

Contents

Author: [Drew Mazurek](#)

Contributors:

- [Susan Bramhall](#)
- [Howard Gilbert](#)
- [Andy Newman](#)
- [Andrew Petro](#)

Version: 1.0

Release Date: May 4, 2005

Copyright © 2005, Yale University

1. Introduction

This is the official specification of the CAS 1.0 and 2.0 protocols. It is subject to change.

1.1. Conventions & Definitions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119^[1].

- “Client” refers to the end user and/or the web browser.
- “Server” refers to the Central Authentication Service server.
- “Service” refers to the application the client is trying to access.
- “Back-end service” refers to the application a service is trying to access on behalf of a client. This can also be referred to as the “target service.”
- <LF> is a bare line feed (ASCII value 0x0a).

2. CAS URIs

CAS is an HTTP^[2,3]-based protocol that requires each of its components to be accessible through specific URIs. This section will discuss each of the URIs.

2.1. /login as credential requestor

The /login URI operates with two behaviors: as a credential requestor, and as a credential acceptor. It responds to credentials by acting as a credential acceptor and otherwise acts as a credential requestor.

If the client has already established a single sign-on session with CAS, the web browser presents to CAS a secure cookie containing a string identifying a ticket-granting ticket. This cookie is called the ticket-granting cookie. If the ticket-granting cookie keys to a valid ticket-granting ticket, CAS may issue a service ticket provided all the other conditions in this specification are met. See Section 3.6 for more information on ticket-granting cookies.

2.1.1. parameters

The following HTTP request parameters may be passed to /login while it is acting as a credential requestor. They are all case-sensitive, and they all MUST be handled by /login.

- service [OPTIONAL] - the identifier of the application the client is trying to access. In almost all cases, this will be the URL of the application. Note that as an HTTP request parameter, this URL value MUST be URL-encoded as described in Section 2.2 of RFC 1738^[4]. If a service is not specified and a single sign-on session does not yet exist, CAS SHOULD request credentials from the user to initiate a single sign-on session. If a service is not specified and a single sign-on session already exists, CAS SHOULD display a message notifying the client that it is already logged in.
- renew [OPTIONAL] - if this parameter is set, single sign-on will be bypassed. In this case, CAS will require the client to present credentials regardless of the existence of a single sign-on session with CAS. This parameter is not compatible with the “gateway” parameter. Services redirecting to the /login URI and login form views posting to the /login URI SHOULD NOT set both the “renew” and “gateway” request parameters. Behavior is undefined if both are set. It is RECOMMENDED that CAS implementations ignore the “gateway” parameter if “renew” is set. It is RECOMMENDED that when the renew parameter is set its value be “true”.
- gateway [OPTIONAL] - if this parameter is set, CAS will not ask the client for credentials. If the client has a pre-existing single sign-on session with CAS, or if a single sign-on session can be established through non-interactive means (i.e. trust authentication), CAS MAY redirect the client to the URL specified by the “service” parameter, appending a valid service ticket. (CAS also MAY interpose an advisory page informing the client that a CAS authentication has taken place.) If the client does not have

a single sign-on session with CAS, and a non-interactive authentication cannot be established, CAS MUST redirect the client to the URL specified by the “service” parameter with no “ticket” parameter appended to the URL. If the “service” parameter is not specified and “gateway” is set, the behavior of CAS is undefined. It is RECOMMENDED that in this case, CAS request credentials as if neither parameter was specified. This parameter is not compatible with the “renew” parameter. Behavior is undefined if both are set. It is RECOMMENDED that when the gateway parameter is set its value be “true”.

