Cockpit Project Management Persistence 1.0 Component Specification

1. Design

This component provides an implementation of the ProjectPersistence interface from the Project Management component. The implementation will adapt the ContestManagerBean from the Studio Contest Manager component to provide the necessary functionality. This component will allow the Project Management component to work with studio contests.

This component is intended to be used as an adapter, to allow the Auto Pilot component to manage Studio projects normally. As such, this design takes into account how the Auto Pilot component currently makes use of the Project Management component. The Auto Pilot component makes use of Project Management in the ActiveAutoPilotSource file. It makes use of Project Management by searching for specific projects which require auto-pilot. The current design documents the needed behavior to support ActiveAutoPilotSource, as well as the suggested changes to Studio Contest Manager.

Since this component is an adapter, it makes little sense for the Studio Contest Manager to be dependent on this component (which would cause an indirect dependency on Project Management component). As such, the design and the suggested changes take care to make sure that no dependency will come into existence.

1.1 Design Patterns

The **Strategy** pattern is used by ProjectToContestConverter interface when defining the conversion method used to convert entities to/from “Project Management” and “Contest and Submission Entities” components.

This component is an **Adapter** that adapts the ProjectPersistence interface into the ContestManager interface.

1.2 Industry Standards

EJB 3.0
JNDI

1.3 Required Algorithms

The following tables will outline the properties in Contest and Submission Entities component, and the corresponding property in the Project Management component. It will specify the method(s) used in each class to get/set the property, and some special notes (if necessary):

1.3.1 Converting Contest to Project (and vice-versa)

NOTE: Project setter with a "*", also includes the constructor method parameter in parenthesis. Developer should use the constructor method parameter to initialize the property instead of the setter method when creating a Project.

Contest Setter	Contest Getter	Project Setter	Project Getter	Notes
setContestId	getContestId	setId* (projectId)	getId	Project id is restricted to positive numbers, while contested is unrestricted. Developer should note that contest id should also be positive for compatibility with project id. When creating a Project, if contestId is null, developer must use constructor that doesn't accept projectId to indicate unspecified id.
setName	getName	setProperty	getProperty	The String constant PROPERTY_CONTEST_NAME is used as the key for the Project property.
setContest Channel	getContest Channel	setProject Category* (projectCategory)	getProject Category	See Section 1.3.2 on how to convert ContestChannel to ProjectCategory
setProjectId	getProjectId	setProperty	getProperty	The String constant PROPERTY_PROJECT_ID is used as the key for the Project property.
setTcDirect ProjectId	getTcDirect ProjectId	setProperty	getProperty	The String constant PROPERTY_TC_DIRECT_PROJECT_ID is used as the key for the Project property.
setStatus	getStatus	setProject Status* (projectStatus)	getProject Status	See the Section 1.3.3 on how to convert ContestStatus into ProjectStatus
setForumId	getForumId	setProperty	getProperty	The String constant PROPERTY_FORUM_ID is used as the key for the Project property.
setEventId	getEventId	setProperty	getProperty	The String constant PROPERTY_EVENT_ID is used as the key for the Project property.
setSubmissions	getSubmissions	setProperty	getProperty	The String constant PROPERTY_SUBMISSIONS is used for the Project Property.
setFileTypes	getFileTypes	setProperty	getProperty	The String constant PROPERTY_FILE_TYPES is used for the Project Property.
setResults	getResults	setProperty	getProperty	The String constant PROPERTY_RESULTS is used

				for the Project Property.
setDocuments	getDocuments	setProperty	getProperty	The String constant PROPERTY_DOCUMENTS is used for the Project Property.
setConfig	getConfig	setProperty	getProperty	The String constant PROPERTY_CONFIG is used for the Project Property.
setContestType	getContestType	setProperty	getProperty	The String constant PROPERTY_CONTEST_ TYPE is used for the Project Property.
setStartDate	getStartDate	setProperty	getProperty	The String constant PROPERTY_START_DATE is used for the Project Property.
setEndDate	getEndDate	setProperty	getProperty	The String constant PROPERTY_END_DATE is used for the Project Property.
setWinner Announcement Deadline	getWinner Announcement Deadline	setProperty	getProperty	The String constant PROPERTY_WINNER_ANNOUNCEMENT_DEADLINE is used for the Project Property.
setCreatedUser	getCreatedUser	setCreation User	getCreation User	The Project accessor methods are inherited from the base class AuditableObject. Developer should note that the Project.creationUser is a String, while Contest.createdUser is a Long. The Project should contain values that are parseable to Long for compatibility purposes.

