Aggregate scores using ManagerHelper#getScorecardAggregator():
    scoreAggregator.aggregateScores(submissionScores) (aggregations)

    Calculate placements:
```

```
    scoreAggregator.calcPlacements(aggregations) (placements)

    Get minimum passing score using ManagerHelper#getScorecardManager() (minScore)
    Get "Failed Checkpoint Review" submission status (failedStatus)

    For each submission from submissions:
        Find the corresponding rankedSubmission from placements where
rankedSubmission.getId() == submission.getId()
        Set placement to the submission:
            submission.setPlacement(rankedSubmission.getRank() * 1L)
        If rankedSubmission.getAggregatedScore() < minScore
            Set status of the submission to "Failed Checkpoint Review":
                submission.setSubmissionStatus(failedStatus)
    End For

    Get all project prizes: project.getPrizes() (prizes)
    Remove from prizes all prizes for which prize.getPrizeType().getDescription()
is not equal to "Checkpoint Prize".
    Sort prizes by placement.
    Sort submissions by placement, and if equal - by creation timestamps.

    currentPrize = prizes.get(0);
    currentPrizeIndex = 0;
    currentPrizeNumLeft = currentPrize.getNumberOfSubmissions();
    For i = 0 .. submissions.length - 1:
        If placement of submissions[i] == currentPrize.getPlace()
            If currentPrizeNumLeft > 0
                Decrease currentPrizeNumLeft by 1.
            Else continue;
        Else
            Increase currentPrizeIndex by 1.
            If currentPrizeIndex >= prizes.size()
                break;
            Else
                currentPrize = prizes.get(currentPrizeIndex);
                currentPrizeNumLeft = currentPrize.getNumberOfSubmissions();
        Set prize to the submission: submission.setPrize(currentPrize);
    End For

    For each submission from submissions:
        Update submission in persistence:
            ManagerHelper#getUploadManager().updateSubmission(submission)
    End For
```

```
Call sendEmail(phase,map) method of AbstractPhaseHandler
-  supply the map with the following:
-  SUBMITTER: a list containing 1 element for each submitter. Each submitter is
   represented as a map with following key SUBMITTER_HANDLE, SUBMITTER_SCORE
```

To get SUBMITTER array:
  Use **Deliverable Management** to search for submission with: projectId = phase.getProject().getId()

