



Documentation CAS à destination des éditeurs

Sommaire

Préambule.....	1
Présentation de CAS.....	2
Intérêt.....	2
Fonctionnement de base.....	2
Synoptique des échanges (1ère connexion).....	2
Synoptique des échanges (2ème connexion).....	3
Ticket-Granting Cookie (TGC)	3
Service Ticket (ST).....	3
Fonctionnement en mode proxy.....	4
Synoptique des échanges (obtention du PGT).....	4
Synoptique des échanges (accès à une application tierce).....	5
Proxy-Granting-Ticket (PGT).....	5
Proxy-Ticket (PT).....	5
Implémentation côté éditeur.....	6
Mise en place.....	6
Création d'une page d'accès à la ressource.....	6
Informations techniques.....	6
Document LOMFR de description de ressource.....	7
Test.....	7
Annexes.....	8
Retour de validation CAS.....	8
Authentification réussie.....	8
Authentification échouée.....	8
Page d'accès exemple en PHP.....	9
Notes.....	10
Page d'accès exemple en Java.....	11
Notes.....	12
Page d'accès exemple en ASP.....	13

Préambule

Ce document présente succinctement le système d'authentification unique (SSO¹) utilisé dans le projet CORRELYCE : CAS et particulièrement son utilisation pour la connexion aux ressources éditoriales en ligne.

Un SSO est un système qui permet de centraliser l'authentification au sein d'applications connexes. Ceci a plusieurs avantages :

- simplifier la gestion des mots de passe : c'est le même login / mot de passe qui sert pour plusieurs applications
- gagner du temps lors de l'utilisation de plusieurs applications successivement : un utilisateur

¹ Single Sign On : Authentification Unique

connecté sur une application peut passer à une autre (s'il y est autorisé bien sûr) sans avoir à se ré authentifier

- simplifier la conception des applications en déportant la gestion de l'authentification vers un entrepôt central (utilisant un annuaire LDAP ou une base de données par exemple)

Présentation de CAS

Note : cette présentation est fortement inspirée de l'article CAS² sur Wikipedia.

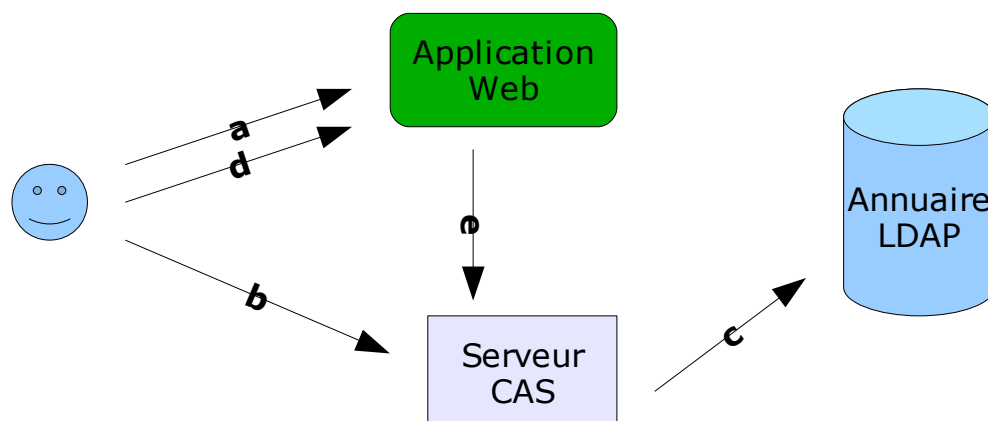
CAS³ est un système d'authentification unique développé par l'Université de Yale. C'est un mécanisme très solide, qui est implanté dans plusieurs universités et organismes dans le monde. CAS est une application Web écrite en Java et distribuée comme un logiciel libre.

Intérêt

Il évite de s'authentifier à chaque fois qu'on accède à une application en mettant en place un système de ticket. CAS est un système de SSO : on s'authentifie sur un site web, et on est alors authentifié sur tous les sites web qui utilisent le même serveur CAS.

