Confluence Management 1.0 Component Specification

1. Design

This component provides a framework for adding pages to Confluence, as well as retrieving information about pages in Confluence. Access to the Confluence service will be through the Confluence SOAP API, and authorization / authentication will happen through that API as well.
This design has improvement to retry user calls in case of failure(please refer 1.3.4), which is configurable.

1.1 Design Patterns

Strategy pattern is used when creating ConfluenceService instance.

1.2 Industry Standards

SOAP

1.3 Required Algorithms

1.3.1 Design details

Information about the Confluence SOAP API can be found here:
http://confluence.atlassian.com/display/DOC/Remote+API+Specification

The Base Page is located in the wiki based on the space mapping described above. The page name is constructed per the table below:

Type	**Page Name**
Component Base Page	Catalog+Type+Name
Component Version Page	Catalog+Type+Name+Version
Application Base Page	ApplicationCode+Name
Application Version Page	ApplicationCode+Name+Version

1.3.2 Obtaining ConfluenceSoapService

WSDL file for the Confluence Service:
http://confluence.atlassian.com/rpc/soap-glue/confluenceservice-v1.wsdl.

Developer should use WSDL2Java tool from the Axis2 distribution to generate classes:
WSDL2Java -uri confluence.wsdl.
You can refer to the http://ws.apache.org/axis2/1_4_1/quickstartguide.html for more details.
These classes should be added to the component's classpath.

Then, to create soap service, use:
confluenceService = new ConfluenceSoapServiceStub(confluenceUrl);

1.3.3 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 rethrown.

1.3.4 Connection

The connection to Confluence could conceivably be lost at various points during the usage of this component. If this happens, a ConfluenceConnectionException should be thrown, and on each subsequent call, an attempt should be made to reestablish the connection. If the reestablishment of the connection fails, another ConfluenceConnectionException should be thrown. This way, when the connection is available again, the API will function as normal.

So, each service call should be retried of possible.
This functionality can be achieved this way: each call to the confluence service in the DefaultConfluenceManager should be done as follows:

```
method (params…) {
    // need to make call to ConfluenceService: confluenceService.doSomething(…);
    int retriesLeft = maxRetriesNumber;
    String causeMessage = null;
    do {
        try {
            // trying to make service call
            confluenceService.doSomething(…);
            break;
        } catch (RemoteException e) {
            --retriesLeft;
            causeMessage = e.getMessage();
            // try to re-establish the connection
            try {
                confluenceService = new ConfluenceSoapServiceStub(confluenceUrl);
// please look into e.getMessage() to find out if exception was due to connection problems or not. if not,
//throw the most suitable exception (it can be ConfluenceAuthenticatedException,
//ConfluenceNotAuthorizedException etc, or even ConfluenceManagerException). if yes (connection
//problem), continue trying to reconnect;
            } catch (RemoteException ex) {
                // this is surely connection exception
            }

        }
    } while(retriesLeft > 0);

    if(retriesLeft == 0) {
        throw new ConfluenceConnectionException(“connection to Confluence service failed: ”
+ causeMessage);
    }
}
```

1.3.5 *Creation of Confluence page for component*

boolean basePageCreated = false, versionPageCreated = false;
All space characters in pageName should be replaced with ‘+’.
Construct the basePageName & versionPageName:
basePageName = catalog.getStringName() + ‘+’ + componentType.getStringName() + ‘+’ + pageName;
versionPageName = basePageName + ‘+’ + version;

1) get the page space according to asset type:
    String space = spaceLocations.get(assetType);
2) make the call to Confluence service to determine if base page exists:
    RemotePage basePage = confluenceService.getPage(token, space, basePageName);
3) if base page does not exist:

- create the base page:
  basePage = new RemotePage();
- get the url to template file according to page type:
  String templatePath = templates.get(ConfluencePageType.COMPONENT_BASE_PAGE);
- get the template page content:
  String templateContent = confluenceService.getPage(token, space, templatePath).getContent();
- set retrieved content to the base page:
  basePage.setContent(templateContent);
- set base page properties:
  basePage.setSpace(space);
  basePage.setTitle(basePageName);
- create the base page in the Confluence service:
  confluenceService.storePage(token, basePage);
- basePageCreated = true;