```
  For each Submission returned, get the owner: submission.getUpload().getOwner()
      Use Resource Management to get submitter (submitter is a resource):
            submitter = ResourceManager.getResource(submitterId)
      To get the SUBMITTER_SCORE use submission.getInitialScore();
```

1.3.43 Logging

Logging using the Logging Wrapper has been added to the following handlers:

- CheckpointSubmissionPhaseHandler
- CheckpointScreeningPhaseHandler

- CheckpointReviewPhaseHandler
- SpecificationSubmissionPhaseHandler
- SpecificationReviewPhaseHandler
- RegistrationPhaseHandler
- ReviewPhaseHandler
- AppealsPhaseHandler
- AggregationPhaseHandler
- AggregationReviewPhaseHandler
- PostMortemPhaseHandler

There is no coherent strategy for logging found across those four classes, confirming that they have been added ad hoc. As such, it is necessary to outline logging in each case, as is done below. The classes do use the helper class LogMessage.

1.3.44 ReviewPhaseHandler

canPerform - stop: Logs details whether the dependencies are done, whether reviews are finished, and all test cases uploaded, all at INFO level. The methods to check the last two conditions perform additional DEBUG and INFO logging detailing their processing.

1.3.45 AppealsPhaseHandler

canPerform – stop: Logs just the error if something goes wrong during the check whether appeals phase can be stopped.

1.3.46 AggregationPhaseHandler

canPerform - stop: Logs at WARN level that this stage cannot begin because there are no winners or there is no aggregator.

Uses ERROR level logging for error conditions in the perform method and when there are no winners or there is no aggregator.

Logs at INFO level when creating the aggregation worksheet.

1.3.47 AggregationReviewPhaseHandler

Logs at DEBUG level when entering or exiting canPerform method, when at the code point of checking if the phase is starting or ending, and when automatically approving aggregation reviews. Logs at INFO when entering process of approving the aggregation review

Logs at WARN when it cannot approve reviews automatically.

Logs in the following manner when checking of aggregation review is done:
- INFO if there is no review or aggregation phase in the project
- DEBUG if there is not aggregator assigned
- INFO if the aggregator or submitter has not commented

1.3.48 SpecificationSubmissionPhaseHandler and SpecificationReviewPhaseHandler

These classes just need to log all thrown exceptions at ERROR level. Logging in the new handlers must be performed consistently with the existing handlers.

1.3.49 RegistrationPhaseHandler

canPerform – start: Logs just the error if something goes wrong when checking whether all parent projects are completed.

1.3.50 PostMortemPhaseHandler

perform – start: Logs the warning when user has pending terms, and thus cannot be assigned as a Post-Mortem Reviewer.

1.4 Component Class Overview

ManagerHelper
This is the helper class to load manager instances from a default configuration namespace.
Changes in 1.6.1:
- Removed screeningManager together with its getter.
- Removed private initScreeningManager() method.

AbstractPhaseHandler
This abstract class is used as a base class for all phase handlers. This class contains logic in the constructor to load configuration settings for a phase handler. Settings include database connection, email template and email related information.

AggregationPhaseHandler
This class implements PhaseHandler interface to provide methods to check if a phase can be executed and to add extra logic to execute a phase. It will be used by Phase Management component. It is configurable using an input namespace. The configurable parameters include database connection and email sending. This class handles the aggregation phase. If the input is of other phase types, PhaseNotSupportedException will be thrown.
Please see details about this handler in the section 1.3.34.
Changes in 1.6.1:
- canPerform() method was updated to return not only true/false value, but additionally an explanation message in case if operation cannot be performed

AggregationReviewPhaseHandler
This class implements PhaseHandler interface to provide methods to check if a phase can be executed and to add extra logic to execute a phase. It will be used by Phase Management component. It is configurable using an input namespace. The configurable parameters include database connection and email sending. This class handles the aggregation review phase. If the input is of other phase types, PhaseNotSupportedException will be thrown.
Please see details about this handler in the section 1.3.35.
Changes in 1.6.1:
- canPerform() method was updated to return not only true/false value, but additionally an explanation message in case if operation cannot be performed

AppealsPhaseHandler
This class implements PhaseHandler interface to provide methods to check if a phase can be executed and to add extra logic to execute a phase. It will be used by Phase Management component. It is configurable using an input namespace. The configurable parameters include database connection and email sending. This class handles the appeals phase. If the input is of other phase types, PhaseNotSupportedException will be thrown.
Please see details about this handler in the section 1.3.32.
Changes in 1.6.1:
- canPerform() method was updated to return not only true/false value, but additionally an explanation message in case if operation cannot be performed

AppealsResponsePhaseHandler
This class implements PhaseHandler interface to provide methods to check if a phase can be executed and to add extra logic to execute a phase. It will be used by Phase Management component. It is configurable using an input namespace. The configurable parameters include

database connection and email sending. This class handles the appeals response phase. If the input is of other phase types, PhaseNotSupportedException will be thrown.
Please see details about this handler in the section 1.3.33.
Changes in 1.6.1:
- canPerform() method was updated to return not only true/false value, but additionally an explanation message in case if operation cannot be performed

ApprovalPhaseHandler
This class implements PhaseHandler interface to provide methods to check if a phase can be executed and to add extra logic to execute a phase. It will be used by Phase Management component. It is configurable using an input namespace. The configurable parameters include database connection and email sending. This class handles the approval phase. If the input is of other phase types, PhaseNotSupportedException will be thrown.
Please see details about this handler in the section 1.3.38.
Changes in 1.4: Updated not to use ContestDependencyAutomation. Approval phase ends when at the scheduled time even when there are no submitted approval scorecards.
Changes in 1.6.1:
- canPerform() method was updated to return not only true/false value, but additionally an explanation message in case if operation cannot be performed

FinalFixPhaseHandler
This class implements PhaseHandler interface to provide methods to check if a phase can be executed and to add extra logic to execute a phase. It will be used by Phase Management component. It is configurable using an input namespace. The configurable parameters include database connection and email sending. This class handles the aggregation phase. If the input is of other phase types, PhaseNotSupportedException will be thrown.
Please see details about this handler in the section 1.3.36.
Changes in 1.6.1:
- canPerform() method was updated to return not only true/false value, but additionally an explanation message in case if operation cannot be performed

FinalReviewPhaseHandler
