CAS Protocol 3.0 Specification

<https://apereo.github.io/cas/5.2.x/protocol/CAS-Protocol-Specification.html>

# **作者&版本**

Author: Drew Mazurek

Contributors:

* Susan Bramhall
* Howard Gilbert
* Andy Newman
* Andrew Petro
* Robert Oschwald [CAS 3.0]
* Misagh Moayyed

Version: 3.0.2

Release Date: 2015-01-13

Copyright © 2005， Yale University

Copyright © 2017， Apereo Inc.

# **简介**

这是CAS协议 1.0，2.0和3.0版本的官方规范。

Central Authentication Service(以下简称CAS)是一个网络的单点登陆(SSO)/单点登出(SLO)协议。它允许用户在访问多个应用程序时，仅需向中央CAS服务器提供一次凭据(如userid和password)

## 约定和定义

本文档中出现的关键字“MUST”，“MUST NOT”， “REQUIRED”， “SHALL”， “SHALL NOT”， “SHOULD”， “SHOULD NOT”， “RECOMMENDED”， “MAY”， 和 “OPTIONAL” 依照[RFC 2119](http://www.ietf.org/rfc/rfc2119.txt)中的描述进行解释。

* “Client” 是指终端用户和/或Web浏览器。
* “CAS Client” 是指与Web应用程序集成并通过CAS协议与CAS服务器交互的软件组件。
* “Server” 是指CAS中央身份验证服务服务器。
* “Service” 是指客户试图访问的应用程序。
* “Back-end service” 是指服务试图代表客户端访问的应用程序。这也可以被称为“目标服务”。
* “SSO” 是指单点登录。
* “SLO” 是指单点登出。
* ”<LF>” 是空行(ASCII值0x0a)

## 参考实现

[Apereo CAS-Server](https://www.apereo.org/projects/cas)是CAS协议规范的官方参考实现。

Apereo CAS Server 4.x/5.x支持CAS协议3.0规范。

# **CAS URIs**

CAS是一个基于HTTP的协议，要求每个组件都可以通过特定的URI访问*。*本节将讨论每个URI：

|  |  |
| --- | --- |
| URI | Description |
| [*/login*](#_/login_凭证请求者) | 凭证请求/接受 |
| [*/logout*](#_/logout) | 销毁CAS session(注销) |
| [*/validate*](#_/validate_[CAS_1.0]) | 服务Ticket验证 |
| */*[*serviceValidate*](#_/serviceValidate_[CAS_2.0]) | 服务Ticket验证[CAS 2.0] |
| [/proxyValidate](#_/proxyValidate_[CAS_2.0]) | 服务/代理Ticket验证[CAS 2.0] |
| [/proxy](#_/proxy_[CAS_2.0]) | 代理Ticket服务[CAS 2.0] |
| [/p3*/serviceValidate*](#_/p3/serviceValidate_[CAS_3.0]) | 服务Ticket验证[CAS 3.0] |
| [/p3/proxyValidate](#_/p3/proxyValidate_[CAS_3.0]) | 服务/代理Ticket验证[CAS 3.0] |

## ***/login* 凭证请求者**

*/login* URI使用两种行为：作为凭证请求者和凭证接受者。它通过充当凭证接受者来对凭证做出响应，并且以其他方式充当凭证请求者。

如果客户端已经与CAS建立单一登录会话(SSO session)，则Web浏览器向CAS提交一个包含字符串的安全cookie用以鉴定TGT(ticket-granting ticket)，这个cookie被称为TGC(ticket-granting cookie)*。*如果TGC给有效的票证授予票证，则CAS MAY 颁发ST(service ticket)，只要符合本规范中的所有其他条件即可。有关TGC的更多信息，请参阅第3.6节。

### 参数

下面的HTTP请求参数可以作为凭证请求者传递给*/login。*区分大小写的， MUST由 */login*来处理。

* *service* [OPTIONAL] -客户端试图访问的应用程序的标识符。几乎所有情况下，这就是应用程序的URL。作为HTTP请求参数，该URL值MUST按照[RFC 3984](https://tools.ietf.org/rfc/rfc3984.txt)第2.2节中的描述进行URL编码。
	+ 如果未指定service并且SSO session尚不存在，CAS SHOULD 向用户请求凭据以启动SSO session。
	+ 如果未指定service并且SSO session已存在，CAS SHOULD 显示一条消息，通知客户端它已经登录。

注意：强烈RECOMMENDED通过服务管理工具对所有service URL进行过滤，以便只有授权和已知的客户端应用才能使用CAS服务器。将服务管理工具开放以允许对所有应用程序的宽松访问将潜在地增加服务攻击和其他安全漏洞的风险。