Fonctionnement de base

Synoptique des échanges (1ère connexion)



Dans son fonctionnement de base, une application CASifiée⁴ s'en remet à CAS pour son authentification. I.e. lors de l'accès d'un utilisateur anonyme à une page protégée :

1. L'utilisateur accède à l'application Web (a)
2. Il est redirigé vers le formulaire de login de CAS (b)
3. L'utilisateur s'authentifie (le serveur cas vérifie l'identité fournie auprès de l'annuaire LDAP (c))
4. Si l'authentification échoue (mauvais couple login/mot de passe), une page d'erreur CAS est affichée
5. Si l'authentification réussit, CAS renvoie un « Ticket Granting Cookie » (TGC) à l'utilisateur et le redirige vers l'application initiale avec un « Service Ticket » (ST) en paramètre (d)

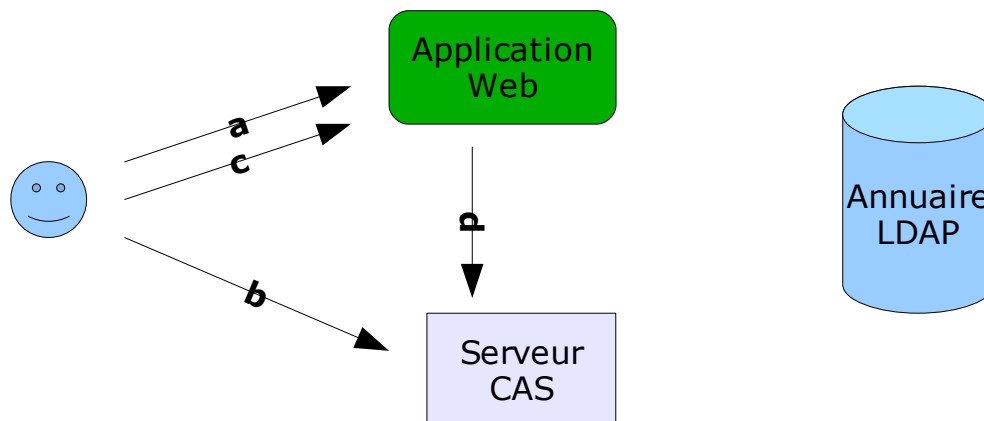
² http://fr.wikipedia.org/wiki/Central_Authentication_Service

³ Central Authentication Service : <http://www.ja-sig.org/products/cas/index.html>

⁴ une application CASifiée est une application qui se connecte à un serveur CAS pour son identification

- L'application valide le ST reçu auprès du serveur CAS (e) et reçoit en échange l'information de login sur l'utilisateur

Synoptique des échanges (2ème connexion)



Dès lors, l'utilisateur est authentifié auprès du serveur CAS, s'il cherche à accéder à une page protégée d'une autre application connecté au même serveur CAS :

- L'utilisateur accède à l'application Web (a)
- Il est redirigé vers le serveur CAS (b)
- Celui-ci lit le cookie TGC fourni et redirige l'utilisateur vers l'application Web avec un nouveau ticket ST en paramètre (c)
- L'application valide le ST reçu auprès du serveur CAS (d) et reçoit en échange l'information de login sur l'utilisateur

Note : l'utilisateur n'a pas eu à se réauthentifier (c'est tout l'intérêt du SSO).

Ce fonctionnement de base est utilisé dans Correlyce et permet, par exemple, à un utilisateur connecté à l'application Correlyce de voir ses informations de connexion lorsqu'il bascule sur l'application SPIP actus.

Retour sur les notions de TGC et ST :

Ticket-Granting Cookie (TGC)

C'est un cookie de session qui est transmis par le serveur CAS au navigateur du client lors de la phase de login. Ce cookie ne peut être lu / écrit que par le serveur CAS, sur canal sécurisé (https). Ce cookie est présent durant toute la session de l'utilisateur et permet d'obtenir des Service Ticket auprès du serveur CAS.

Si le navigateur web n'accepte pas les cookies, l'utilisateur devra se ré-authentifier à chaque appel au serveur CAS.

Service Ticket (ST)

Ce ticket va servir à authentifier une personne pour une application web donnée. Il est envoyé par le serveur CAS après que l'utilisateur s'est authentifié et est transporté dans l'URL.