2.1.2. URL examples of /login

Simple login example:

```
https://server/cas/login?service=http%3A%2F%2Fwww.service.com
```

Don't prompt for username/password:

```
https://server/cas/login?service=http%3A%2F%2Fwww.service.com&gateway=true
```

Always prompt for username/password:

```
https://server/cas/login?service=http%3A%2F%2Fwww.service.com&renew=true
```

2.1.3. response for username/password authentication

When /login behaves as a credential requestor, the response will vary depending on the type of credentials it is requesting. In most cases, CAS will respond by displaying a login screen requesting a username and password. This page MUST include a form with the parameters, “username”, “password”, and “It”. The form MAY also include the parameter, “warn”. If “service” was specified to /login, “service” MUST also be a parameter of the form, containing the value originally passed to /login. These parameters are discussed in detail in Section 2.2.1. The form MUST be submitted through the HTTP POST method to /login which will then act as a credential acceptor, discussed in Section 2.2.

2.1.4. response for trust authentication

Trust authentication accommodates consideration of arbitrary aspects of the request as a basis for authentication. The appropriate user experience for trust authentication will be highly deployer-specific in consideration of local policy and of the logistics of the particular authentication mechanism implemented.

When `/login` behaves as a credential requestor for trust authentication, its behavior will be determined by the type of credentials it will be receiving. If the credentials are valid, CAS MAY transparently redirect the user to the service. Alternately, CAS MAY display a warning that credentials were presented and allow the client to confirm that it wants to use those credentials. It is RECOMMENDED that CAS implementations allow the deployer to choose the preferred behavior. If the credentials are invalid or non-existent, it is RECOMMENDED that CAS display to the client the reason authentication failed, and possibly present the user with alternate means of authentication (e.g. username/password authentication).

2.1.5. response for single sign-on authentication

If the client has already established a single sign-on session with CAS, the client will have presented its HTTP session cookie to `/login` and behavior will be handled as in Section 2.2.4. However, if the “renew” parameter is set, the behavior will be handled as in Section 2.1.3 or 2.1.4.

2.2. `/login` as credential acceptor

When a set of accepted credentials are passed to `/login`, `/login` acts as a credential acceptor and its behavior is defined in this section.

2.2.1. parameters common to all types of authentication

The following HTTP request parameters MAY be passed to `/login` while it is acting as a credential acceptor. They are all case-sensitive and they all MUST be handled by `/login`.

- `service` [OPTIONAL] - the URL of the application the client is trying to access. CAS MUST redirect the client to this URL upon successful authentication. This is discussed in detail in Section 2.2.4.
- `warn` [OPTIONAL] - if this parameter is set, single sign-on MUST NOT be transparent. The client MUST be prompted before being authenticated to another service.

2.2.2. parameters for username/password authentication

In addition to the OPTIONAL parameters specified in Section 2.2.1, the following HTTP request parameters MUST be passed to `/login` while it is acting as a credential acceptor for username/password authentication. They are all case-sensitive.

- username [REQUIRED] - the username of the client that is trying to log in
- password [REQUIRED] - the password of the client that is trying to log in
- lt [REQUIRED] - a login ticket. This is provided as part of the login form discussed in Section 2.1.3. The login ticket itself is discussed in Section 3.5.

2.2.3. parameters for trust authentication

There are no REQUIRED HTTP request parameters for trust authentication. Trust authentication may be based on any aspect of the HTTP request.

2.2.4. response

One of the following responses MUST be provided by /login when it is operating as a credential acceptor.

- successful login: redirect the client to the URL specified by the “service” parameter in a manner that will not cause the client’s credentials to be forwarded to the service. This redirection MUST result in the client issuing a GET request to the service. The request MUST include a valid service ticket, passed as the HTTP request parameter, “ticket”. See Appendix B for more information. If “service” was not specified, CAS MUST display a message notifying the client that it has successfully initiated a single sign-on session.
- failed login: return to /login as a credential requestor. It is RECOMMENDED in this case that the CAS server display an error message be displayed to the user describing why login failed (e.g. bad password, locked account, etc.), and if appropriate, provide an opportunity for the user to attempt to login again.

2.3. /logout

/logout destroys a client’s single sign-on CAS session. The ticket-granting cookie (Section 3.6) is destroyed, and subsequent requests to /login will not obtain service tickets until the user again presents primary credentials (and thereby establishes a new single sign-on session).

2.3.1. parameters

The following HTTP request parameter MAY be specified to /logout. It is case sensitive and SHOULD be handled by /logout.

