

# Online Review Login Component Specification

## 1. Design

The component provides the login and logout action support for the Online Review application. Users can pass the user name and password for authentication. In order to make the component as flexible as possible, this component leverages the TopCoder Authenticator Factory framework to make the authentication pluggable. Before integrating the authenticator, application users must implement AuthResponseParser interface to custom their own "login" and "logout" function. In the current version, SecurityManagerAuthenticator is implemented to authenticate the user name and password by employing the LoginBean from the Security Manager component.

### 1.1 Design Patterns

The LoginActionServlet and LoginActions follow the MVC design pattern of struts.

AuthResponseParser implements the Strategy pattern.

### 1.2 Industry Standards

Struts

EJB

### 1.3 Required Algorithms

#### 1.3.1 *Forward to pages*

LoginActions in this component is responsible for forwarding the request to target pages based on the authentication response or user action.

If authentication succeeded, users will be forwarded to the "success" page.

If failed, users will be forwarded to the "failure" page.

If users logged out, they will be forwarded to the "logout" page.

Following is the forward rules specified in the struts-config.xml:

```
<action-mappings>

<action path="/login" unknown="false"
    type="com.cronos.onlinereview.login.LoginActions"

    validate="true" name="loginForm" scope="session" parameter="method"
    input="login.jsp">

    <exception
        type="com.topcoder.security.authenticationfactory.AuthenticateException"
        key="exception.com.cronos.onlinereview.login.LoginActions.login.Authentic
        ateException" path="/login.jsp" />

    <exception type="
        com.topcoder.security.authenticationfactory.MissingPrincipalKeyException"
        key="exception.com.cronos.onlinereview.login.LoginActions.login.MissingPr
        incipalException" path="/login.jsp" />

    <exception type="
```

```

        com.topcoder.security.authenticationfactory.InvalidPrincipalException"
        key="exception.com.cronos.onlinereview.login.LoginActions.login.InvalidPr
        incipalException" path="/login.jsp" />

<exception
    type="com.cronos.onlinereview.login.
    LoginActions.AuthResponseParserException"
    key="exception.com.cronos.onlinereview.login.LoginActions.login.AuthRespo
    nseParserException" path="/login.jsp" />

    <forward name="success" path="/projectlist.jsp" redirect="false"/>

    <forward name="failure" path="/login.jsp" redirect="false"/>
</action>

<action path="/logout" unknown="false"
    type="com.cronos.onlinereview.login.LoginActions"

    validate="false" scope="session" parameter="method" input="login.jsp">

    <exception
        type="com.cronos.onlinereview.login.
        LoginActions.AuthResponseParserException"
        key="exception.com.cronos.onlinereview.login.LoginActions.logout.AuthResp
        onseParserException" path="/login.jsp" />

        <forward name="logout" path="/login.jsp" redirect="false"/>
</action>
</action-mappings>

```

### 1.3.2 *Dynamical validation of form fields*

This component uses DynaValidatorForm to validate the form fields. The form field is specified in the form bean element of struts-config:

```

<form-beans>

    <form-bean name="loginForm" dynamic="true"
        type="org.apache.struts.validator.DynaValidatorForm">

        <form-property name="userName" type="java.lang.String"/>

        <form-property name="password" type="java.lang.String"/>

    </form-bean>
</form-beans>

```

We can use struts validation framework to specify the validation rules: Following is part of validation.xml:

```

<formset>

<form name="loginForm">

    <field property="userName" depends="required ">

<msg name="required" key="error.com.cronos.onlinereview.login.username"/>

```

```

        </field>

        <field property="password" depends="required">

        <msg name="required" key="error.com.cronos.onlinereview.login.password"/>

        </field>

    </form>

</formset>

```

### 1.3.3 *Resource bundle for the error messages by LoginActions*

Following is the resource message keys for the above two algorithms:

```

exception.com.cronos.onlinereview.login.LoginActions.login.AuthenticateException=U
nexpected error occurred during authentication

exception.com.cronos.onlinereview.login.LoginActions.login.InvalidPrincipalExcepti
on= Incorrect user name or password

exception.com.cronos.onlinereview.login.LoginActions.login.MissingPrincipalKeyExce
ption= User name or password required

exception.com.cronos.onlinereview.login.LoginActions.login.AuthResponseParserExcep
tion= Unexpected error occurred during authentication

exception.com.cronos.onlinereview.login.LoginActions.logout.AuthResponseParserExce
ption= Unexpected error occurred

error.com.cronos.onlinereview.login.password=Password required

error.com.cronos.onlinereview.login.username=User name required

```

## 1.4 Component Class Overview

### **LoginActions:**

LoginActions defines login and logout actions for this component.

Login action is responsible for logging in the user to the application,

Logout action is responsible for logging out the user.

If any exception occurs in both action methods, resource key should be added to the ActionMessage.

Since all the exceptions will be caught in both methods, Logger is employed to log any caught exception. Note that log mechanism is optional.

This class is thread safe.

### **SecurityManagerAuthenticator:**

This class is used to authenticate user. The authentication work will be delegated to the LoginBean. The login bean will be retrieved from the JNDI Context by the bean name.

This class is thread safe since it does not contain any mutable inner state.

### **AuthResponseParser**

This interface defines the contract of keeping the track of the log state stored in the request or session. The setLoginState method is used to set the user log state (failed or succeeded). The unsetLoginState method is used to unset the user log state.