1.3.2 Converting ContestChannet to ProjectCategory (and vice-versa)

NOTE: ProjectCategory setter with a "*", also includes the constructor method parameter in parenthesis. Developer should use the constructor method parameter to initialize the property instead of the setter method when creating a ProjectCategory.

Contest Channel Setter	Contest Channel Getter	Project Category Setter	Project Category Getter	Notes
setContest ChannelId	getContest ChannelId	setId* (id)	getId	ProjectCategory id is restricted to positive numbers, while ContestChannel id is unrestricted. Developer should note that ContestChannel id should also be positive for compatibility.
setDescription	getDescription	setDescription	getDescription	ContestChannel allows null description, while ProjectCategory does not. Empty String description ("") is used for

				ProjectCategory description if a null is found in ContestChannel description.
setName	getName	setName* (name)	getName	ContestChannel allows null/empty String name, while ProjectCategory does not. Developer should note that ContestChannel name should also be not-null and not empty String for compatibility.
setParent ChannelId	getParent ChannelId	N/A	N/A	This property is dropped when converting from ContestChannel to ProjectCategory.
setFileType	getFileType	setProject Type* (projectType)	getProject Type	See Section 1.3.4 on how to convert a StudioFileType into a ProjectType

1.3.3 Converting ContestStatus to ProjectStatus (and vice-versa)

NOTE: ProjectStatus setter with a "*", also includes the constructor method parameter in parenthesis. Developer should use the constructor method parameter to initialize the property instead of the setter method when creating a ProjectStatus.

ContestStatus Setter	ContestStatus Getter	ProjectStatus Setter	ProjectStatus Getter	Notes
setContestStatusId	getContestStatusId	setId* (id)	getId	ProjectStatus id is restricted to positive numbers, while ContestStatus is unrestricted. Developer should note that ContestStatus id should also be positive for compatibility.
setDescription	getDescription	setDescription	getDescription	ContestStatus allows null description, while ProjectStatus does not. An empty String ("") description is specified to the ProjectStatus in the case of null description.
setStatuses	getStatuses	N/A	N/A	This property is dropped when converting ContestStatus to ProjectStatus

setName	getName	setName* (name)	getName	ContestStatus allows null name, while ProjectStatus does not. Developer should note that ContestStatus name should also be not null and not empty String for compatibility.

1.3.4 Converting StudioFileType to ProjectType (and vice-versa)

NOTE: ProjectType setter with a "*", also includes the constructor method parameter in parenthesis. Developer should use the constructor method parameter to initialize the property instead of the setter method when creating a ProjectType.

StudioFileType Setter	StudioFileType Getter	ProjectType Setter	ProjectType Getter	Notes
setStudioFileType	getStudioFileType	setId* (id)	getId	ProjectType is restricted to positive numbers, while StudioFileType is not. Developer should note that StudioFileType should also have positive values for compatibility.
setMimeTypes	getMimeTypes	N/A	N/A	This property is dropped when converting StudioFileType to ProjectType
setSort	getSort	N/A	N/A	This property is dropped when converting StudioFileType to ProjectType
setImageFile	isImageFile	N/A	N/A	This property is dropped when converting StudioFileType to ProjectType.
setExtension	getExtension	setName* (name)	getName	ProjectType.name is restricted to not null and not empty String, while StudioFileType.extension is not. Developer should note that extension should also be not null and not empty for compatibility.
setDescription	getDescription	setDescription	getDescription	ProjectType description is restricted to non-nulls, while StudioFileType description is not. If a null StudioFileType

				description is found, then an empty String will be provided to the ProjectType.

1.3.5 Converting Project Filters to Contest Filters

It is necessary to support the searchProjects feature, since the ActiveAutoPilotSource uses certain search features, and the PM would not allow a different implementation of the AutoPilot interface. The following table lists the Filter type and name built using ProjectFilterUtility, and lists the corresponding Filter type and name to convert it to. Note that this also depends on the changes that are outlined in Section 1.3.6 (suggested changes to the TC Studio Contest Manager 1.0 component).