- url [OPTIONAL] - if “url” is specified, the URL specified by “url” SHOULD be on the logout page with descriptive text. For example, “The application you just logged out of has provided a link it would like you to follow. Please click here to access [http://www.go-back.edu.“](http://www.go-back.edu.” “http://www.go-back.edu.”“)

2.3.2. response

/logout MUST display a page stating that the user has been logged out. If the “url” request parameter is implemented, /logout SHOULD also provide a link to the provided URL as described in Section 2.3.1.

2.4. /validate [CAS 1.0]

/validate checks the validity of a service ticket./validate is part of the CAS 1.0 protocol and thus does not handle proxy authentication. CAS MUST respond with a ticket validation failure response when a proxy ticket is passed to /validate.

2.4.1. parameters

The following HTTP request parameters MAY be specified to /validate. They are case sensitive and MUST all be handled by /validate.

- service [REQUIRED] - the identifier of the service for which the ticket was issued, as discussed in Section 2.2.1.
- ticket [REQUIRED] - the service ticket issued by /login. Service tickets are described in Section 3.1.
- renew [OPTIONAL] - if this parameter is set, ticket validation will only succeed if the service ticket was issued from the presentation of the user’s primary credentials. It will fail if the ticket was issued from a single sign-on session.

2.4.2. response

/validate will return one of the following two responses:

On ticket validation success:

```
yes<LF>
username<LF>
```

On ticket validation failure:

no<LF>
<LF>

2.4.3. URL examples of /validate

Simple validation attempt:

```
https://server/cas/validate?service=http%3A%2F%2Fwww.service.com&ticket=...
```

Ensure service ticket was issued by presentation of primary credentials:

```
https://server/cas/validate?service=http%3A%2F%2Fwww.service.com&ticket=...
```

2.5. /serviceValidate [CAS 2.0]

/serviceValidate checks the validity of a service ticket and returns an XML-fragment response. /serviceValidate MUST also generate and issue proxy-granting tickets when requested. /serviceValidate MUST NOT return a successful authentication if it receives a proxy ticket. It is RECOMMENDED that if /serviceValidate receives a proxy ticket, the error message in the XML response SHOULD explain that validation failed because a proxy ticket was passed to /serviceValidate.

2.5.1. parameters

The following HTTP request parameters MAY be specified to /serviceValidate. They are case sensitive and MUST all be handled by /serviceValidate.

- service [REQUIRED] - the identifier of the service for which the ticket was issued, as discussed in Section 2.2.1.
- ticket [REQUIRED] - the service ticket issued by /login. Service tickets are described in Section 3.1.
- pgtUrl [OPTIONAL] - the URL of the proxy callback. Discussed in Section 2.5.4.
- renew [OPTIONAL] - if this parameter is set, ticket validation will only succeed if the service ticket was issued from the presentation of the user's primary credentials. It will fail if the ticket was issued from a single sign-on session.

2.5.2. response

/serviceValidate will return an XML-formatted CAS serviceResponse as described in the XML schema in Appendix A. Below are example responses:

On ticket validation success:

```
<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>
  <cas:authenticationSuccess>
    <cas:user>username</cas:user>
    <cas:proxyGrantingTicket>PGTIOU-84678-8a9d...
  </cas:proxyGrantingTicket>
  </cas:authenticationSuccess>
</cas:serviceResponse>
```

On ticket validation failure:

```
<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>
  <cas:authenticationFailure code="INVALID_TICKET">
    Ticket ST-1856339-aA5Yuvrxzpv8TaulcYQ7 not recognized
  </cas:authenticationFailure>
</cas:serviceResponse>
```

2.5.3. error codes

The following values MAY be used as the “code” attribute of authentication failure responses. The following is the minimum set of error codes that all CAS servers MUST implement. Implementations MAY include others.