This class implements PhaseHandler interface to provide methods to check if a phase can be executed and to add extra logic to execute a phase. It will be used by Phase Management component. It is configurable using an input namespace. The configurable parameters include database connection, email sending. This class handles the final review phase. If the input is of other phase types, PhaseNotSupportedException will be thrown.
Please see details about this handler in the section 1.3.37.
Change in 1.4: Updated not to use ContestDependencyAutomation.
Changes in 1.6.1:
- canPerform() method was updated to return not only true/false value, but additionally an explanation message in case if operation cannot be performed

PostMortemPhaseHandler
This class implements PhaseHandler interface to provide methods to check if a phase can be executed and to add extra logic to execute a phase. It will be used by Phase Management component. It is configurable using an input namespace. The configurable parameters include database connection, email sending and the required number of registration. This class handles the registration phase. If the input is of other phase types, PhaseNotSupportedException will be thrown.
Please see details about this handler in the section 1.3.39.
Changes in 1.6.1:
- canPerform() method was updated to return not only true/false value, but additionally an explanation message in case if operation cannot be performed

SpecificationSubmissionPhaseHandler

This class implements PhaseHandler interface to provide methods to check if a phase can be executed and to add extra logic to execute a phase. It will be used by Phase Management component. It is configurable using an input namespace. The configurable parameters include database connection and email sending parameters. This class handles the specification submission phase. If the input is of other phase types, PhaseNotSupportedException will be thrown.
Please see details about this handler in the section 1.3.26.
Changes in 1.6.1:
- canPerform() method was updated to return not only true/false value, but additionally an explanation message in case if operation cannot be performed

SpecificationReviewPhaseHandler
This class implements PhaseHandler interface to provide methods to check if a phase can be executed and to add extra logic to execute a phase. It will be used by Phase Management component. It is configurable using an input namespace. The configurable parameters include database connection and email sending parameters. This class handles the specification review phase. If the input is of other phase types, PhaseNotSupportedException will be thrown.
Please see details about this handler in the section 1.3.27.
Changes in 1.6.1:
- canPerform() method was updated to return not only true/false value, but additionally an explanation message in case if operation cannot be performed

RegistrationPhaseHandler
This class implements PhaseHandler interface to provide methods to check if a phase can be executed and to add extra logic to execute a phase. It will be used by Phase Management component. It is configurable using an input namespace. The configurable parameters include database connection, email sending and the required number of registration. This class handles the registration phase. If the input is of other phase types, PhaseNotSupportedException will be thrown.
Please see details about this handler in the section 1.3.28.
Change in 1.4: Dependency projects are checked and project start delayed if required only if this phase is the first phase in the project.
Changes in 1.6.1:
- canPerform() method was updated to return not only true/false value, but additionally an explanation message in case if operation cannot be performed

ReviewPhaseHandler
This class implements PhaseHandler interface to provide methods to check if a phase can be executed and to add extra logic to execute a phase. It will be used by Phase Management component. It is configurable using an input namespace. The configurable parameters include database connection and email sending. This class handles the review phase. If the input is of other phase types, PhaseNotSupportedException will be thrown.
Please see details about this handler in the section 1.3.31.
Changes in 1.6.1:
- canPerform() method was updated to return not only true/false value, but additionally an explanation message in case if operation cannot be performed

ScreeningPhaseHandler
This class implements PhaseHandler interface to provide methods to check if a phase can be executed and to add extra logic to execute a phase. It will be used by Phase Management component. It is configurable using an input namespace. The configurable parameters include database connection, email sending. This class handles the screening phase. If the input is of other phase types, PhaseNotSupportedException will be thrown.
Please see details about this handler in the section 1.3.30.
Changes in 1.6.1:
- canPerform() method was updated to return not only true/false value, but additionally an explanation message in case if operation cannot be performed

SubmissionPhaseHandler
This class implements PhaseHandler interface to provide methods to check if a phase can be executed and to add extra logic to execute a phase. It will be used by Phase Management component. It is configurable using an input namespace. The configurable parameters include database connection, email sending and the required number of submissions that pass screening. This class handles the submission phase. If the input is of other phase types, PhaseNotSupportedException will be thrown.
Please see details about this handler in the section 1.3.29.
Changes in 1.6.1:
- canPerform() method was updated to return not only true/false value, but additionally an explanation message in case if operation cannot be performed
- Removed private methods arePassedSubmissionsEnough(), getManualScreeningPasses() and getAutoScreeningPasses().

CheckpointReviewPhaseHandler
This class implements PhaseHandler interface to provide methods to check if a phase can be executed and to add extra logic to execute a phase. It will be used by Phase Management component. It is configurable using an input namespace. The configurable parameters include database connection and email sending. This class handles the checkpoint review phase. If the input is of other phase types, PhaseNotSupportedException will be thrown.
Please see details about this handler in the section 1.3.39.
Changes in 1.6.1:
- canPerform() method was updated to return not only true/false value, but additionally an explanation message in case if operation cannot be performed

CheckpointScreeningPhaseHandler
This class implements PhaseHandler interface to provide methods to check if a phase can be executed and to add extra logic to execute a phase. It will be used by Phase Management component. It is configurable using an input namespace. The configurable parameters include database connection, email sending. This class handles the checkpoint screening phase. If the input is of other phase types, PhaseNotSupportedException will be thrown.
Please see details about this handler in the section 1.3.38.
Changes in 1.6.1:
- canPerform() method was updated to return not only true/false value, but additionally an explanation message in case if operation cannot be performed

CheckpointSubmissionPhaseHandler
This class implements PhaseHandler interface to provide methods to check if a phase can be executed and to add extra logic to execute a phase. It will be used by Phase Management component. It is configurable using an input namespace. The configurable parameters include database connection, email sending and the required number of submissions that pass screening. This class handles the checkpoint submission phase. If the input is of other phase types, PhaseNotSupportedException will be thrown.
