

# Using CAS without the Login Screen

## Motivation

We wanted to be more flexible in the use of the login UI, so e.g. wanted to embed it in several places as a small panel. Moreover, we wanted to understand CAS as a pure service, not having to maintain layout information twice. Also, we wanted to support both, login postings from another site and direct login at the CAS server as a fallback.

## Other proposed solutions

- 'Using CAS from external link or custom external form' page
- 'Using CAS without the CAS login screen' page
- A google group discussion

## Proposed Solutions that do not work

There have been two proposals, which we haven't found to be suitable:

- AJAX: Ajax requests are sandboxed and that sandbox is even more strict than the two-dot rule applied to cookies: It really has to be exactly the same server name in order to work
- IFrames: In this [posting to the cas-dev mailing list](#), it has been suggested to use an iFrame to embed a login panel on a remote site. This has two limitations:
  - How to get rid of the Frame after login? in the default view a successful login would lead to redirect only in that frame. If you use target="parent" in the from tag, how do you display errors then?
  - You would have limited layout possibilities.

So for our requirements, none of the proposed solutions were adequate.

## Our Solution

We decided to use Javascript triggered redirects, which allows us to use this mechanism also on static html pages (e.g. CMS based contents). While loading we request a login ticket, then the login form displays and then it submits to the CAS login URL. If Errors are present at CAS we redirect to the referring page, adding a parameter "error\_message" to that request.

## Client side

Here is the static remote login form we use:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Test remote Login using JS</title>
<script type="text/javascript">
function prepareLoginForm() {
    $('myLoginForm').action = casLoginURL;
    $("lt").value = loginTicket;
}

function checkForLoginTicket() {
    var loginTicketProvided = false;
    var query = '';
    casLoginURL = 'https://pcdt057:8443/cas/login';
    thisPageURL = 'https://pcdt057:8443/cas/test-remote-login-
js+error.html';
```

```

        casLoginURL += '?login-at=' + encodeURIComponent (thisPageURL);

query      = window.location.search;
query      = query.substr (1);

var param   = new Array();
//var value = new Array();
var temp    = new Array();
param      = query.split ('&');

i = 0;
while (param[i]) {
    temp      = param[i].split ('=');
    if (temp[0] == 'lt') {
        loginTicket = temp[1];
        loginTicketProvided = true;
    }
    if (temp[0] == 'error_message') {
        error = temp[1];
    }
    i++;
}
if (!loginTicketProvided) {
    location.href = casLoginURL + '&get-lt=true';
}
}

function $(id) {
    return document.getElementById(id);
}
var loginTicket;
var error;
var casLoginURL;
var thisPageURL;

checkForLoginTicket();
onload = prepareLoginForm;
</script>
</head>
<body>
<h2>Test remote Login using JS</h2>
<form id="myLoginForm" action="" method="post">
<input type="hidden" value="submit" name="_eventId"/>
<table>
<tr>
    <td id="txt_error" colspan="2">

    <script type="text/javascript" language="javascript">
    <!--
    if ( error ) {
        error = decodeURIComponent (error);
        document.write (error);
    }
    </script>

```

```

    }
    //-->
</script>

</td>
</tr>
<tr>
    <td>Benutzer:</td>
    <td><input type="text" value="konrad" name="username" ></td>
</tr>
<tr>
    <td>Password:</td>
    <td><input type="text" value="konrad" name="password" ></td>
</tr>
<tr>
    <td>Login Ticket:</td>
    <td><input type="text" name="lt" id="lt" value=""></td>
</tr>
<tr>
    <td>Service:</td>
    <td><input type="text" name="service" value="http://quiz.jacomac.
de"></td>
</tr>
<tr>
    <td align="right" colspan="2"><input type="submit" /></td>
</tr>
</table>
</form>
</body>
</html>

```

## CAS side

Adapting only the views and introducing if-else statements there meant having logic that belongs to the controllers in the view. Therefore we decided to inject our add-on into the spring web flow. Here are the changes we have made to login-webflow.xml (changes highlighted in blue):