Ce ticket ne peut être utilisé qu'une seule fois. Il y a ensuite dialogue direct entre l'application web

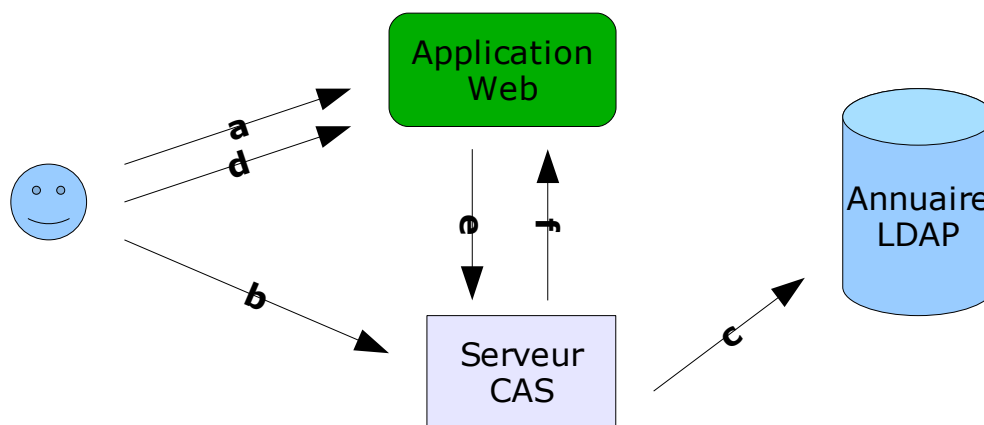
et le CAS via un GET http, avec le ST en paramètre. En réponse, le serveur CAS retourne l'identifiant de la personne, et donc l'authentifie. Il invalide également le ticket (libération des ressources associées).

En fait, ce ticket concerne une personne, pour un service, et utilisable une seule fois.

Fonctionnement en mode proxy

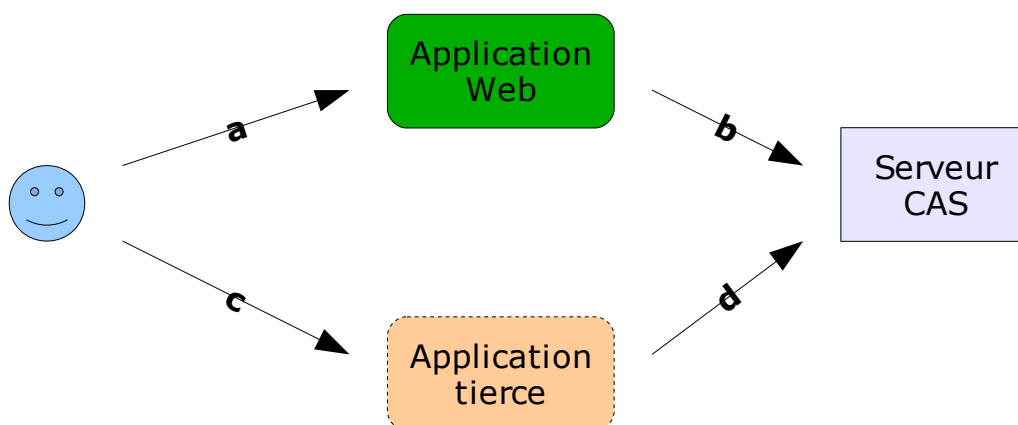
Le fonctionnement en mode proxy permet à une application proxy CAS de donner accès à d'autres applications clientes sous condition. Ces applications clientes valideront le ticket transmis par l'application proxy auprès du serveur CAS.

Synoptique des échanges (obtention du PGT)



1. Un utilisateur se connecte à une application Web proxy (a)
2. Il est redirigé vers le formulaire de login de CAS (b)
3. le serveur CAS vérifie son identité auprès de l'annuaire LDAP
4. L'utilisateur récupère un cookie TGC et est redirigé vers l'application Web avec un ST en paramètre (d)
5. L'application Web valide le ST et demande un « Proxy Granting Ticket » (PGT) (phase e)
6. Le serveur CAS rappelle l'application Web de manière asynchrone (f) pour lui fournir le PGT demandé pour l'utilisateur

Synoptique des échanges (accès à une application tierce)



L'utilisateur souhaite accéder à une application cliente via l'application proxy

1. L'utilisateur accède à l'application Web proxy (a) pour demander l'accès à l'application tierce (via un lien sur une page par exemple)
2. l'application proxy demande un « Proxy Ticket » (PT) auprès du serveur CAS pour l'application tierce (b)
3. l'application proxy redirige l'utilisateur vers l'application tierce avec le ticket PT en paramètre (c)
4. l'application tierce valide le ticket reçu auprès du serveur CAS pour obtenir des informations de connexion sur l'utilisateur (d)

Notes :

- c'est l'application proxy qui est chargée de faire tous les tests nécessaires pour autoriser ou non l'accès à l'application cliente demandée (i.e. générer ou non le PT nécessaire)
- l'application cliente ne peut être accédée par l'utilisateur qu'en passant par l'application proxy

Dans Correlyce, le fonctionnement en mode proxy est utilisé pour donner accès aux ressources des éditeurs, cela permet de vérifier si l'utilisateur qui demande la ressource y a bien droit (abonnement) avant de générer le ticket nécessaire.