4) make the call to Confluence service to determine if version page exists:
RemotePage versionPage = confluenceService.getPage(token, space, versionPageName);
5) if version page does not exist:

- create the version page:
  versionPage = new RemotePage();
- get the url to template file according to page type:
  String templatePath = templates.get(ConfluencePageType.COMPONENT_VERSION_PAGE);
- get the template page content:
  String templateContent = confluenceService.getPage(token, space, templatePath).getContent();
- set retrieved content to the base page:
  versionPage.setContent(templateContent);
- set version page properties:
  versionPage.setSpace(space);
  versionPage.setTitle(versionPageName);
  versionPage.setVersion(Integer.parseInt(version));
- create the version page in the Confluence service:
  confluenceService.storePage(token, versionPage);
- versionPageCreated = true;

6) create the Page object and set its values based on basePage and versionPage (this object should map to basePage and contain url to the version page).
Page resultPage = …;
resultPage.setBasePageUrl(basePage.getUrl());
resultPage.setVersionUrl(versionPage.getUrl());

7) determine action - ConfluencePageCreatedAction:
if(basePageCreated && versionPageCreated) action = ConfluencePageCreatedAction. BASE_PAGE_AND_VERSION_CREATED;
….
8) Create the result - ConfluencePageCreationResult object using resultPage and action.
9) return result.

1.3.6 Creation of Confluence page for application

boolean basePageCreated = false, versionPageCreated = false;
All space characters in pageName should be replaced with ‘+’.
Construct the basePageName & versionPageName:
basePageName = applicationCode + ‘+’ + pageName;
versionPageName = basePageName + ‘+’ + version;

1) get the page space according to asset type and replace $CODENAME$ with applicationCode value (will only occur for asset type = APPLICATION_SPECIFICATION):
String space = spaceLocations.get(assetType);
space = space.replace(“$CODENAME$”, applicationCode);
2) make the call to Confluence service to determine if base page exists:
RemotePage basePage = confluenceService.getPage(token, space, basePageName);
3) if base page does not exist:

- create the base page:
  basePage = new RemotePage();
- get the url to template file according to page type:
  String templatePath = templates.get(ConfluencePageType.APPLICATION_BASE_PAGE);
- get the template page content:
  String templateContent = confluenceService.getPage(token, space, templatePath).getContent();
- set retrieved content to the base page:
  basePage.setContent(templateContent);
- set base page properties:
  basePage.setSpace(space);
  basePage.setTitle(basePageName);
- create the base page in the Confluence service:
  confluenceService.storePage(token, basePage);

```
    •   basePageCreated = true;
4) make the call to Confluence service to determine if version page exists:
        RemotePage versionPage = confluenceService.getPage(token, space, versionPageName);
5) if version page does not exist:
    •   create the version page:
        versionPage = new RemotePage();
    •   get the url to template file according to page type:
        String templatePath = templates.get(ConfluencePageType.APPLICATION_VERSION_PAGE);
    •   get the template page content:
        String templateContent = confluenceService.getPage(token, space, templatePath).getContent();
    •   set retrieved content to the base page:
        versionPage.setContent(templateContent);
    •   set version page properties:
        versionPage.setSpace(space);
        versionPage.setTitle(versionPageName);
        versionPage.setVersion(Integer.parseInt(version));
    •   create the version page in the Confluence service:
        confluenceService.storePage(token, versionPage);
    •   versionPageCreated = true;
6) create the Page object and set its values based on basePage and versionPage (this object should
map to basePage and contain url to the version page).
        Page resultPage = …;
7) determine action - ConfluencePageCreatedAction:
        if(basePageCreated && versionPageCreated) action = ConfluencePageCreatedAction.
BASE_PAGE_AND_VERSION_CREATED;
        ….
8) Create the result - ConfluencePageCreationResult object using resultPage and action.
9) return result.
```