- INVALID_REQUEST - not all of the required request parameters were present
- INVALID_TICKET - the ticket provided was not valid, or the ticket did not come from an initial login and “renew” was set on validation. The body of the <cas:authenticationFailure> block of the XML response SHOULD describe the exact details.
- INVALID_SERVICE - the ticket provided was valid, but the service specified did not match the service associated with the ticket. CAS MUST invalidate the ticket and disallow future validation of that same ticket.
- INTERNAL_ERROR - an internal error occurred during ticket validation

For all error codes, it is RECOMMENDED that CAS provide a more detailed message as the body of the <cas:authenticationFailure> block of the XML response.

2.5.4. proxy callback

If a service wishes to proxy a client's authentication to a back-end service, it must acquire a proxy-granting ticket. Acquisition of this ticket is handled through a proxy callback URL. This URL will uniquely and securely identify the back-end service that is proxying the client's authentication. The back-end service can then decide whether or not to accept the credentials based on the back-end service's identifying callback URL.

The proxy callback mechanism works as follows:

1. The service that is requesting a proxy-granting ticket specifies upon initial service ticket or proxy ticket validation the HTTP request parameter "pgtUrl" to /serviceValidate (or /proxyValidate). This is a callback URL of the service to which CAS will connect to verify the service's identity. This URL **MUST** be HTTPS, and CAS **MUST** verify both that the SSL certificate is valid and that its name matches that of the service. If the certificate fails validation, no proxy-granting ticket will be issued, and the CAS service response as described in Section 2.5.2 **MUST NOT** contain a <proxyGrantingTicket> block. At this point, the issuance of a proxy-granting ticket is halted, but service ticket validation will continue, returning success or failure as appropriate. If certificate validation is successful, issuance of a proxy-granting ticket proceeds as in step 2.
2. CAS uses an HTTP GET request to pass the HTTP request parameters "pgtId" and "pgtIou" to the pgtUrl. These entities are discussed in Sections 3.3 and 3.4, respectively.
3. If the HTTP GET returns an HTTP status code of 200 (OK), CAS **MUST** respond to the /serviceValidate (or /proxyValidate) request with a service response (Section 2.5.2) containing the proxy-granting ticket IOU (Section 3.4) within the <cas:proxyGrantingTicket> block. If the HTTP GET returns any other status code, excepting HTTP 3xx redirects, CAS **MUST** respond to the /serviceValidate (or /proxyValidate) request with a service response that **MUST NOT** contain a <cas:proxyGrantingTicket> block. CAS **MAY** follow any HTTP redirects issued by the pgtUrl. However, the identifying callback URL provided upon validation in the <proxy> block **MUST** be the same URL that was initially passed to /serviceValidate (or /proxyValidate) as the "pgtUrl" parameter.
4. The service, having received a proxy-granting ticket IOU in the CAS response, and both a proxy-granting ticket and a proxy-granting ticket IOU from the proxy callback, will use the proxy-granting ticket IOU to correlate the proxy-granting ticket with the validation response. The service will then use the proxy-granting ticket for the acquisition of proxy tickets as described in Section 2.7.

2.5.5. URL examples of /serviceValidate

Simple validation attempt:

```
https://server/cas/serviceValidate?service=http%3A%2F%2Fwww.service.com&...
```

Ensure service ticket was issued by presentation of primary credentials:

```
https://server/cas/serviceValidate?service=http%3A%2F%2Fwww.service.com&... ST-1856339-aA
```

Pass in a callback URL for proxying:

```
https://server/cas/serviceValidate?service=http%3A%2F%2Fwww.service.com&...
```

2.6. /proxyValidate [CAS 2.0]

/proxyValidate MUST perform the same validation tasks as /serviceValidate and additionally validate proxy tickets. /proxyValidate MUST be capable of validating both service tickets and proxy tickets.

2.6.1. parameters

/proxyValidate has the same parameter requirements as /serviceValidate. See Section 2.5.1.