Proxy-Granting-Ticket (PGT)

Il est envoyé par le serveur CAS à une application web 'proxy CAS' disposant d'un ST valide. Ce ticket confère au proxy CAS la possibilité de demander au serveur CAS de générer un Proxy Ticket (PT) pour une application tierce et une personne donnée.

Proxy-Ticket (PT)

Il est généré par le serveur CAS à la demande d'un proxy CAS. Il permet d'authentifier l'utilisateur pour un service distant, avec lequel le client web n'a pas d'accès direct. Le service distant l'utilisera comme le ST⁵.

Le PT est lié au service distant et n'est pas rejouable.

⁵ mais sur une URL différente. CAS dispose de deux services distincts pour la validation de ses tickets : serviceValidate (pour les ST) et proxyValidate (pour les PT)

Implémentation côté éditeur

L'utilisation de CAS est basé sur des standards de l'internet : HTTP(s) et XML et ne requiert pas de langage de programmation particulier.

Pour pouvoir prendre en compte les demandes d'accès provenant de Correlyce, deux opérations sont nécessaires :

1. Mettre en place une page d'accès à la ressource
2. Indiquer l'adresse de cette page dans le document LOMFR correspondant

Mise en place

Création d'une page d'accès à la ressource

La page d'accès à la ressource doit être une page « dynamique » (dans un langage tel que Java, PHP, ASP, ...) et doit répondre selon le protocole HTTPS.

Actions à réaliser pour vérifier la validité de l'appelant :

1. vérifier qu'un paramètre nommé « ticket » est bien passé en paramètre, sinon afficher un message d'erreur
2. appeler la page <https://cas.correlyce.fr/cas/proxyValidate> avec 2 paramètres :
 1. « ticket » : le paramètre ticket passé en paramètre URL
 2. « service » : l'URL de la page courante⁶
3. analyser la réponse XML du serveur CAS :
 1. erreur d'authentification : afficher une erreur à l'utilisateur
 2. authentification réussie : extraire les informations d'identifiant numérique d'utilisateur, sa classe, son rôle, le siren et le rne de son établissement et donner accès à la ressource

Notes :

- Les bibliothèques clientes CAS disponibles⁷ dans de nombreux langages de programmation cachent ces échanges clients-serveurs. Il suffit généralement de leur fournir l'adresse du serveur CAS (<https://cas.correlyce.fr>) et d'indiquer que la page est un client proxy CAS.
- Un éditeur / diffuseur qui propose plusieurs ressources à Correlyce n'est pas obligé de mettre en place une page d'accès par ressource mais peut proposer un point d'entrée unique avec un paramètre URL différent par ressource (paramètre « idressource » par exemple qui contient un identifiant unique de ressource dans le système de l'éditeur)

Informations techniques

- Le serveur CAS de correlyce est à l'adresse <https://cas.correlyce.fr>.
- La page d'accès à la ressource doit être disponible en https.
- Les tags XML à extraire de la validation réussie d'un PT sont :
 - cas:user : identifiant numérique unique d'un utilisateur

⁶ Attention, l'URL de cette page doit être celle inscrite dans le document LOMFR. En effet, le ticket PT généré par CAS prend en compte cette URL et la validation du PT vérifie l'adéquation de l'URL passé lors de la génération du PT et l'url courante.

⁷ Voir la page <http://www.ja-sig.org/wiki/display/CASC/Clients>.

- cas:siren : Siren de l'établissement (ou de l'entreprise pour un éditeur)
- cas:rne : Rne de l'établissement (ou de l'entreprise pour un éditeur) (optionnel)
- cas:profile : « EDITEUR », « ELEVE », « PROFESSEUR » ou « AUTRE »
- cas:class : classe de l'élève ou classes de l'enseignant (optionnel)

Document LOMFR de description de ressource

L'adresse URL de la page d'accès à la ressource doit être consignée dans la section « 4.3 localisation » du document LOMFR correspondant.