```

<?xml version="1.0" encoding="UTF-8"?>
<flow xmlns="http://www.springframework.org/schema/webflow"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
        http://www.springframework.org/schema/webflow
        http://www.springframework.org/schema/webflow/spring-webflow-
1.0.xsd">

```

```
<start-state idref="provideLoginTicketToRemoteRequestor" />
```

```
<action-state id="provideLoginTicketToRemoteRequestor">
```

```
<action bean="provideLoginTicketToRemoteRequestorAction" />
```

```
<transition on="loginTicketRequested" to="viewRedirectToRequestor" />
```

```
<transition on="continue" to="automaticCookiePathSetter" />
</action-state>
```

```
<view-state id="viewRedirectToRequestor" view="bmRedirectToRequestorView">
<transition on="submit" to="bindAndValidate" />
</view-state>
```

```
<action-state id="automaticCookiePathSetter">
    <action bean="automaticCookiePathSetterAction" />
    <transition on="success"
        to="ticketGrantingTicketExistsCheckAction" />
</action-state>

<action-state id="ticketGrantingTicketExistsCheckAction">
    <action bean="ticketGrantingTicketExistsAction" />
    <transition on="ticketGrantingTicketExists"
        to="hasServiceCheck" />
    <transition on="noTicketGrantingTicketExists"
        to="gatewayRequestCheck" />
</action-state>

<action-state id="gatewayRequestCheck">
    <action bean="gatewayRequestCheckAction" />
    <transition on="gateway" to="redirect" />
    <transition on="authenticationRequired" to="viewLoginForm" />
</action-state>

<action-state id="hasServiceCheck">
    <action bean="hasServiceCheckAction" />
    <transition on="authenticatedButNoService"
        to="viewGenericLoginSuccess" />
    <transition on="hasService" to="renewRequestCheck" />
</action-state>

<action-state id="renewRequestCheck">
    <action bean="renewRequestCheckAction" />
    <transition on="authenticationRequired" to="viewLoginForm" />
    <transition on="generateServiceTicket"
        to="generateServiceTicket" />
</action-state>

<!--
    <action-state id="startAuthenticate">
        <action bean="x509Check" />
        <transition on="success" to="sendTicketGrantingTicket" />
        <transition on="error" to="viewLoginForm" />
    </action-state>
-->
<view-state id="viewLoginForm" view="casLoginView">
    <transition on="submit" to="bindAndValidate" />
</view-state>

<action-state id="bindAndValidate">
    <action bean="authenticationViaFormAction" />
```

```

    <transition on="success" to="submit" />
    <transition on="error" to="viewLoginForm" />
</action-state>

<action-state id="submit">
    <action bean="authenticationViaFormAction" method="submit" />
    <transition on="warn" to="warn" />
    <transition on="success" to="sendTicketGrantingTicket" />
    <transition on="error" to="viewLoginForm" />

```

```
<transition on="errorForRemoteRequestor" to="viewRedirectToRequestor" />
```

```

</action-state>

    <action-state id="sendTicketGrantingTicket">
        <action bean="sendTicketGrantingTicketAction" />
        <transition on="success" to="serviceCheck" />
    </action-state>

    <action-state id="serviceCheck">
        <action bean="hasServiceCheckAction" />
        <transition on="authenticatedButNoService"
            to="viewGenericLoginSuccess" />
        <transition on="hasService" to="generateServiceTicket" />
    </action-state>

    <action-state id="generateServiceTicket">
        <action bean="generateServiceTicketAction" />
        <transition on="success" to="warn" />
        <transition on="error" to="viewLoginForm" />
        <transition on="gateway" to="redirect" />
    </action-state>