1.3.7 *Retrieval of Confluence page for component*

All space characters in pageName should be replaced with ‘+’.
Construct the base page name:
basePageName = catalog.getStringName() + ‘+’ + componentType.getStringName() + ‘+’ + pageName;
Construct the full page name:
fullPageName = basePageName + ‘+’ + version;

```
1) get the page space according to asset type:
        String space = spaceLocations.get(assetType);
2) make the call to Confluence service to determine if page exists:
        RemotePage page = confluenceService.getPage(token, space, fullPageName);
3) if the page does not exist:
    •   return null;
4) get the base page (we need this to get base page url)
        RemotePage basePage = confluenceService.getPage(token, space, basePageName);
5) create resulting page
        Page result = new Page();
6) set resulting page properties according to remote page and method arguments:
        result.setContent(page.getContent());
        result.setVersionUrl(page.getUrl());
        result.setBasePageUrl(basePage.getUrl());
        result.setAssetName(pageName);
        result.setAssetType(assetType);
        ....
```

1.3.8 *Retrieval of Confluence page for application*

All space characters in pageName should be replaced with ‘+’.
Construct the base page name:
basePageName = applicationCode + ‘+’ + pageName;
Construct the full page name:

```
fullPageName = basePageName + ‘+’ + version;


1) get the page space according to asset type and replace $CODENAME$ with applicationCode value
(will only occur for asset type = APPLICATION_SPECIFICATION):
        String space = spaceLocations.get(assetType);
        space = space.replace(“$CODENAME$”, applicationCode);

2) make the call to Confluence service to determine if page exists:
        RemotePage page = confluenceService.getPage(token, space, fullPageName);
3) if the page does not exist:
    •   return null;
4) get the base page (we need this to get base page url)
        RemotePage basePage = confluenceService.getPage(token, space, basePageName);
5) create resulting page
        Page result = new Page();
6) set resulting page properties according to remote page and method arguments:
        result.setContent(page.getContent());
        result.setVersionUrl(page.getUrl());
        result.setBasePageUrl(basePage.getUrl());
        result.setAssetName(pageName);
        result.setAssetType(assetType);
        …
```

1.4 Component Class Overview

package com.topcoder.confluence

ConfluenceManager [interface]:

This interface provides common operations to work with Confluence, such as login, create and retrieve pages using different settings, log out.

Thread-safety: implementations need to be thread-safe.

package com.topcoder.confluence.managerimpl

DefaultConfluenceManager:

This class represents the default implementation of ConfluenceManager interface. It provides operations to work with Confluence, such as login, create and retrieve pages using different settings, log out. This class can be configured via file-based runtime supplied configuration. It provides functionality to retry user calls in case of failure.

Thread-safety: this class is immutable, buts its field currentRetries is not final so changeable. Access to this field should be synchronized.

package com.topcoder.confluence.entities

Page:

This class represents the page created or retrieved from Confluence. It contains parameters that describe the page and also its content. When page is retrieved from Confluence, it will be transfromed in this object to provide more convenient access and relevant information to user. This class represents an entity so it should implement Serializable interface. Also this class should be serializable to XML for use with the Confluence Service component.

Thread-safety: this class is mutable so not thread-safe. Access to its state should be fully synchronized to ensure thread-safety. If instances of this class won't be changed during the processing, they can be used in thread-safe manner.

ConfluenceAssetType [enum]:

This enum represents the asset type, such as component design, component development, application specification, application architecture, application assembly, application testing. This class should be serializable to XML for use with the Confluence Service component.

Thread-safety: this is an enumeration whose state is immutable and hence thread-safe

ComponentType [enum]:

This enum represents the component type, that can be either custom or generic. This class should be serializable to XML for use with the Confluence Service component.

Thread-safety: this is an enumeration whose state is immutable and hence thread-safe

ConfluenceCatalog [enum]:

This enum represents the catalog, that can be either java or .net. This class should be serializable to XML for use with the Confluence Service component.

Thread-safety: this is an enumeration whose state is immutable and hence thread-safe

ConfluencePageType [enum]:

This enum represents the page type, that can be base/version page of the component, or base/version page of the application.

Thread-safety: this is an enumeration whose state is immutable and hence thread-safe

ConfluencePageCreatedAction [enum]:

This enum represents the action that occurred while creating/retrieving pages from Confluence. Possible actions are: base page and version page were both created, base page existed but version page was created, both base page and version page existed. This class should be serializable to XML for use with the Confluence Service component.

Thread-safety: this is an enumeration whose state is immutable and hence thread-safe

ConfluencePageCreationResult:

This class stores the info about outcome of creation of the page, and the page itself. This class represents an entity so it should implement Serializable interface. Also this class should be serializable to XML for use with the Confluence Service component.