Cette information est visible uniquement de l'éditeur de la ressource ou de l'administrateur dans l'application Correlyce et n'est pas divulguée à l'extérieur.

Test

Les tests d'accès à la ressource peuvent être effectués directement par les éditeurs dans leur interface de gestion des titres :

- cliquer sur le contenu de la colonne « Accès » du titre choisi
- cliquer sur le lien « tester » dans la nouvelle page
- l'utilisateur est alors redirigé vers l'url d'accès indiquée dans le document LOMFR avec un ticket valide en paramètre

Annexes

Retour de validation CAS

Authentification réussie

```
<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>
  <cas:authenticationSuccess>
    <cas:user>Uam00010</cas:user>
    <cas:rne>0131313Z</cas:rne>
    <cas:siren>602060147</cas:siren>
    <cas:profile>EDITEUR</cas:profile>
    <cas:proxies>
      <cas:proxy>https://www.correlyce.fr/correlyce/casProxy/receptor</cas:proxy>
    </cas:proxies>
  </cas:authenticationSuccess>
</cas:serviceResponse>
```

Le tag « cas:authenticationSuccess » indique que l'authentification est réussie.

Authentification échouée

```
<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>
  <cas:authenticationFailure code='INVALID_TICKET'>
    le ticket 'ST-63-PJraQneKSfeXgrI4pvn20sXlFKQHcLWuMrU-20' est inconnu
  </cas:authenticationFailure>
</cas:serviceResponse>
```

Le tag « cas:authenticationFailure » indique que l'authentification a échoué

Page d'accès exemple en PHP

```
<?php
/* =====
 * = Constantes
 * ===== */
// L'adresse de base du serveur CAS
define('CAS_BASE_URL', 'https://cas.correlyce.fr/cas/');

// L'adresse de la ressource que vous voulez mettre à disposition de Correlyce
define('SERVICE_URL', 'https://demo.tech.fr/correlyce/');

/* =====
 * = Fonctions
 * ===== */
// Fonction pratique pour récupérer le contenu d'un tag
function get_node_value($domElt, $nodeName) {
    $nodes = $domElt->get_elements_by_tagname($nodeName);
    if (sizeof($nodes) == 0) {
        return '';
    }
    return trim($nodes[0]->get_content());
}

/* =====
 * = Programme principal
 * ===== */
// Pas de ticket passé en paramètre => refuser l'accès
if (!isset($_GET['ticket'])) {
    die("Acces interdit [1]");
}

// Récupération du ticket
$ticket = $_GET['ticket'];

// Validation du ticket reçu
$xmlcontent = file_get_contents(CAS_BASE_URL . 'proxyValidate'
    . '?ticket=' . $ticket . '&service=' . urlencode(SERVICE_URL));

// Parsing du XML, XML mal formé => refuser l'accès
if (!$dom = domxml_open_mem($xmlcontent)) {
    die("Acces interdit [2]");
}
```

```

// On vérifie que le nom de l'élément principal de la réponse
// est bien "serviceResponse"
$rootElt = $dom->document_element();
if ($rootElt->node_name() != 'serviceResponse') {
    die("Accès interdit [3]");
}

// Ensuite, on cherche authenticationSuccess
$successElts = $rootElt->get_elements_by_tagname('authenticationSuccess');
if (sizeof($successElts) == 0) {
    die("Accès interdit [4]");
}

// Et on affiche toutes les infos dont on dispose
$user = get_node_value($successElts[0], 'user');
$profile = get_node_value($successElts[0], 'profile');
$class = get_node_value($successElts[0], 'class');
$rne = get_node_value($successElts[0], 'rne');
$siren = get_node_value($successElts[0], 'siren');
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Accès à la ressource : ok</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
</head>
<body>
<strong>Identifiant numérique</strong> : <?php echo $user; ?><br/>
<strong>Rôle</strong> : <?php echo $profile; ?><br/>
<strong>Classe(s)</strong> : <?php echo $class; ?><br/>
<strong>RNE</strong> : <?php echo $rne; ?><br/>
<strong>SIREN</strong> : <?php echo $siren; ?><br/>
</body>
</html>

```

Notes

Ce fichier d'exemple nécessite que PHP soit installé avec les modules openssl, domxml et la directive `allow_url_fopen=On`.