Please see details about this handler in the section 1.3.37.
Changes in 1.6.1:
- canPerform() method was updated to return not only true/false value, but additionally an explanation message in case if operation cannot be performed

PhaseStatusLookupUtility
This class provides the function to get lookup id from a lookup name of "phase_status_lu" table. Since lookup id/value pairs do not change in the database per installation, this class caches the id/value pairs to minimize queries to the database. This class is used in various methods of PhaseHandler implementations.

PhaseTypeLookupUtility

This class provides the function to get lookup id from a lookup name of "phase_type_lu" table. Since lookup id/value pairs do not change in the database per installation, this class caches the id/value pairs to minimize queries to the database. This class is used in various methods of PhaseHandler implementations.

ProjectInfoTypeLookupUtility
This class provides the function to get lookup id from a lookup name of "project_info_type_lu" table. Since lookup id/value pairs do not change in the database per installation, this class caches the id/value pairs to minimize queries to the database. This class is used in various methods of PhaseHandler implementations.

ResourceRoleLookupUtility
This class provides the function to get lookup id from a lookup name of "resource_role_lu" table. Since lookup id/value pairs do not change in the database per installation, this class caches the id/value pairs to minimize queries to the database. This class is used in various methods of PhaseHandler implementations.

SubmissionStatusLookupUtility
This class provides the function to get lookup id from a lookup name of "submission_status_lu" table. Since lookup id/value pairs do not change in the database per installation, this class caches the id/value pairs to minimize queries to the database. This class is used in various methods of PhaseHandler implementations.

SubmissionTypeLookupUtility
This class provides the function to get lookup id from a lookup name of "submission_type_lu" table. Since lookup id/value pairs do not change in the database per installation, this class caches the id/value pairs to minimize queries to the database. This class is used in various methods of PhaseHandler implementations.

UploadStatusLookupUtility
This class provides the function to get lookup id from a lookup name of " upload_status_lu" table. Since lookup id/value pairs do not change in the database per installation, this class caches the id/value pairs to minimize queries to the database. This class is used in various methods of PhaseHandler implementations.

UploadTypeLookupUtility
This class provides the function to get lookup id from a lookup name of " upload_type_lu" table. Since lookup id/value pairs do not change in the database per installation, this class caches the id/value pairs to minimize queries to the database. This class is used in various methods of PhaseHandler implementations.

NotificationTypeLookupUtility
This class provides the function to get lookup id from a lookup name of " notification_type_lu" table. Since lookup id/value pairs do not change in the database per installation, this class caches the id/value pairs to minimize queries to the database. This class is used in various methods of PhaseHandler implementations.

EmailOptions
This class represents email options. It is a container for the set of options related to email to be send, such as subject, from address, template name and whether the email is to be sent or not.

LogMessage
This class encapsulates the entry log data and generates consistent log messages.
Changes in 1.4: Made just some trivial fixes to meet TopCoder standards.

PhasesHelper

The class having helper methods to perform argument validation and other phase related methods used by the PhaseHandler implementations.
Changes in 1.4:

- Some methods that perform common operations used by existing and new handlers were added: isFirstPhase() and insertSpecSubmissionSpecReview().
- getScreeningTasks() and searchActiveSubmissions() methods were updated to support submission types.

Changes in 1.6:

- Removed getScreeningTasks method.
- Updated arePhaseDependenciesMet() and canPhaseStart() methods to return OperationCheckResult instead of simply boolean value.

1.5 Component Exception Definitions

ConfigurationException
Represents an exception related to loading configuration settings. Inner exception should be provided to give more details about the error. It is used in PhaseHandler implementation classes.
Changes in 1.6.1:
- Extends BaseCriticalException instead of BaseException
- Added new constructors to meet TopCoder standards

PhaseNotSupportedException
This exception is thrown by a phase handler when the input phase is not the type the handler can handle. It is used in phase handler classes.
Changes in 1.6.1:
- Added new constructors to meet TopCoder standards

1.6 Thread Safety

This design is thread safe. Methods in lookup classes are marked as synchronized for thread safety access to the underlying map. Phase handler classes in this component are immutable so they are thread-safe by default. The manager components used in this component are not thread-safe. But since they are used in this component in a single thread manner in a helper class and phase handlers, this should not be a problem.
Thread safety of this component was not changed in the version 1.6.1.

2. Environment Requirements

2.1 Environment

Development language: Java 1.5
Compile target: Java 1.5, Java 1.6
QA Environment: RedHat Linux 4, Windows 2000, Windows 2003

2.2 TopCoder Software Components

- Configuration Manager 2.2 – used to read the configuration information.
- DB Connection Factory 1.1 – used to create the connection to the database.
- Base Exception 2.0 – used as the base for all custom exception classes.
- Email Engine 3.2 – used to send email
- Document Generator 3.1.1 - used to generate email template.
- Logging Wrapper 1.2 – used to log in selective classes
- Phase Management 1.1 – used to search for phases
- Review Management 1.0 – used to search for reviews
- Resource Management 1.1.1 – used to search for resources.
- Project Management 1.2 – used to search for projects.

- Scorecard Management 1.0.1 – used to search for scorecards.
- Deliverable Management 1.2 – used to search for uploaded deliverables
- Review Score Aggregator 1.0 – Used to aggregate review scores
- Search Builder 1.3.1 – used to perform filter-based searches
- ID Generator 3.0 – used to generate unique IDs
- Data Validation 1.1.1 – used to validate string and long values
- Scorecard Data Structure 1.0 – defines scorecard entities
- Review Data Structure 1.0 – defines review entities
- Project Phases 2.0 – defines project phase entities
- User Project Data Store 1.0.1 – defines ExternalUser and UserRetrieval

2.3 Third Party Components

- None

3. Installation and Configuration

3.1 Package Name

com.cronos.onlinereview.phases
com.cronos.onlinereview.phases.lookup
com.cronos.onlinereview.phases.logging

3.2 Configuration Parameters

For a phase handler class:
(Note that $schemeName is place holder for actual scheme for role names)
(Similarly, xx is to be replaced with either start or end)

Parameter	Description	More information
ConnectionFactoryNS	The namespace that contains settings for DB Connection Factory.	String – Required
ConnectionName	The name of the connection that will be used by DBConnectionFactory to create connection. If missing, default connection will be created.	String - Optional