    <!--
    The "warn" action makes the determination of whether to redirect
    directly to the requested
    service or display the "confirmation" page to go back to the
    server.
    -->
    <action-state id="warn">
        <action bean="warnAction" />
        <transition on="redirect" to="redirect" />
        <transition on="warn" to="showWarningView" />
    </action-state>

    <!--
    the "viewGenericLogin" is the end state for when a user attempts
    to login without coming directly from a service.
    They have only initialized their single-sign on session.
    -->

```

```

<end-state id="viewGenericLoginSuccess"
  view="casLoginGenericSuccessView" />

<!--
  The "showWarningView" end state is the end state for when the user
  has requested privacy settings (to be "warned") to be turned on. It
  delegates to a
    view defines in default_views.properties that display the "Please
  click here to go to the service." message.
-->
<end-state id="showWarningView" view="casLoginConfirmView" />

<!--
  The "redirect" end state allows CAS to properly end the workflow
  while still redirecting
    the user back to the service required.
-->
<end-state id="redirect"
  view="externalRedirect:${externalContext.requestParameterMap
['service']}${requestScope.ticket == null ? '' : (externalContext.
requestParameterMap['service'].indexOf('?') != -1 ? '&' : '?') +
'ticket=' + requestScope.ticket}" />

<global-transitions>
  <transition to="viewServiceErrorView"
    on-exception="org.jasig.cas.services.
UnauthorizedServiceException" />
</global-transitions>
</flow>

```

## Problems encountered and suggested solutions

The solution works well, but it has a very dirty edge: introducing a new event in the submit method that allows us to distinguish between a login posting from the CAS Site and another site. Here we had to redeclare the `submit` method in `org.jasig.cas.web.flow.AuthenticationViaFormAction` (changes marked in blue):

```
[...]      try {
            final String ticketGrantingTicketId = this.
centralAuthenticationService
                .createTicketGrantingTicket(credentials);
            ContextUtils.addAttribute(context,
                AbstractLoginAction.
REQUEST_ATTRIBUTE_TICKET_GRANTING_TICKET,
                ticketGrantingTicketId);
            setWarningCookie(response, warn);
            return success();
        } catch (final TicketException e) {
            populateErrorsInstance(context, e);
        }
```

```
// START: ChangesBusinessMart: check, whether the posting has been sent from a remote server
String myServerName = request.getLocalName();
String referrer = request.getParameter("login-at");
if (referrer != null && referrer.indexOf(myServerName) == -1) {
    return result("errorForRemoteRequestor");
}
```

```
return error();
}
[...]
```

This could have been avoided by

1. making this submit method overridable (non-final) to add new events in a derived class. Or
2. to make such a distinction of events in the standard CAS distribution.

Maybe this can be accomplished somehow in future versions of CAS?

## Listing of the other files mentioned in this solution

provideLoginTicketToRemoteRequestorAction -> introduces a new parameter get-It to signal that a new login ticket shall be issued to the HTTP Referer:

```

package de.businessmart.sso.cas.flow;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.jasig.cas.web.flow.util.ContextUtils;
import org.springframework.beans.factory.InitializingBean;
import org.springframework.webflow.action.AbstractAction;
import org.springframework.webflow.core.collection.MutableAttributeMap;
import org.springframework.webflow.execution.Event;
import org.springframework.webflow.execution.RequestContext;

import de.businessmart.sso.cas.CasUtility;

/**
 * Opens up the CAS web flow to allow external retrieval of a login ticket.
 * @author konrad.wulf
 *
 */
public class ProvideLoginTicketToRemoteRequestorAction extends
AbstractAction
{

    @Override
    protected Event doExecute(RequestContext context) throws Exception
    {
        final HttpServletRequest request = ContextUtils.
getHttpServletRequest(context);

        if (request.getParameter("get-lt") != null && request.
getParameter("get-lt").equalsIgnoreCase("true")) {
            return result("loginTicketRequested");
        }
        return result("continue");
    }
}

```

[viewRedirectToRequestor](#) actually does the redirects to the referrer:



```

<%@page import="de.businessmart.sso.cas.CasUtility"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>
<% String separator = "";
String referrer = request.getHeader("Referer");
referrer = CasUtility.resetUrl(referrer);
if (referrer != null && referrer.length() > 0) {
    separator =(referrer.indexOf("?") > -1)? "&" : "?"; %>
<html>
    <head>
        <script>
            var redirectURL = "<%= referrer + separator %
>lt=${flowExecutionKey}";
            <spring:hasBindErrors name="credentials">
                redirectURL += '&error_message=' + encodeURIComponent ('<c:
forEach var="error" items="{errors.allErrors}"><spring:message
code="{error.code}" text="{error.defaultMessage}" /></c:forEach>');
            </spring:hasBindErrors>
            window.location.href = redirectURL;
        </script>
    </head>
    <body></body>
</html><%
}
else
{
    out.print("You better know what to do here.");
} %>

```

And for completeness, here is the CasUtility that removes obsolete parameters from the query string:

```

package de.businessmart.sso.cas;

public class CasUtility {
    /**
     * Removes the previously attached GET parameters "lt" and
     "error_message" to be able to send new ones.
     * @param casUrl
     * @return
     */
    public static String resetUrl ( String casUrl) {
        String cleanedUrl;
        String[] paramsToBeRemoved = new String[]{"lt", "error_message",
"get-lt"};
        cleanedUrl = removeHttpGetParameters(casUrl, paramsToBeRemoved);
        return cleanedUrl;
    }
}

```

```

/**
 * Removes selected HTTP GET parameters from a given URL
 * @param casUrl
 * @param paramsToBeRemoved
 * @return
 */
public static String removeHttpGetParameters (String casUrl, String[]
paramsToBeRemoved) {
    String cleanedUrl = casUrl;
    if (casUrl != null)
    {
        // check if there is any query string at all
        if (casUrl.indexOf("?") == -1)
        {
            return casUrl;
        } else
        {
            // determine the start and end position of the parameters
            to be removed

            int startPosition, endPosition;
            boolean containsOneOfTheUnwantedParams = false;
            for (String paramToBeErased : paramsToBeRemoved)
            {
                startPosition = -1;
                endPosition = -1;
                if (cleanedUrl.indexOf("? " + paramToBeErased + "=") >
-1)
                {
                    startPosition = cleanedUrl.indexOf("? "
                        + paramToBeErased + "=") + 1;
                } else if (cleanedUrl.indexOf("& " + paramToBeErased +
"=") > -1)
                {
                    startPosition = cleanedUrl.indexOf("& "
                        + paramToBeErased + "=") + 1;
                }
                if (startPosition > -1)
                {
                    int temp = cleanedUrl.indexOf("&", startPosition);
                    endPosition = (temp > -1) ? temp + 1 : cleanedUrl
                        .length();
                    // remove that parameter, leaving the rest
                    untouched

                    cleanedUrl = cleanedUrl.substring(0, startPosition)
                        + cleanedUrl.substring(endPosition);
                    containsOneOfTheUnwantedParams = true;
                }
            }

            // wenn nur noch das Fragezeichen vom query string übrig
            oder am schluss ein "&", dann auch dieses entfernen
            if (cleanedUrl.endsWith("?") || cleanedUrl.endsWith("&"))
            {

```

```
        cleanedUrl = cleanedUrl.substring(0,
            cleanedUrl.length() - 1);
    }

    // erst zurückgeben, wenn wir auch überprüft haben, ob
    nicht ein parameter mehrfach angegeben wurde...
    if (!containsOneOfTheUnwantedParams)
        return casUrl;
    else
        cleanedUrl = removeHttpGetParameters(cleanedUrl,
paramsToBeRemoved);
    }
}
return cleanedUrl;
}
}
```