The values of the Project Filter are copied into the converted Studio Filter (unless the notes say otherwise).

ProjectFilterUtility Filter Name values are retrieved as constants from the ProjectFilterUtility class.

ProjectFilterUtility Filter Type	ProjectFilterUtility Filter Name	Studio Filter Type	Studio Filter Name	Notes
EqualToFilter	PROJECT_TYPE_ID	EqualToFilter	STUDIO_FILE_TYPE_ID	Studio search should look for contests whose contest channel have a file type id specified.
InFilter	PROJECT_TYPE_ID	InFilter	STUDIO_FILE_TYPE_ID	Same as previous row.
EqualToFilter	PROJECT_TYPE_NAME	EqualToFilter	STUDIO_FILE_TYPE_EXTENSION	Studio search should look for contests whose contest channel has a file type
InFilter	PROJECT_TYPE_NAME	InFilter	STUDIO_FILE_TYPE_EXTENSION	Same as previous row.
EqualToFilter	PROJECT_CATEGORY_ID	EqualToFilter	STUDIO_CONTEST_CHANNEL_ID	Studio search should look for contests with contest channel of specified id.
InFilter	PROJECT_CATEGORY_ID	InFilter	STUDIO_CONTEST_CHANNEL_ID	Same as previous row.
EqualToFilter	PROJECT_CATEGORY_NAME	EqualToFilter	STUDIO_CONTEST_CHANNEL_NAME	Studio search should look for contests with contest channel of specified name.
InFilter	PROJECT_CATEGORY_NAME	InFilter	STUDIO_CONTEST_CHANNEL_NAME	Same as previous row.
EqualToFilter	PROJECT_STATUS_ID	EqualToFilter	STUDIO_CONTEST_	Studio search

			STATUS_ID	should look for contests with contest status of specified id.
InFilter	PROJECT_STATUS_ID	InFilter	STUDIO_CONTEST_STATUS_ID	Same as previous row.
EqualToFilter	PROJECT_STATUS_NAME	EqualToFilter	STUDIO_CONTEST_STATUS_NAME	Studio search should look for contests with contest status of specified name.
InFilter	PROJECT_STATUS_NAME	InFilter	STUDIO_CONTEST_STATUS_NAME	Same as previous row.
EqualToFilter	PROJECT_PROPERTY_NAME	See Notes	See Notes	If the value is equal to any of the constants defined in ProjectToContestConverterImpl class, then return an always true filter (a NotFiltee wrapping a NullFilter for STUDIO_CONTEST_ID). Otherwise, return an always false filter (a NullFilter for STUDIO_CONTEST_ID).
InFilter	PROJECT_PROPERTY_NAME	See Notes	See Notes	Same as previous row.
EqualFilter	PROJECT_PROPERTY_VALUE	See Notes	See Notes	This should be an OrFilter, comprised of several EqualFilters. Further notes are given after the table.
InFilter	PROJECT_PROPERTY_VALUE	See Notes	See Notes	This should be an OrFilter, comprised of several InFilters. Further notes are given after the table.
AndFilter	N/A	AndFilter	N/A	The inner filters of the input Filter should be processed according to this

				table, and then inserted into the output AndFilter.
OrFilter	N/A	OrFilter	N/A	The inner filters of the input Filter should be processed according to this table, then inserted into the output OrFilter.
NotFilter	N/A	NotFilter	N/A	The inner filters of the input Filter should be processed according to this table, then inserted into the output NotFilter.

Handling Special AndFilters

The converter should be able to handle a special type of AndFilter, as produced by *ProjectUtilityFilter.buildPropertyProjectEqualFilter*.

In order to do this, the converter MUST first check each AndFilter, and verify if the size of the compound filters list is 2. If yes, then the converter must proceed to check if the contents of the AndFilter are an EqualToFilter with column name "PROJECT_PROPERTY_NAME" and an EqualToFilter with column name "PROJECT_PROPERTY_VALUE". If yes, then the converted form of the given AndFilter should be an EqualToFilter. The column name of the converted Filter is dependent on the PROJECT_PROPERTY_NAME filter, and the value of the converted Filter is copied from PROJECT_PROPERTY_VALUE.