ManagerHelperNamespace	The namespace that will be used by ManagerHelper class. If missing, default namespace of this class will be used.	String - Optional
Schemes	Property, contains all the supported schemes to send out the email	List of Child property, whose name attribute are the desired scheme name. For example `<Property name="Schemes">` `<Property name="DefaultScheme">` `<Value>Manager</Value>` `<Value>Reviewer</Value>` `<Value>Observer</Value>` `</Property>` …

		`</Property>`
Schemes/[schemeName]	Property, contains all the resource role names for the given scheme.	List of String values, which are the resource role names. "*" can be used to mach every role name in persistence.
[$schemeName]/xxPhaseEmail / EmailTemplateName	Contains the template name settings for start or end phase email for the particular roles.	String – Required
[$schemeName]/xxPhaseEmail / EmailSubject	Settings for the email subject for start or end phase email for the particular roles.	String – Optional
[$schemeName]/xxPhaseEmail / EmailFromAddress	Settings for the from email address for start or end phase email for the particular roles.	String – Required
[$schemeName]/xxPhaseEmail/SendEmail	If it is specified, email will be send for the particular role. If missing, email will not be sent for the particular role.	String - Optional
[$schemeName]/xxPhaseEmail/Priority	The priority for email options for this roles. If not set, default is 0.	Integer – Optional

For a ManagerHelper class:

Parameter	**Description**	**More information**
ProjectManager.ClassName	The full class name of the ProjectManager implementation.	String – Required
ProjectManager.Namespace	The configuration namespace to initialize the ProjectManager instance. If missing the default constructor is used to create the instance.	String – Optional
ProjectLinkManager.ClassName	The full class name of the ProjectLinkManager implementation.	String – Required
ProjectLinkManager.Namespace	The configuration namespace to initialize the ProjectLinkManager instance. If missing the default constructor is used to create the instance.	String – Optional
ScorecardManager.ClassName	The full class name of the ScorecardManager implementation.	String – Required
ScorecardManager.Namespace	The configuration namespace to initialize the ScorecardManager instance. If missing the default constructor is used to create the instance.	String – Optional
ReviewManager.ClassName	The full class name of the ReviewManager implementation.	String – Required
ReviewManager.Namespace	The configuration namespace to initialize the ReviewManager instance. If missing the default	String – Optional

	constructor is used to create the instance.	
UploadManager.ClassName	The full class name of the UploadManager implementation.	String – Required
UploadManager. UploadSearchBundleName	The SearchBundle name used to search for uploads.	String – Required
UploadManager. SubmissionSearchBundleName	The SearchBundle name used to search for submissions.	String – Required
UploadManager. UploadIdGeneratorName	The name to load IDGenerator for uploads.	String – Required
UploadManager. UploadTypeIdGeneratorName	The name to load IDGenerator for upload types.	String – Required
UploadManager. UploadStatusIdGeneratorName	The name to load IDGenerator for upload statuses.	String – Required
UploadManager. SubmissionIdGeneratorName	The name to load IDGenerator for submissions.	String – Required
UploadManager. SubmissionStatusIdGeneratorName	The name to load IDGenerator for submission statuses.	String – Required
UploadManager. PersistenceClassName	The name of the UploadPersistence implementation	String – Required
ResourceManager.ClassName	The full class name of the ResourceManager implementation.	String – Required
ResourceManager. ResourceSearchBundleName	The SearchBundle name used to search for resources.	String – Required
ResourceManager. ResourceRoleSearchBundleName	The SearchBundle name used to search for resource roles.	String – Required
ResourceManager. NotificationSearchBundleName	The SearchBundle name used to search for notifications.	String – Required
ResourceManager. NotificationTypeSearchBundleName	The SearchBundle name used to search for notification types.	String – Required
ResourceManager. ResourceIdGeneratorName	The name to load IDGenerator for resources.	String – Required
ResourceManager. NotificationTypeIdGeneratorName	The name to load IDGenerator for notifications.	String – Required
ResourceManager. ResourceRoleIdGeneratorName	The name to load IDGenerator for resource roles.	String – Required
ResourceManager. PersistenceClassName	The name of ResourcePersistence implementation	String – Required
PhaseManager.ClassName	The full class name of the PhaseManager implementation.	String – Required
PhaseManager.Namespace	The configuration namespace to initialize the PhaseManager instance. If missing the default constructor is used to create the instance.	String – Optional
ConnectionFactoryNS	Namespace for db connection factory	String – Required
ConnectionName	connection name.	String – Optional. If missing default connection is used.
SearchBundleManagerNS	namespace for search bundle manager	String – Required

UserProjectDataStore. UserRetrievalClassName	The full class name of the UserRetrieval implementation.	String – Required
UserProjectDataStore. UserRetrievalNamespace	The configuration namespace to initialize the UserRetrieval instance. If missing the default constructor is used to create the instance.	String – Optional
ScorecardAggregator. Namespace	The configuration namespace to initialize the ReviewScoreAggregator instance.	String – Required
ProjectRoleTermsOfUseClassName	The full class name of the ProjectRoleTermsOfUse implementation	String - Required
UserTermsOfUseClassName	The full class name of the UserTermsOfUse implementation	String - Required

3.3 Dependencies Configuration

The connection definitions in DB Connection Factory need to be configured. See the spec of the DB Connection Factory component for details.

4. Usage Notes

4.1 Required steps to test the component

- Extract the component distribution.
- Follow Dependencies Configuration.
- change the configuration in test_files/config/DB_Factory.xml and user_project_data_store.xml,

  please use test_files/sqls/all.sql to set up the table.(make sure you use test_files/sqls/drop.sql) to remove all the old table in your database.(the test_files/DevNullSmtp.jar may help with the email)
- Execute 'ant test' within the directory that the distribution was extracted to.
- The you can run test_files/sqls/drop.sql to drop the table.

4.2 Required steps to use the component

Load the configuration before using this component.

See sample configuration in Phase_handler.xml file for details.

4.3 Demo

This component is used by Phase Management component. The implementation of the interface PhaseManager will call phase handlers' methods.
In PhaseManager#canStart() method:
PhaseHandler#canPerform() method is called to check if a phase can start.
The passed phase instance status should be 'Scheduled'
In PhaseManager#canEnd() method:
PhaseHandler#canPerform() method is called to check if a phase can end.
The passed phase instance status should be 'Open'
In PhaseManager#start() method:
PhaseHandler#perform() method is called to provide additional starting logic.
The passed phase instance status should be 'Scheduled'
In PhaseManager#end() method:
PhaseHandler#perform() method is called to provide additional ending logic.
The passed phase instance status should be 'Open'

PhaseManager implementations also have to provide a mechanism let user to configure each phase type with the corresponding phase hander.

Below is the code which can be used for this demo.