Thread-safety: this class is immutable so thread-safe.

1.5 Component Exception Definitions

ConfluenceManagerException:

This exception shows the most common failure that occurred to confluence manager.

ConfluenceNotAuthorizedException:

This exception occurs when user tries to make call to Confluence service before the authorization was made.

ConfluenceConnectionException:

This exception shows problems with connection to Confluence service.

ConfluenceAuthenticationFailedException:

This exception occurs when user failed authentication.

1.6 Thread Safety

The entity Page is mutable so not thread-safe, but if this object won’t be changed while processing requests, it will be used in thread-safe manner. Other classes are immutable or are designed to be thread-safe, so the component is thread-safe.

2. Environment Requirements

2.1 Environment

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

2.2 TopCoder Software Components

- Configuration API 1.0
  this component allows to retrieve configuration values.
- Configuration Persistence 1.0.1
  this component is needed to support file-based configuration
- Base Exception 2.0
  this component provides base classes for custom exceptions
- Logging Wrapper 2.0
  this component provides the logging functionality

NOTE: The default location for TopCoder Software component jars is../lib/tcs/COMPONENT_NAME/COMPONENT_VERSION relative to the component installation. Setting the tcs_libdir property in topcoder_global.properties will overwrite this default location.

2.3 Third Party Components

- Axis2 : 1.4 : http://ws.apache.org/axis2/
- Confluence : 2.7 : http://confluence.atlassian.com/rpc/soap-glue/confluenceservice-v1.wsdl

NOTE: The default location for 3rd party packages is ../lib relative to this component installation. Setting the ext_libdir property in topcoder_global.properties will overwrite this default location.

3. Installation and Configuration

3.1 Package Name

com.topcoder.confluence
com.topcoder.confluence.impl
com.topcoder.confluence.entities

3.2 Configuration Parameters

Parameter	**Description**	**Values**
confluenceUrl	The url to Confluence service. *Required*	Valid url
logName	The name of the log. *Optional.* Defaults to the com.topcoder.confluence.managerimpl.DefaultConfluenceManager	String
maxRetriesNumber	The maximal number of retries that can be done in case if connection failure. *Optional*. Defaults to 1.	Integer
spaceLocationsMapping	The mapping of asset type to space key. *Required*	
templatesMapping	The mapping of page type to template file location. *Required*	

The table below describes format of *spaceLocationsMapping*

Parameter	**Description**	**Values**
componentDesign	The Confluence space key for component design asset. *Optional*. Defaults to http://www.topcoder.com/wiki/display/docs/Design	Valid url
componentDevelopment	The Confluence space key for component development asset. *Optional*. Defaults to http://www.topcoder.com/wiki/display/docs/Development	Valid url
applicationSpecification	The Confluence space key for component design asset. *Optional*. Defaults to	Valid url with $CODENAME$

	http://www.topcoder.com/wiki/display/projects/$CODENAME$	symbol that will be replaced.
applicationArchitecture	The Confluence space key for component design asset. *Optional*. Defaults to http://www.topcoder.com/wiki/display/docs/Architecture	Valid url
applicationAssembly	The Confluence space key for component design asset. *Optional*. Defaults to http://www.topcoder.com/wiki/display/docs/Assembly	Valid url
applicationTesting	The Confluence space key for component design asset. *Optional*. Defaults to http://www.topcoder.com/wiki/display/docs/Testing	Valid url

The table below describes format of *templatesMapping*

Parameter	**Description**	**Values**
componentBasePage	The template file location for creating component base page. *Required*.	Valid url
componentVersionPage	The template file location for creating component version page. *Required*.	Valid url
applicationBasePage	The template file location for creating application base page. *Required*.	Valid url
applicationVersionPage	The template file location for creating application version page. *Required*.	Valid url

3.3 Dependencies Configuration

None

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

Configure the component.