Refer to the table below to determine the column name to use:

PROJECT_PROPERTY_NAME Filter value [Attributes from ProjectToContestConverterImpl]	Converted Filter Column Name
PROPERTY_CONTEST_NAME	STUDIO_CONTEST_NAME
PROPERTY_PROJECT_ID	STUDIO_CONTEST_PROJECT_ID
PROPERTY_TC_DIRECT_PROJECT_ID	STUDIO_CONTEST_TC_DIRECT_PROJECT_ID
PROPERTY_FORUM_ID	STUDIO_CONTEST_FORUM_ID
PROPERTY_EVENT_ID	STUDIO_CONTEST_EVENT_ID
PROPERTY_START_DATE	STUDIO_CONTEST_START_DATE
PROPERTY_END_DATE	STUDIO_CONTEST_END_DATE
PROPERTY_WINNER_ANNOUNCEMENT_DEADLINE	STUDIO_CONTEST_WINNER_ANNOUNCEMENT_DEADLINE
autoPilotSwitchPropertyName	Call buildAutoPilotSwitchFilter method with value from PROPERTY_PROPERTY_VALUE Filter.

Values that are not present in the table may be safely ignored. This is because it will not be possible to search for the other properties (Collections of objects) conveniently without specifying additional Filters. It will not be possible to construct such with existing ProjectFilterUtility.

Handling PROJECT_PROPERTY_VALUE filters

Since the ProjectToContestConverterImpl converts certain contest properties into Project properties, a PROJECT_PROPERTY_VALUE filter will be converted into an AndFilter, with 8 inner EqualTo or InFilters. The values of the inner filters will be copied from the original Filter, and the column names of the inner filters will be as follows:

- PROPERTY_CONTEST_NAME
- PROPERTY_PROJECT_ID
- PROPERTY_TC_DIRECT_PROJECT_ID
- PROPERTY_FORUM_ID
- PROPERTY_EVENT_ID
- PROPERTY_START_DATE
- PROPERTY_END_DATE
- PROPERTY_WINNER_ANNOUNCEMENT_DEADLINE

This approach allows us to simulate the behavior of an actual PROJECT_PROPERTY_VALUE filter.

Handling Unrecognized Filters

Filters are used to restrict the set of returned results. For unrecognized Filters, the default converter will convert it into an “always true” filter. This means that unrecognized Filters do not restrict the set of returned results (and are effectively ignored). An example “always true” would be a NotFilter wrapping a NullFilter for STUDIO_CONTEST_ID property.

Note that for compound filters, it may be more efficient if developer simply didn’t include the unrecognized filter in the converted compound filter (if this is possible).

1.3.6 Suggested Changes to Studio Contest Manager Component

The following lists the changes to Studio Contest Manager that are needed to support the design:

- Add a searchContests(Filter filter) method to ContestManager and implementations. The following Filters need to be supported:
    - STUDIO_FILE_TYPE_ID
      This should search for Contests with ContestChannels whose StudioFileType ids match the specified value.

    - STUDIO_FILE_TYPE_EXTENSION
      This should search for Contests with ContestChannels whose StudioFileType extensions match the specified value.

    - STUDIO_CONTEST_CHANNEL_ID
      This should search for Contests with ContestChannels whose ids match the specified value.

    - STUDIO_CONTEST_CHANNEL_NAME
      This should search for Contests with ContestChannels whose names match the specified value.

    - STUDIO_CONTEST_STATUS_ID
      This should search for Contests with ContestStatus whose ids match the specified value.

    - STUDIO_CONTEST_STATUS_NAME
      This should search for Contests with ContestStatus whose names match the specified value. Note that this is the only property that needs to be supported for the Studio Contest Management component to work with Auto Pilot component.

    - STUDIO_CONTEST_ID
      This should search for Contests with ids that match the specified value.

- STUDIO_CONTEST_NAME
  This should search for Contests with names that match the specified value.

- STUDIO_CONTEST_PROJECT_ID
  This should search for Contests with projectIds that match the specified value.

- STUDIO_CONTEST_DIRECT_PROJECT_ID
  This should search for Contests with directProjectIds that match the specified value.

- STUDIO_CONTEST_FORUM_ID
  This should search for Contests with contestForumIds that match the specified value.

- STUDIO_CONTEST_EVENT_ID
  This should search for Contests with contestEventIds that match the specified value.