```
//init the phase management component.
PhaseManager phaseManager = new
DefaultPhaseManager("com.topcoder.management.phase.DefaultPhaseManager");

//init the phase handler class.
PhaseHandler phaseHandler = new ScreeningPhaseHandler
(ScreeningPhaseHandler.DEFAULT_NAMESPACE);

// register a phase handler for dealing with canStart() and canEnd()
PhaseType phaseType = new PhaseType(3, "Screening");
phaseManager.registerHandler(phaseHandler, phaseType,
PhaseOperationEnum.START);
phaseManager.registerHandler(phaseHandler, phaseType,
PhaseOperationEnum.END);

//get the phase instance.
cleanupPhases();
Project project = setupPhases();
Phase[] phases = project.getAllPhases();
Phase phase = phases[2];

//prepare data

//canStart method will call canPerform() and perform() methods of the
phaseHandler.
OperationCheckResult checkResult = phaseManager.canStart(phase);

if (checkResult.isSuccess()) {
    phaseManager.start(phase, "1001");
} else {
    // print out a reason why phase cannot be started
    System.out.println(checkResult.getMessage());
    // Explanation can be one of the following for Screening phase:
    // "Dependency Submission phase is not yet ended"
    // "Phase start time is not yet reached"
    // "Contest has no submissions"
    return;
}

//prepare data

//canEnd method will call canPerform() and perform() methods of the
phaseHandler
checkResult = phaseManager.canEnd(phase);

if (checkResult.isSuccess()) {
    phaseManager.end(phase, "1001");
} else {
    // print out a reason why phase cannot be started
    System.out.println(checkResult.getMessage());
    // Explanation can be one of the following for Screening phase:
```

```
    // "Not all primary screening scorecards are committed
    //     (see submission with ID=123)"
    // "Individual screening scorecard is not committed"
    return;
}
```

For 1.2 version, there is no specific demo needed since the API is the same, just the internal of it is a bit different.

Here is a customer scenario describing the difference.
Consider the case where for the end of the screening phase, user babut (a manager) is configured to be sent out email with template A.

Also, babut is an observer of this project and configured to be sent our email with template_observer. (But since the observer priority is lower than Manager, so only one email will be sent in template A format).

Whereas, user prunthaban/ fastprogrammer (submitters) are configured to be sent out email with template B.

```
// let assume phase is screening phase
phaseManager.end(phase, "1001");
```

The user babut is sent out a mail with template A.
The user prunthaban/ fastprogrammer are sent out a mail with template B.

This is the email received by user babut:

```
Wed, Nov 18, 2009 12:50 PM EST
Hello The babut guy,
Handle: babut
This is the notification on project: Online Review Phases
OR link:
http://software.topcoder.com/review/actions/ViewProjectDetails.do?method=v
iewProjectDetails&pid=30007860
Version: 1.1
This is the end of the Screening phase.