4.3 Demo

Assume that ConfigurationManager.properties has such entry:
com.topcoder.confluence.DefaultConfluenceManager=DefaultConfluenceManager.xml

Where DefaultConfluenceManager.xml has such data:

```
<?xml version="1.0"?>
<CMConfig>
   <Config name="com.topcoder.confluence.DefaultConfluenceManager">
      <property name="confluenceUrl">
         <value>http://confluence.atlassian.com/rpc/soap-axis/confluenceservice-v1</value>
      </property>
      <property name="logName">
         <value>com.topcoder.confluence.DefaultConfluenceManager</value>
```

```
    </property>
    <property name="maxRetriesNumber">
      <value>2</value>
    </property>
    <property name="spaceLocationsMapping">
      <property name="componentDesign">
        <value>http://www.topcoder.com/wiki/display/docs/Design</value>
      </property>
      <property name="componentDevelopment">
        <value>http://www.topcoder.com/wiki/display/docs/Development</value>
      </property>
      <property name="applicationSpecification">
        <value>http://www.topcoder.com/wiki/display/projects/$CODENAME$</value>
      </property>
      <property name="applicationArchitecture">
        <value>http://www.topcoder.com/wiki/display/docs/Architecture</value>
      </property>
      <property name="applicationAssembly">
        <value>http://www.topcoder.com/wiki/display/docs/Assembly</value>
      </property>
      <property name="applicationTesting">
        <value>http://www.topcoder.com/wiki/display/docs/Testing</value>
      </property>
    </property>
    <property name="templatesMapping">
      <property name="componentBasePage">
        <value>http://www.topcoder.com/wiki/display/docs/template1</value>
      </property>
      <property name="componentVersionPage">
        <value>http://www.topcoder.com/wiki/display/docs/template2</value>
      </property>
      <property name="applicationBasePage">
        <value>http://www.topcoder.com/wiki/display/docs/template3</value>
      </property>
      <property name="applicationVersionPage">
        <value>http://www.topcoder.com/wiki/display/docs/template4</value>
      </property>
    </property>
  </Config>
</CMConfig>
```

1) DefaultConfluenceManager instance can be created:

```
// manager will have settings from the file above, because it corresponds to default namespace.
ConfluenceManager manager = new DefaultConfluenceManager();
```

2) Log in and save authorization token:

```
String token = manager.login(“admin”, “password”);
```

3) Assume user wants to create such page: component design page for the new component ‘My New Component’, and the component is Java Custom.

```
// should pass token
ConfluencePageCreationResult page1result = manager.createPage(token, “My New Component”, “1”,
ConfluenceAssetType.COMPONENT_DESIGN, ConfluenceCatalog.JAVA, ComponentType.CUSTOM);

// if the component is new, the call page1result.getActionTaken() will return
//BASE_PAGE_AND_VERSION_CREATED.
```

4) Different properties of the created page can be get:

```
Page page1 = page1result.getPage();
String page1base = page1.getBasePageUrl();
String page1version = page1.getVersionUrl();
String content = page1.getContent();
```

5) Assume user wants to wants add new version of such page: application specification page for the application ‘My Favourite Application’ from the .net catalog.

```
// should pass token
// should pass application code
ConfluencePageCreationResult page2result = manager.createPage(token, “My Favourite Application”,
“2”, ConfluenceAssetType.APPLICATION_SPECIFICATION, ConfluenceCatalog.DOT_NET,
“randomAppCode”);

// page2result.getActionTaken() should return BASE_PAGE_EXISTED_VERSION_CREATED.
```

6) Assume user wants create the component development doc page with name ‘My Component X’ which will be .NET generic.

```
Page page3 = new Page();
page3.setAssetName(“My Component X”);
page3.setAssetType(ConfluenceAssetType.COMPONENT_DEVELOPMENT);
page3.setCatalog(ConfluenceCatalog.DOT_NET);
page3.setComponentType(ComponentType.GENERIC);


// save page
ConfluencePageCreationResult page3result  = manager.createPage(token, page3);
```

7) Now user wants to retrieve created page that he’s just created

```
Page page3copy = manager.retrievePage(token, “My Component X”, “1”,
ConfluenceAssetType.COMPONENT_DEVELOPMENT, ConfluenceCatalog.DOT_NET,
ComponentType.GENERIC);
```

8) Now user wants to retrieve the application specification for the ‘My Favourite Application’

```
Page page2copy = manager.retrievePage(token, “My Favourite Application”, “2”,
ConfluenceAssetType.APPLICATION_SPECIFICATION, ConfluenceCatalog.DOT_NET,
“randomAppCode”);
```

9) Log out from Confluence

```
manager.logout(token);
```

5. Future Enhancements

Add support for more operations with Confluence service.