This page last changed on Sep 30, 2008 by ghostar.

1. Scope

1.1 Overview

This component provides a framework for adding projects and versions to JIRA, as well as retrieving information about pages in JIRA. Access to the JIRA service will be through the JIRA SOAP API, and authorization / authentication will happen through that API as well.
Information about the JIRA SOAP API can be found here:
http://confluence.atlassian.com/display/JIRA/Creating+a+SOAP+Client
http://docs.atlassian.com/software/jira/docs/api/rpc-jira-plugin/latest/com/atlassian/jira/rpc/soap/JiraSoapService.html

1.2 Logic Requirements

This component must implement all the functionality shown on the "Manager Class Diagram" in the provided interface TCUML.

1.2.1 Interfaces

The following interface will be provided by this component:

- JiraManager

o This interface defines functionality for logging a user in and out of JIRA, as well as creating new project and versions, as well as retrieving information about projects and versions created.

1.2.2 Accessing the SOAP services

Sample code in accessing the SOAP services can be found through the JIRA links above. If the designer wishes, Axis can be used in this design to generate the SOAP client class used to access the JIRA SOAP services.

1.2.3 Authorization

The authorization for users to be able to retrieve information and create pages in specific locations will be handled by Jira. Before making an API call, the caller is expected to login with a username and password, returning a token. That same token is then passed to the subsequent calls into Jira. This token is used to verify the user has access to the specific areas being manipulated. If a user attempts an unauthorized call, a JiraNotAuthorizedException should be thrown.

1.2.4 Token expiration

If the token obtained for authorization expires or becomes stale, a JiraSecurityTokenExpiredException should be thrown.

1.2.5 Mapping

The DefaultJiraManager class contains mappings of ContestType to string names, used to retrieve the notification scheme, permission scheme, and issue security scheme values for each project created.
When the project is created, the notification scheme, permission scheme, and issue security scheme are required to be set in the RemoteProject instance used. The schemes can be cached, as an enhancement, but this is not a requirement.
If no mapping is given for a specific ContestType for any of the scheme names, the following defaults should be used:

Notification Scheme Names:

Component Type	Name
Generic / Custom	"Component Management Notification Scheme"
Application	"Client Project Notification Scheme"

Permission Scheme Names:

Component Type	Name
Generic	"Generic Component Permission Scheme"
Application	"Client Projects"
Custom	"Custom Component Permissions Scheme

Issue Security Scheme Names:

Component Type	Name	
Generic	"Generic Component Issues Scheme"	
Application	"Client Project Issues Scheme"	
Custom	"Custom Component Issues Scheme"	

1.2.6 Entities

The two result classes (JiraProjectRetrievalResult, and JiraProjectCreationResult), as well as the entity classes (JiraVersion, JiraProject) are just basic data holders that represent the entities used by the framework. They should all implement the Serializable interface. Also, each entity must be serializable to XML, for use with the JIRA Service component. This should be done through the use of class annotations, like ""@XmlAccessorType(XmlAccessType.FIELD)", "@XmlType(name = "..." propOrder = "...")

1.2.7 Creating the project and version

The flow for creating a project and version can have a couple of different paths. When creating a project, the user provides the base project information and a version. If the base project already exists, a new version page is added, but if the base project doesn't already exist, it is first created, and then the version page is added under it. A creation result is returned, indicating the action taken, as well as the entities provided, with updated ID's.

1.2.8 Connection

The connection to JIRA could conceivably be lost at various points during the usage of this component. If this happens, the manager should attempt once to reestablish the connection, and if that fails, a JiraConnectionException should be thrown, and on each subsequent calls, an attempt should be made to reestablish the connection. If the reestablishment of the connection fails, another JiraConnectionException should be thrown, until the connection has been reestablished. This way, when the connection is available again, the API will function as normal.

1.2.9 Logging

Each manager method called should be logged at the DEBUG level, including the parameter information. If errors occur in any call, a message should be logged at the WARNING level before the exception is re-thrown.

1.2.10 Configuration

Configuration must occur through the Configuration API and the Configuration API Object Factory Plugin. Besides the normal ConfigurationObject constructors for the classes that involve configuration, constructors should be added that use the Configuration Persistence component to retrieve configuration information saved in configuration files.

1.3 Required Algorithms

The flow of creating the project and version, with the various different conditions, must be shown and described in the CS.

1.4 Example of the Software Usage

This component will be used as a direct API for manipulating JIRA information.

1.5 Future Component Direction

Further manager implementations can be added.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

Designs must adhere to the interface diagram definition found in the architecture TCUML file provided [JIRA_Services_Architecture.tcuml]. Designers can choose to add more methods to the interfaces, but must keep the ones defined on the diagram as a minimum. Changes to the interfaces should be approved in the forum.

2.1.3 Environment Requirements

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

2.1.4 Package Structure

com.topcoder.jira

3. Software Requirements

3.1 Administration Requirements

3.1.1 *What elements of the application need to be configurable?*

- The log to use
- The connection information for the JIRA API
- The mapping of template and location strings for the different page types

3.2 Technical Constraints

3.2.1 *Are there particular frameworks or standards that are required?*

Jira 3.1.1+SOAP

3.2.2 *TopCoder Software Component Dependencies:*

- Configuration API 1.0
- Configuration Persistence 1.0.1
- Base Exception 2.0
- Logging Wrapper 2.0

**Please review the TopCoder Software component catalog for existing components that can be used in the design.

3.2.3 *Third Party Component, Library, or Product Dependencies:*

JIRA 3.1.1

3.2.4 *QA Environment:*

- Solaris 7
- RedHat Linux 7.1
- Windows 2000- Windows 2003
- Informix 10.0

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines.

3.4 Required Documentation

3.4.1 *Design Documentation*

- Use-Case Diagram
- Class Diagram
- Sequence Diagram
- Component Specification

3.4.2 Help / User Documentation

- Design documents must clearly define intended component usage in the 'Documentation' tab of the TopCoder UML Tool.