2.6.2. response

/proxyValidate will return an XML-formatted CAS serviceResponse as described in the XML schema in Appendix A. Below are example responses:

On ticket validation success:

```
<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>
  <cas:authenticationSuccess>
    <cas:user>username</cas:user>
    <cas:proxyGrantingTicket>PGTIOU-84678-8a9d...</cas:proxyGrantingTicket>
    <cas:proxies>
      <cas:proxy>https://proxy2/pgtUrl</cas:proxy>
      <cas:proxy>https://proxy1/pgtUrl</cas:proxy>
    </cas:proxies>
  </cas:authenticationSuccess>
</cas:serviceResponse>
```

Note that when authentication has proceeded through multiple proxies, the order in which the proxies were traversed MUST be reflected in the <cas:proxies> block. The most recently-visited proxy MUST be the first proxy listed, and all the other proxies MUST be shifted down as new proxies are added. In the above example, the service identified by <https://proxy1/pgtUrl> was visited first, and that service proxied authentication to the service identified by <https://proxy2/pgtUrl>.

On ticket validation failure:

```
<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>
  <cas:authenticationFailure code="INVALID_TICKET">
    ticket PT-1856376-1HMg086Z2ZKeByc5XdYD not recognized
  </cas:authenticationFailure>
</cas:serviceResponse>
```

2.6.3 URL examples of /proxyValidate

/proxyValidate accepts the same parameters as /serviceValidate. See Section 2.5.5 for use examples, substituting “proxyValidate” for “serviceValidate”.

2.7. /proxy [CAS 2.0]

/proxy provides proxy tickets to services that have acquired proxy-granting tickets and will be proxying authentication to back-end services.

2.7.1. parameters

The following HTTP request parameters MUST be specified to /proxy. They are both case-sensitive.

- pgt [REQUIRED] - the proxy-granting ticket acquired by the service during service ticket or proxy ticket validation
- targetService [REQUIRED] - the service identifier of the back-end service. Note that not all back-end services are web services so this service identifier will not always be a URL. However, the service identifier specified here MUST match the “service” parameter specified to /proxyValidate upon validation of the proxy ticket.

2.7.2. response

/proxy will return an XML-formatted CAS serviceResponse as described in the XML schema in Appendix A. Below are example responses:

On request success:

```
<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>
  <cas:proxySuccess>
    <cas:proxyTicket>PT-1856392-b98xZrQN4p90ASrw96c8</cas:proxyTicket>
  </cas:proxySuccess>
</cas:serviceResponse>
```

On request failure:

```
<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>
  <cas:proxyFailure code="INVALID_REQUEST">
    'pgt' and 'targetService' parameters are both required
  </cas:proxyFailure>
</cas:serviceResponse>
```

2.7.3. error codes

The following values MAY be used as the “code” attribute of authentication failure responses. The following is the minimum set of error codes that all CAS servers MUST implement. Implementations MAY include others.

- INVALID_REQUEST - not all of the required request parameters were present
- BAD_PGT - the pgt provided was invalid
- INTERNAL_ERROR - an internal error occurred during ticket validation

For all error codes, it is RECOMMENDED that CAS provide a more detailed message as the body of the <cas:authenticationFailure> block of the XML response.

2.7.4. URL example of /proxy

Simple proxy request:

```
https://server/cas/proxy?targetService=http%3A%2F%2Fwww.service.com&pgt=.....
```

3. CAS Entities

3.1. service ticket

A service ticket is an opaque string that is used by the client as a credential to obtain access to a service. The service ticket is obtained from CAS upon a client’s presentation of credentials and a service identifier to /login as described in Section 2.2.

3.1.1. service ticket properties

- Service tickets are only valid for the service identifier that was specified to /login when they were generated. The service identifier SHOULD NOT be part of the service ticket.
- Service tickets MUST only be valid for one ticket validation attempt. Whether or not validation was successful, CAS MUST then invalidate the ticket, causing all future validation attempts of that same ticket to fail.
- CAS SHOULD expire unvalidated service tickets in a reasonable period of time after they are issued. If a service presents for validation an expired service ticket, CAS MUST respond with a validation failure response. It is RECOMMENDED that the validation response include a descriptive message explaining why validation failed. It is RECOMMENDED that the duration a service ticket is valid before it expires be no longer than five minutes. Local security and CAS usage considerations MAY determine the optimal lifespan of unvalidated service tickets.
- Service tickets MUST contain adequate secure random data so that a ticket is not guessable.
- Service tickets MUST begin with the characters, “ST-”.
- Services MUST be able to accept service tickets of up to 32 characters in length. It is RECOMMENDED that services support service tickets of up to 256 characters in length.