Screening results:
Handle          Score          Result

prunthaban      90.0           Pass Screening
fastprogrammer  70.0           Failed Screening
```

This is the email received by prunthaban(fastprogrammer's is similar):

```
Wed, Nov 18, 2009 12:50 PM EST
Hello Allen Ivern,
Handle: prunthaban
This is the notification on project: Online Review Phases
OR link:
http://software.topcoder.com/review/actions/ViewProjectDetails.do?method=v
iewProjectDetails&pid=30007860
Version: 1.1
This is the end of the Screening phase.
```

For 1.4 version, two new handlers added, SpecificationSubmissionPhaseHandler and SpecificationReviewPhaseHandler.

Sample usage of SpecificationReviewPhaseHandler:

```
SpecificationReviewPhaseHandler handler = new
SpecificationReviewPhaseHandler(PHASE_HANDLER_NAMESPACE);

Project project = setupPhasesForSpec(true);
Phase[] phases = project.getAllPhases();
phases[0].setPhaseStatus(PhaseStatus.CLOSED);

Phase phase = phases[1];
phase.setPhaseStatus(PhaseStatus.SCHEDULED);

// insert a submission with specification submission type
Connection conn = getConnection();
Resource resource = createResource(4, 101L, 1, 17);
insertResources(conn, new Resource[] {resource});
Upload upload = createUpload(1, project.getId(), 4, 1, 1, "parameter");
insertUploads(conn, new Upload[] {upload});

Submission submission = createSubmission(1, upload.getId(), 1, 2);
insertSubmissions(conn, new Submission[] {submission});
// it now can perform start
If (handler.canPerform(phase).isSuccess()) {
    // start
    handler.perform(phase, "1001");
}
// we will close the phase now
// create a reviewer
Resource reviewer = createResource(5, 102L, 1, 15);
insertResources(conn, new Resource[] {reviewer});
insertResourceInfo(conn, reviewer.getId(), 1, "3");
Scorecard scorecard = createScorecard(1, 1, 2, 6, "name", "1.0", 75.0f,
100.0f);
 Review review = createReview(11, 5, 1, 1, true, 90.0f);
//  add a rejected comment
review.addComment(createComment(1111, reviewer.getId(), "Approved", 14,
"Specification Review Comment"));
insertScorecards(conn, new Scorecard[] {scorecard});
insertReviews(conn, new Review[] {review});
insertCommentsWithExtraInfo(conn, new long[] {1}, new long[]
{reviewer.getId()}, new long[] {review.getId()}, new String[] {"Approved
Comment"}, new long[] {14}, new String[] {"Approved"});
// change the phase status to open
phase.setPhaseStatus(PhaseStatus.OPEN);
// we can stop the phase now
If (handler.canPerform(phase).isSuccess()) {
    // stop the phase
    handler.perform(phase, "1001");
}
```

Sample usage of SpecificationSubmissionPhaseHandler:

```
SpecificationSubmissionPhaseHandler handler = new
```

```
 SpecificationSubmissionPhaseHandler(PHASE_HANDLER_NAMESPACE);
Project project = setupPhasesForSpec(true);
Phase[] phases = project.getAllPhases();
Phase phase = phases[0];

// set with scheduled status.
phase.setPhaseStatus(PhaseStatus.SCHEDULED);
// phase can be start now
If (handler.canPerform(phase).isSuccess()) {
    // start the phase
    handler.perform(phase, "operator");
}
// we will stop the phase now
// set with open status.
phase.setPhaseStatus(PhaseStatus.OPEN);
Resource reviewer = createResource(1, 102L, 1, 15);
insertResources(conn, new Resource[] {reviewer});
insertResourceInfo(conn, reviewer.getId(), 1, "3");

// create a registration
Resource resource = createResource(4, 101L, 1, 17);
insertResources(conn, new Resource[] {resource});
insertResourceInfo(conn, resource.getId(), 1, "4");

// insert upload/submission
Upload upload = createUpload(1, project.getId(), 1, 1, 1, "parameter");
insertUploads(conn, new Upload[] {upload});
Submission submission = createSubmission(1, upload.getId(), 1, 2);
insertSubmissions(conn, new Submission[] {submission});
// we can stop the phase
If (handler.canPerform(phase).isSuccess()) {
    // stop it now
    handler.perform(phase, "operator");
}
```

Please see sample email templates for new SpecificationSubmissionPhaseHandler and SpecificationReviewPhaseHandler phases in the "email templates" folder provided together with this specification.

For 1.6 version, three new handlers added, CheckpointSubmissionPhaseHandler, CheckpointScreeningPhaseHandler and CheckpointReviewPhaseHandler.

Notes, the specific values will be changed according to the database definition for new types.

Sample usage of CheckpointSubmissionPhaseHandler:

```
CheckpointSubmissionPhaseHandler handler = new
 CheckpointSubmissionPhaseHandler(PHASE_HANDLER_NAMESPACE);
Project project = setupPhases();
Phase[] phases = project.getAllPhases();

// assume the 14th phase is checkpoint submission phase.
```

```
Phase phase = phases[13];

// set with scheduled status.
phase.setPhaseStatus(PhaseStatus.SCHEDULED);
// phase can be start now
If (handler.canPerform(phase).isSuccess()) {
    // start the phase
    handler.perform(phase, "operator");
}
// we will stop the phase now
// set with open status.
phase.setPhaseStatus(PhaseStatus.OPEN);
Resource screener = createResource(1, 102L, 1, 20);
insertResources(conn, new Resource[] { screener});
insertResourceInfo(conn, screener.getId(), 1, "3");

// create a registration
Resource resource = createResource(4, 101L, 1, 19);
insertResources(conn, new Resource[] {resource});
insertResourceInfo(conn, resource.getId(), 1, "4");