- STUDIO_CONTEST_START_DATE
  This should search for Contests with start dates that match the specified value.

- STUDIO_CONTEST_END_DATE
  This should search for Contests with end dates that match the specified value.

- STUDIO_CONTEST_WINNER_ANNOUNCEMENT_DEADLINE
  This should search for Contests with winner announcement deadlines that match the specified value.

1.3.7 Handling Conversion Exception

ConversionException is thrown in the cases where it is impossible to convert from one entity to the other. This usually occurs when a Contest property value is null or invalid when converted to a Project property value. The following section will list the situations when Conversion Exception must be thrown. If a given situation is not listed here, then the developer must clarify the issue in the forums:

Converting Contest to Project

- If contestId is less than 0. (Note that in Project context, ‘0’ id means unspecified id)
- If contestChannel is null. (Project disallows null ProjectCategory)
- If contestStatus is null. (Project disallows null ProjectStatus)

All other Contest properties may be null or any value, since Project allows any sort of custom property value.

Converting Project to Contest

- If Project contains a creationUser that is not parseable to a Long.

Converting ContestChannel to ProjectCategory

- If contestChannelId is null, or less than or equal to 0. (ProjectCategory disallows non-positive ids, and uses primitive data-types, so null cannot be specified either).
- If ContestChannel name is null or an empty String. (ProjectCategory disallows null or empty String name).
- If ContestChannel.fileType is null. (ProjectCategory disallows null ProjectType).

Converting ContestStatus to ProjectStatus

- If contestChannelId is null, or less than or equal to 0. (ProjectStatus disallows non-positive ids, and uses primitive data-types, so null cannot be specified either)
- If ContestStatus name is null or an empty String. (ProjectStatus disallows null or empty String name).

Converting StudioFileType to ProjectType

- If StudioFileType.studioFileType is null or less than or equal to 0. (ProjectType disallows non-positive ids, and uses primitive data-types, so null cannot be specified either).
- If StudioFileType.extension is null or an empty String. (ProjectType disallows null or empty String names).

1.4 Component Class Overview

Package com.topcoder.management.project.studio

ProjectToContestConverter [Interface]

This is an interface that defines the Strategy for converting entities to/from "Project Management" and "Contest and Submission Entities" components. It offers methods of converting the different data classes between each. It also offers a method of converting the Filter names built by Project Filter Utility, into acceptable Filter names by the Contest Manager component.

ContestManagerProjectAdapter

This is an implementation of the ProjectManager interface from the "TC Project Management" 1.0 component. This implementation will transform the Project entities from TC Project Management to/from the Contest entities found in TC Contest and Submission Entities 1.0 component. Once transformed, this implementation will then delegate the actual operation to an instance of a ContestManager (from "TC Studio

Contest Manager" 1.0 component). This allows the Studio Contests to be managed as TC Projects, and allows the Studio application to be used in conjunction with the TC Auto Pilot component.

Package com.topcoder.management.project.studio.converter

ProjectToContestConverterImpl

This class is the default way of converting Project entities from "TC Project Management" component into Contest entities from "TC Contest and Submission Entities" component. It will use the methods outlined in the algorithm section of the CS to perform the conversion.

1.5 Component Exception Definitions

ConversionException

This exception is thrown if a converter is provided with an object that it cannot accurately convert.

1.6 Thread Safety

This component is designed such that it introduces no additional thread-safety issues on top of existing thread-safety conditions of Studio Contest Manager component. All classes in the design maintain state that is intended to be changed only during the initialization of the component. The “work” methods of the component are stateless, and therefore thread-safe.

Note that the current documentation of the Studio Contest Manager states that the thread-safety is dependent on the implementation. This means that if an non-thread-safe implementation is used, then this component will also not be thread-safe, and it is necessary for synchronization or multiple instances to be used.

2. Environment Requirements

2.1 Environment

- Development language:
  - Java 1.5
- Compile Target:
  - Java 1.5
  - Java 1.6

2.2 TopCoder Software Components

  - TopCoder Base Exception 2.0 – This is used to provide the base exception for the custom exceptions defined.