3.2. proxy ticket

A proxy ticket is an opaque string that a service uses as a credential to obtain access to a back-end service on behalf of a client. Proxy tickets are obtained from CAS upon a service’s presentation of a valid proxy-granting ticket (Section 3.3), and a service identifier for the back-end service to which it is connecting.

3.2.1. proxy ticket properties

- Proxy tickets are only valid for the service identifier specified to /proxy when they were generated. The service identifier SHOULD NOT be part of the proxy ticket.
- Proxy tickets MUST only be valid for one ticket validation attempt. Whether or not validation was successful, CAS MUST then invalidate the ticket, causing all future validation attempts of that same ticket to fail.
- CAS SHOULD expire unvalidated proxy tickets in a reasonable period of time after they are issued. If a service presents for validation an expired proxy ticket, CAS MUST respond with a validation failure response. It is RECOMMENDED that the validation response include a descriptive message explaining why validation failed. It is RECOMMENDED that the duration a proxy ticket is valid before it expires be no longer than five

minutes. Local security and CAS usage considerations MAY determine the optimal lifespan of unvalidated proxy tickets.

- Proxy tickets **MUST** contain adequate secure random data so that a ticket is not guessable.
- Proxy tickets **SHOULD** begin with the characters, “PT-”. Proxy tickets **MUST** begin with either the characters, “ST-” or “PT-”.
- Back-end services **MUST** be able to accept proxy tickets of up to 32 characters in length. It is **RECOMMENDED** that back-end services support proxy tickets of up to 256 characters in length.

3.3. proxy-granting ticket

A proxy-granting ticket is an opaque string that is used by a service to obtain proxy tickets for obtaining access to a back-end service on behalf of a client. Proxy-granting tickets are obtained from CAS upon validation of a service ticket or a proxy ticket. Proxy-granting ticket issuance is described fully in Section 2.5.4.

3.3.1. proxy-granting ticket properties

- Proxy-granting tickets **MAY** be used by services to obtain multiple proxy tickets. Proxy-granting tickets are not one-time-use tickets.
- Proxy-granting tickets **MUST** expire when the client whose authentication is being proxied logs out of CAS.
- Proxy-granting tickets **MUST** contain adequate secure random data so that a ticket is not guessable in a reasonable period of time through brute-force attacks.
- Proxy-granting tickets **SHOULD** begin with the characters, “PGT-”.
- Services **MUST** be able to handle proxy-granting tickets of up to 64 characters in length. It is **RECOMMENDED** that services support proxy-granting tickets of up to 256 characters in length.

3.4. proxy-granting ticket IOU

A proxy-granting ticket IOU is an opaque string that is placed in the response provided by /serviceValidate and /proxyValidate used to correlate a service ticket or proxy ticket validation with a particular proxy-granting ticket. See Section 2.5.4 for a full description of this process.

3.4.1. proxy-granting ticket IOU properties

- Proxy-granting ticket IOUs **SHOULD NOT** contain any reference to their associated proxy-granting tickets. Given a particular PGTIU, it **MUST**

NOT be possible to derive its corresponding PGT through algorithmic methods in a reasonable period of time.

- Proxy-granting ticket IOUs MUST contain adequate secure random data so that a ticket is not guessable in a reasonable period of time through brute-force attacks.
- Proxy-granting ticket IOUs SHOULD begin with the characters, “PGTIOU-”.
- Services MUST be able to handle PGTIOUs of up to 64 characters in length. It is RECOMMENDED that services support PGTIOUs of up to 256 characters in length.