// insert a submission with checkpoint submission type
Upload upload = createUpload(1, project.getId(), 1, 1, 1, "parameter");
insertUploads(conn, new Upload[] {upload});
// assume the checkpoint submission type id is 3.
Submission submission = createSubmission(1, upload.getId(), 1, 3);
insertSubmissions(conn, new Submission[] {submission});
// we can stop the phase
If (handler.canPerform(phase).isSuccess()) {
    // stop it now
    handler.perform(phase, "operator");
}
```

Sample usage of CheckpointScreeningPhaseHandler:

```
CheckpointScreeningPhaseHandler handler = new
CheckpointScreeningPhaseHandler(PHASE_HANDLER_NAMESPACE);

Project project = setupPhases();
Phase[] phases = project.getAllPhases();

// assume following positions are the desired phases.
Phase submissionPhase = phases[13];
Phase screeningPhase = phases[14];
submissionPhase.setPhaseStatus(PhaseStatus.CLOSED);
screeningPhase.setPhaseStatus(PhaseStatus.SCHEDULED);

// insert a submission with checkpoint submission type
Connection conn = getConnection();
Resource resource = createResource(4, 101L, 1, 19);
insertResources(conn, new Resource[] {resource});
Upload upload = createUpload(1, project.getId(), 4, 1, 1, "parameter");
insertUploads(conn, new Upload[] {upload});

// assume the checkpoint submission type id is 3.
Submission submission = createSubmission(1, upload.getId(), 1, 3);
insertSubmissions(conn, new Submission[] {submission});
```

```
      // it now can perform start
      if (handler.canPerform(screeningPhase).isSuccess()) {
          // start
          handler.perform(screeningPhase, "1001");
      }
      // we will close the phase now
      // create a checkpoint screener
      Resource screener = createResource(5, 102L, 1, 20);
      insertResources(conn, new Resource[] {screener});
      insertResourceInfo(conn, screener.getId(), 1, "3");
      Scorecard scorecard = createScorecard(1, 1, 2, 6, "name", "1.0", 75.0f,
100.0f);
      Review review = createReview(11, 5, 1, 1, true, 90.0f);
      insertScorecards(conn, new Scorecard[] {scorecard});
      insertReviews(conn, new Review[] {review});
      // change the phase status to open
      phase.setPhaseStatus(PhaseStatus.OPEN);
      // we can stop the phase now
      if (handler.canPerform(screeningPhase).isSuccess()) {
          // stop the phase
          handler.perform(screeningPhase, "1001");
      }

      Sample usage of CheckpointReviewPhaseHandler:

      CheckpointReviewPhaseHandler handler = new
      CheckpointReviewPhaseHandler(PHASE_HANDLER_NAMESPACE);

      Project project = setupPhases();
      Phase[] phases = project.getAllPhases();
      Phase submissionPhase = phases[13];
      Phase reviewPhase = phases[15];

      // test with scheduled status.
      reviewPhase.setPhaseStatus(PhaseStatus.SCHEDULED);

      // time has passed, but dependency not met.
      reviewPhase.setActualStartDate(new Date());

      // time has passed and dependency met.

reviewPhase.getAllDependencies()[0].getDependency().setPhaseStatus(PhaseStatus.C
LOSED);

      // And we set up a active submission in screening phase
      Connection conn = getConnection();

      Resource submitter = createResource(1, submissionPhase.getId(), 1, 19);

      Upload upload = createUpload(1, 1, submitter.getId(), 1, 1, "parameter");
      Submission submission = createSubmission(1, 1, 1, 3);

      insertResources(conn, new Resource[] {submitter});
      insertResourceInfo(conn, submitter.getId(), 1, "11111");
      insertUploads(conn, new Upload[] {upload});
      insertSubmissions(conn, new Submission[] {submission});
```

```
        // it now can perform start
        if (handler.canPerform(reviewPhase).isSuccess()) {
            // start
            handler.perform(reviewPhase, "1001");
        }

        // we will close the phase now
        // create a checkpoint reviewer
        Resource reviewer = createResource(6, reviewPhase.getId(), 1, 21);
        insertResources(conn, new Resource[] {reviewer});
        insertResourceInfo(conn, reviewer.getId(), 1, "2");
        // insert scorecard
        Scorecard sc = this.createScorecard(1, 1, 2, 1, "name", "1.0", 75.0f,
100.0f);
        insertScorecards(conn, new Scorecard[] {sc});
        Review review = createReview(1, reviewer.getId(), submission.getId(),
sc.getId(), true, 77.0f);
        insertReviews(conn, new Review[] {review});

        // change the phase status to open
        reviewPhase.setPhaseStatus(PhaseStatus.OPEN);
        // we can stop the phase now
        if (handler.canPerform(reviewPhase).isSuccess()) {
            // stop the phase
            handler.perform(reviewPhase, "1001");
        }
```

Please see sample email templates for new CheckpointSubmissionPhaseHandler, CheckpointScreeningPhaseHandler and CheckpointReviewPhaseHandler phases in the "email templates" folder provided together with this specification.

5. Future Enhancements

Additional phase handlers can be added.
Additional lookup classes can be added.