- TopCoder Project Management 1.0 – The component provides an implementation of the ProjectManager interface from this component. The Project data entities also come from this component.
- TopCoder Contest and Submission Entities 1.0 – This is used to provide the Contest data entities that are used.
- TopCoder Studio Contest Manager 1.0 – This is used to provide the ContestManager class to delegate the management calls to.
- TopCoder Search Builder 1.3.2 – The Search Builder component provides the Filter classes which are transformed by this component.

2.3 Third Party Components

None

3. Installation and Configuration

3.1 Package Names

com.topcoder.management.project.studio
com.topcoder.management.project.studio.converter

3.2 Configuration Parameters

All configuration is done through the constructor, and via setters. It should be possible to configure the given classes using Object Factory.

3.3 Dependencies Configuration

N/A

4. Usage Notes

4.1 Required steps to test the component

- Extract the component distribution.
- Follow Dependencies Configuration.
- Execute 'ant test' within the directory that the distribution was extracted to.

4.2 Required steps to use the component

- Instantiate the ContestManagerProjectAdapter
- Invoke the methods on the adapter and use it like you would normally use a ProjectManager implementation. The calls are automatically adapted to the ContestManager.

4.3 Demo

```
   long projectId = 1;
    long categoryId = 2;
    long statusId = 3;

    // Initialize the Adapter
    ProjectManager manager = new
ContestManagerProjectAdapter("java:comp/env/contestManagerRemote");

    // Also create a ContestManager for reference
    InitialContext ic = new InitialContext();
    ContestManager contestMngr = (ContestManager)
ic.lookup("java:comp/env/contestManagerRemote");

    // Initialize start/end dates.
    Calendar startDate = Calendar.getInstance();
    Calendar endDate = Calendar.getInstance();

    // Project ends in one week.
    endDate.add(Calendar.DATE, 7);

    // Create a Project
    ProjectType projectType = new ProjectType(projectId, "Flash", "Macromedia
Flash Project");

    ProjectCategory category = new ProjectCategory(categoryId, "Open", "Open To
All", projectType);

    ProjectStatus regStatus = new ProjectStatus(statusId, "Registration",
"Registration Phase");

    Project targetProject = new Project(category, regStatus);

    // Set some custom properties in the Project
    targetProject.setProperty(ProjectToContestConverterImpl.PROPERTY_START_DATE,
startDate.getTime());

    targetProject.setProperty(ProjectToContestConverterImpl.PROPERTY_END_DATE,
endDate.getTime());

    // Create the Project in the manager
    manager.createProject(targetProject, "Ignored Parameter");

    // Result should be visible in contest manager
    Contest contest = contestMngr.getContest(projectId);

    // Should be project id
    System.out.println(contest.getContestId());

    // Should be start date
    System.out.println(contest.getStartDate());

    // Should be end date
    System.out.println(contest.getEndDate());

    // Should be equivalent to Project Status
```

```
    ContestStatus contestStatus = contest.getStatus();

    System.out.println(contestStatus.getName());
    System.out.println(contestStatus.getDescription());
    System.out.println(contestStatus.getContestStatusId());

    // Update the Status to Screening
    ProjectStatus screenStatus = new ProjectStatus(4, "Screening", "Screening
Phase");

    targetProject.setProjectStatus(screenStatus);

    // Status should be updated in contest manager too
    contest = contestMngr.getContest(projectId);

    contestStatus = contest.getStatus();

    System.out.println(contestStatus.getName());
    System.out.println(contestStatus.getDescription());
    System.out.println(contestStatus.getContestStatusId());

    // Update the end date in the contest manager
    contest.setEndDate(new Date());

    contestMngr.updateContest(contest);

    // Result should also be visible in Project Manager
    targetProject = manager.getProject(projectId);

    // Should be the new end date.
    targetProject.getProperty(ProjectToContestConverterImpl.PROPERTY_END_DATE);

    // Perform a search
    Filter filter = ProjectFilterUtility.buildCategoryIdEqualFilter(categoryId);

    // Should return our existing project
    Project[] results = manager.searchProjects(filter);

    // Get all Project Categories
    ProjectCategory[] categories = manager.getAllProjectCategories();

    // Get all Project Types
    ProjectType[] projectCategories = manager.getAllProjectTypes();

    // Get all Project Statuses
    ProjectStatus[] projectStatus = manager.getAllProjectStatuses();
```

5. Future Enhancements

None