3.5. login ticket

A login ticket is a string that is provided by /login as a credential requestor and passed to /login as a credential acceptor for username/password authentication. Its purpose is to prevent the replaying of credentials due to bugs in web browsers.

3.5.1. login ticket properties

- Login tickets issued by /login MUST be probabilistically unique.
- Login tickets MUST only be valid for one authentication attempt. Whether or not authentication was successful, CAS MUST then invalidate the login ticket, causing all future authentication attempts with that instance of that login ticket to fail.
- Login tickets SHOULD begin with the characters, “LT-”.

3.6. ticket-granting cookie

A ticket-granting cookie is an HTTP cookie^[5] set by CAS upon the establishment of a single sign-on session. This cookie maintains login state for the client, and while it is valid, the client can present it to CAS in lieu of primary credentials. Services can opt out of single sign-on through the “renew” parameter described in Sections 2.1.1, 2.4.1, and 2.5.1.

3.6.1. ticket-granting cookie properties

- Ticket-granting cookies MUST be set to expire at the end of the client’s browser session.
- CAS MUST set the cookie path to be as restrictive as possible. For example, if the CAS server is set up under the path /cas, the cookie path MUST be set to /cas.
- The value of ticket-granting cookies MUST contain adequate secure random data so that a ticket-granting cookie is not guessable in a reasonable period of time.

- The value of ticket-granting cookies SHOULD begin with the characters, “TGC-”.

3.7. ticket and ticket-granting cookie character set

In addition to the above requirements, all CAS tickets and the value of the ticket-granting cookie MUST contain only characters from the set {A-Z, a-z, 0-9, and the hyphen character '-'}

Appendix A: CAS response XML schema

<!--

The following is the schema for the Yale Central Authentication

Service (CAS) version 2.0 protocol response. This covers the responses

for the following servlets:

/serviceValidate

/proxyValidate

/proxy

This specification is subject to change.

Author: Drew Mazurek

Version: \$Id: cas2.xsd,v 1.1 2005/02/14 16:19:06 dmazurek Exp \$

-->

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

```
    xmlns:cas="http://www.yale.edu/tp/cas"
```

```
    targetNamespace="http://www.yale.edu/tp/cas"
```

```
    elementFormDefault="qualified" attributeFormDefault="unqualified">
```

```
<xs:element name="serviceResponse" type="cas:ServiceResponseType"/>
```

```
<xs:complexType name="ServiceResponseType">
```

```
<xs:choice>
```

```
<xs:element name="authenticationSuccess" type="cas:AuthenticationSuccessType"/>
```

```
<xs:element name="authenticationFailure" type="cas:AuthenticationFailureType"/>
```

```
<xs:element name="proxySuccess" type="cas:ProxySuccessType"/>

<xs:element name="proxyFailure" type="cas:ProxyFailureType"/>

</xs:choice>

</xs:complexType>

<xs:complexType name="AuthenticationSuccessType">

  <xs:sequence>

    <xs:element name="user" type="xs:string"/>

    <xs:element name="proxyGrantingTicket" type="xs:string" minOccurs="0"/>

    <xs:element name="proxies" type="cas:ProxiesType" minOccurs="0"/>

  </xs:sequence>

</xs:complexType>

<xs:complexType name="ProxiesType">
```

```
<xs:sequence>

<xs:element name="proxy" type="xs:string" maxOccurs="unbounded"/>

</xs:sequence>

</xs:complexType>

<xs:complexType name="AuthenticationFailureType">

<xs:simpleContent>

<xs:extension base="xs:string">

<xs:attribute name="code" type="xs:string" use="required"/>

</xs:extension>

</xs:simpleContent>

</xs:complexType>
```

```
<xs:complexType name="ProxySuccessType">

    <xs:sequence>

        <xs:element name="proxyTicket" type="xs:string"/>

    </xs:sequence>

</xs:complexType>

<xs:complexType name="ProxyFailureType">

    <xs:simpleContent>

        <xs:extension base="xs:string">

            <xs:attribute name="code" type="xs:string" use="required"/>

        </xs:extension>

    </xs:simpleContent>

</xs:complexType>
```

```
</xs:schema>
```

Appendix B: Safe redirection

After a successful login, safely redirecting the client from CAS to its final destination must be handled with care. In most cases, the client has sent credentials to the CAS server over a POST request. By this specification, the CAS server must then forward the user to the application with a GET request.