Page d'accès exemple en Java

```
package fr.tech.cas;

import java.io.IOException;
import java.io.PrintWriter;
import java.io.StringReader;

import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.xml.parsers.ParserConfigurationException;

import org.jdom.Document;
import org.jdom.Element;
import org.jdom.Namespace;
import org.jdom.input.SAXBuilder;
import org.xml.sax.SAXException;

import edu.yale.its.tp.cas.client.ProxyTicketValidator;

public class IndexServlet extends HttpServlet {

    /** Serial version UID. */
    private static final long serialVersionUID = 6081252057505750709L;

    // CAS Namespace
    private static final Namespace CAS_NS
        = Namespace.getNamespace("cas", "http://www.yale.edu/tp/cas");

    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws IOException {

        String CASProxyValidate =
            "https://cas.correlyce.fr/cas/proxyValidate" ;
        String monService = "https://correlyce1.crdp-aix-
marseille.fr:8443/CASClient/servlet/Index";

        // Pas de ticket => pas d'accès
        PrintWriter pw = response.getWriter();
        String ticket = request.getParameter("ticket");
        if (ticket == null) {
            pw.println("Acces interdit [1]");
            return;
        }

        ProxyTicketValidator pv = new ProxyTicketValidator();
        pv.setCasValidateUrl(CASProxyValidate);
        pv.setServiceTicket(ticket);
        pv.setService(monService);
        try {
            pv.validate();
        } catch (SAXException e) {
            pw.println("Acces interdit [2]");
            return;
        } catch (ParserConfigurationException e) {
            pw.println("Acces interdit [3]");
            return;
        }
    }
}
```

```
// Si la validation n'est pas ok => pas d'accès
if (!pv.isAuthenticationSuccessful()) {
    pw.println("Acces interdit [4]");
    return;
}

// Si le document XML rendu est invalide => pas d'accès
Document dom = null;
try {
    SAXBuilder builder = new SAXBuilder();
    dom = builder.build(new StringReader(pv.getResponse()));
} catch (Exception ex) {
    pw.println("Acces interdit [5]");
    return;
}

// Si on ne trouve pas l'élément "authenticationSuccess"
// => pas d'accès
Element rootElt = dom.getRootElement();
Element authSuccessElt
    = rootElt.getChild("authenticationSuccess", CAS_NS);
if (authSuccessElt == null) {
    pw.println("Acces interdit [6]");
    return;
}

// Affichage des informations utilisateur
pw.println("Utilisateur : "
    + authSuccessElt.getChildText("user", CAS_NS));
pw.println("Rôle : "
    + authSuccessElt.getChildText("profile", CAS_NS));
pw.println("Classe(s) : "
    + authSuccessElt.getChildText("classes", CAS_NS));
pw.println("RNE : " + authSuccessElt.getChildText("rne", CAS_NS));
pw.println("SIREN : "
    + authSuccessElt.getChildText("siren", CAS_NS));
}
}
```

Notes

Cet exemple nécessite l'utilisation de la librairie CAS Java⁸ et de JDOM⁹.

⁸ <http://www.yale.edu/tp/cas/cas-client-java-2.1.0/webDoc/>

⁹ <http://www.jdom.org>

Page d'accès exemple en ASP

Contribution de Pacôme Massol (CRDP Aix-Marseille)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <title>Test CAS - Correlyce</title>
  <style type="text/css">
    pre { background-color: #feffcd; border: 1px solid #ffbd6f; padding: 1em; }
  </style>
</head>
<body>
<pre>
<%@ Page aspcompat=true Language="VB" Debug="true" %>
<%
dim cas = "<mettre_l'URL_du_serveur_CAS>"
dim service = "<mettre_l'URL_de_la_ressource>"
dim ticket = request("ticket")

Const SXH_SERVER_CERT_IGNORE_ALL_SERVER_ERRORS = 13056

if ticket = "" then
  response.write("Pas de ticket, pas de ressource :-)")
else
  dim objxml = Server.CreateObject("Msxml2.ServerXMLHTTP.5.0")
  ' ligne ci-dessous pour accepter les connexions SSL si le certificat
est incorrect
  ' donc à supprimer dans un cas réel
  objxml.setOption(2) = SXH_SERVER_CERT_IGNORE_ALL_SERVER_ERRORS
  dim url = cas & "proxyValidate?ticket=" & ticket & "&service=" &
service

  objxml.open("GET", url, false)
  objxml.send()

  response.write(Server.HtmlEncode(objxml.responseText))
end if
%>
</pre>
</body>
</html>
```