In order to integrate other authenticator implementation into this component, application user must also provide the corresponding implementation of this interface to manage the log states.

The implementation of this interface is required to be thread safe.

### **SecurityManagerAuthResponseParser:**

This class is used to manage the log state in request for the SecurityManagerAuthenticator, so it's expected that Response#getDetails() should return com.topcoder.security.TCSubject.

In the setLoginState method, if authentication succeeded, it will store in the user identifier in the session identifier.

In the unsetLoginState method, it will remove the user identifier from the session and invalidate the session.

This class is thread safe since it does not contain any mutable state.

## **1.5 Component Exception Definitions**

### **IllegalArgumentException**

This exception is thrown in various methods if null object is not allowed, or the given string argument is empty. Refer to the documentation in Poseidon for more details.

**NOTE: Empty string means string of zero length or string full of whitespaces.**

### **javax.servlet.ServletException**

It is thrown from LoginActionServlet.

### **java.io.IOException**

It is thrown from LoginActionServlet.

### **ConfigurationException**

This exception is thrown from the constructor taking a namespace argument, if failed to load configuration values from the ConfigManager or the configuration values are invalid.

### **AuthResposenParsingException**

This exception is thrown by AuthResponseParser interface and its implementation if any error occurred when they check and set the http request and response.

### **AuthenticateException**

This exception will be thrown by SecurityManagerAuthenticator

### **InvalidPrincipalException**

This exception will be thrown by SecurityManagerAuthenticator

## **InvalidPrincipalException**

This exception will be thrown by SecurityManagerAuthenticator

## **1.6 Thread Safety**

The servlet engine creates single LoginActionServlet and single LoginActions instance to handle all users' actions, so they must be thread-safe. LoginActionServlet is thread safe, since all its referred components are thread safe. LoginActions is also stateless to ensure the thread-safety.

SecurityManagerAuthenticator is thread safe since it does not contain any mutable inner stat. The authenticate method of the authenticator simply uses the bean to authentication user name and password, both of which will not be modified during authentication, though user might change the password after login.

AuthResponseParser and it's implementation are required to be thread safe. In the current version, SecurityManagerResonseParser is thread safe since it also does not contain any mutable inner state.

## **2. Environment Requirements**

### **2.1 Environment**

Java 1.4 or higher.

### **2.2 TopCoder Software Components**

#### **Configuration Manager 2.1.4**

It is used to load configuration values.

#### **Logging Wrapper 1.2**

It's used to log the exception stack trace.

#### **JNDI Utility 1.0**

It's used to look up the login bean.

#### **Authentication Factory 2.0**

It is used to authenticate the user.

#### **Security Manager 1.1**

It is used by SecurityManagerAuthenticator to authenticate user.

#### **Base Exception 1.0**

Custom exceptions extend the BaseException.

### **2.3 Third Party Components**

Struts 1.2.9.

### 3. Installation and Configuration

#### 3.1 Package Name

com.cronos.onlinereview.login

com.cronos.onlinereview.login.authenticator

#### 3.2 Configuration Parameters

Configuration values for LoginActions

Parameter	Description	Values
<b>auth_response_parser.class</b>	The full qualified class name of the authentication response parser. <b>Required.</b>	com.cronos.onlinereview.login.authenticator.SecurityManagerAuthResponseParser
<b>auth_response_parser.namespace</b>	The configuration namespace used by authentication response parser to load the configuration values. <b>Required.</b>	com.cronos.onlinereview.login.authenticator.SecurityManagerAuthResponseParser
<b>authenticator_name</b>	The authenticator name used to get authenticator from AuthenticatorFactory, <b>Required.</b>	SecurityManagerAuthenticator
<b>logger_name</b>	The logger name used to get log from LogFactory. <b>Optional.</b>	loginActionsLoggerName

Configuration values for SecurityManagerAuthenticator

Parameter	Description	Values
<b>context_name</b>	The context name used to get the Context from JNDIUtils. <b>Optional.</b>	someContextName
<b>login_bean_name</b>	The login bean name used to look up the login bean in the context. <b>Required.</b>	loginBeanName

Configuration values for SecurityManagerAuthResponseParser

Parameter	Description	Values
<b>user_idenfifer_key</b>	The user identifier key name. <b>Required.</b>	user_identifier_key

### 3.3 Dependencies Configuration

ConfigurationManager, LogFactory and Authenticator Factory should be properly configured to make this component work.

## 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

Preload the configuration file into Configuration Manager. Follow demo.

### 4.3 Demo

Assume the configuration file in 3.2 is used here. And follow the algorithm section about how to configure the struts-config.xml and resource message file

#### 4.3.1 Log in

```
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean"%>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html"%>
<%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<html:errors />
<br>
<form action="/LoginPage/login"><input type="text" name="method" value="login">
  <input type="text" name="userName"> <br>
  <input type="text" name="password"> <br>
  <input type="submit" name="login" value="login"></form>
<br>
</body>
</html>
```

#### 4.3.2 Log out

```
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean"%>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html"%>
<%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<html:errors />
<br>
<br>
<br>
<a href="/LoginPage/logout?method=logout">Logout</a>
</body>
</html>
```

## **5. Future Enhancements**

More existing authenticator implementations (like LDAP authenticator) could be integrated into this component.