The HTTP/1.1 RFC^[3] provides a response code of 303: See Other, which provides for the desired behavior: a script that receives data through a POST request can, through a 303 redirection, forward the browser to another URL through a GET request. However, not all browsers have implemented this behavior correctly.

The recommended method of redirection is thus JavaScript. A page containing a `window.location.href` in the following manner performs adequately:

```
<html>
```

```
  <head>
```

```
    <title>Yale Central Authentication Service</title>
```

```
    <script>
```

```
      window.location.href="https://portal.yale.edu/Login?ticket=ST-..." mce_href="http
```

```
    </script>
```

```
  </head>
```

```
<body>
```

```
<noscript>
```

```
<p>CAS login successful.</p>
```

```
<p> Click <a xhref="https://portal.yale.edu/Login?ticket=ST-..." mce_href="https:
```

```
to access the service you requested.<br /> </p>
```

```
</noscript>
```

```
</body>
```

```
</html>
```

Additionally, CAS should disable browser caching by setting all of the various cache-related headers:

- Pragma: no-cache
- Cache-Control: no-store
- Expires: [RFC 1123^[6] date equal to or before now]

The introduction of the login ticket removed the possibility of CAS accepting credentials that were cached and replayed by a browser. However, early versions of Apple's Safari browser contained a bug where through usage of the Back button, Safari could be coerced into presenting the client's credentials to the service it is trying to access. CAS can prevent this behavior by not automatically

redirecting if it detects that the remote browser is one of these early versions of Safari. Instead, CAS should display a page that states login was successful, and provide a link to the requested service. The client must then manually click to proceed.

Appendix C: References

- [1] Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels”, RFC 2119, Harvard University, March 1997.
- [2] Berners-Lee, T., Fielding, R., Frystyk, H., “Hypertext Transfer Protocol - HTTP/1.0”, RFC 1945, MIT/LCS, UC Irvine, MIT/LCS, May 1996.
- [3] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., “Hypertext Transfer Protocol - HTTP/1.1”, RFC 2068, UC Irvine, Compaq/W3C, Compaq, W3C/MIT, Xerox, Microsoft, W3C/MIT, June 1999.
- [4] Berners-Lee, T., Masinter, L., and MaCahill, M., “Uniform Resource Locators (URL)”, RFC 1738, CERN, Xerox Corporation, University of Minnesota, December 1994.
- [5] Kristol, D., Montulli, L., “HTTP State Management Mechanism”, RFC 2965, Bell Laboratories/Lucent Technologies, Epinions.com, Inc., October 2000.
- [6] Braden, R., “Requirements for Internet Hosts - Application and Support”, RFC 1123, Internet Engineering Task Force, October 1989.

Appendix D: CAS License

Copyright (c) 2000-2005 Yale University. All rights reserved.

THIS SOFTWARE IS PROVIDED “AS IS,” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE EXPRESSLY DISCLAIMED. IN NO EVENT SHALL YALE UNIVERSITY OR ITS EMPLOYEES BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED, THE COSTS OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH DAMAGE.

Redistribution and use of this software in source or binary forms, with or without modification, are permitted, provided that the following conditions are met:

1. Any redistribution must include the above copyright notice and disclaimer and this list of conditions in any related documentation and, if feasible, in the redistributed software.
2. Any redistribution must include the acknowledgment, “This product includes software developed by Yale University,” in any related documentation and, if feasible, in the redistributed software.
3. The names “Yale” and “Yale University” must not be used to endorse or promote products derived from this software.

Appendix E: Changes to this Document

May 4, 2005: v1.0 - initial release

March 2, 2012: v1.0.1 - fixed “noscript” typo. apetro per amazurek with credit to Faraz Khan at ASU for catching the typo.