此外，RECOMMENDED仅允许https等安全协议用于客户端应用程序，以进一步加强身份验证客户端。

* *renew* [OPTIONAL] - 如果设置此参数，SSO将被绕过。在这种情况下，无论CAS是否存在SSO session，CAS都要求客户端提供凭证。
	+ 该参数与*gateway*参数不兼容。重定向到 */login* URI的登录表单视图和发布到 */login* URI的服务SHOULD NOT 同时设置*renew*和*gateway*请求参数。如果两者都设置，则请求不明确。
	+ RECOMMENDED CAS设置时忽略*gateway*参数。RECOMMENDED在设置*renew*参数时，其值为“true”。
* *gateway* [OPTIONAL] -如果设置了此参数，则CAS不会要求客户端提供凭据。
	+ 如果客户端与CAS预先存在SSO session，或者可以通过非交互方式(即信任认证)建立SSO session，CAS MAY将客户端重定向到*service*指定的URL，并附加一个有效的service ticket(CAS还可以设置一个通知客户端CAS认证已经发生的建议页面)。
	+ 如果客户端没有与CAS的SSO session，并且无法建立非交互式认证，而且没有*ticket*参数附加到URL，则CAS MUST重定向客户端到*service*参数指定的URL，
	+ 如果没有指定*service*参数并且设置了*gateway*，则CAS的行为是未定义的。RECOMMENDED在这种情况下，CAS请求证书，就好像两个参数都没有指定一样。
	+ 该参数与*renew*参数不兼容。如果两者都设置，则请求不明确。RECOMMENDED设置*gateway*参数时，其值为“true”。
* *method* [OPTIONAL, CAS 3.0] - 发送回复时使用的方法*。*虽然本机HTTP重定向(GET)可以用作默认方法，但需要POST响应的应用程序可以使用此参数来指示方法类型*。*还可以指定HEADER方法来指示CAS最终响应，例如服务和票据应以HTTP响应头的形式返回。由CAS服务器实现决定是否支持POST或HEADER响应。

###  */login*的URL示例

简单的login例子：

[https://cas.example.org/cas*/login*?service=http%3A%2F%2Fwww.example.org%2Fservice](https://cas.example.org/cas/login?service=http%3A%2F%2Fwww.example.org%2Fservice)

不要提示输入用户名/密码：

[https://cas.example.org/cas*/login*?service=http%3A%2F%2Fwww.example.org%2Fservice&gateway=true](https://cas.example.org/cas/login?service=http%3A%2F%2Fwww.example.org%2Fservice&gateway=true)

始终提示输入用户名/密码：

[https://cas.example.org/cas*/login*?service=http%3A%2F%2Fwww.example.org%2Fservice&renew=true](https://cas.example.org/cas/login?service=http%3A%2F%2Fwww.example.org%2Fservice&renew=true)

使用POST响应而不是重定向：

[https://cas.example.org/cas*/login*?method=POST&service=http%3A%2F%2Fwww.example.org%2Fservice](https://cas.example.org/cas/login?method=POST&service=http%3A%2F%2Fwww.example.org%2Fservice)

### username/password认证的响应

当*/login*用作凭证请求者时，响应将根据它请求的凭证类型而变化*。*在大多数情况下，CAS将对通过显示请求username和password的登录视窗进行响应。该页面MUST包含一个带有参数“username”，“password”和“lt”的表单。表单MAY包含参数“warn”*。*如果service被指定为*/login*，service也MUST是表单的一个参数，包含最初传递给 */login*的值。这些参数将在第2.2.1节中详细讨论。必须通过HTTP POST方法将表单提交给 */login*，然后将其作为凭证接受者，在2.2节讨论。

### 已信任认证的响应

信任认证适应对请求的任意方面的考虑作为认证的基础。考虑到本地政策和所实施的特定认证机制的后勤工作，适当的用户体验信任认证将高度依赖部署者。

当 */login*用作信任认证的凭证请求者时，其行为将取决于它将接收的凭证类型。如果凭证有效，CAS MAY将用户透明地重定向到服务。或者，CAS MAY会显示警告，显示凭证并允许客户确认它想要使用这些凭证。RECOMMENDED CAS实施允许部署者选择首选行为。如果凭证无效或不存在，RECOMMENDED CAS向客户端显示认证失败的原因，并可能向用户提供其他认证方式(例如用户名/密码认证)。

### 单点登录认证的响应

如果客户端已经与CAS建立了SSO session，则客户端将向*/login* 提交HTTP会话的cookie，并按照第2.2.4节中的说明处理行为。但是，如果设置了*renew*参数，则行为将按照2.1.3或2.1.4节的规定进行处理。

## */login*凭证接受者

当一组接受的凭证传递给*/login*时，*/login*将充当凭证接受者，其行为在本节中定义。

###  所有类型认证通用的参数

以下HTTP请求参数MAY作为凭证接受方传递给*/login。*区分大小写，MUST通过*/login*处理。

* *service* [OPTIONAL] -客户端尝试访问应用程序的URL。作为一个HTTP请求参数，这个URL值MUST按照[RFC1738](https://tools.ietf.org/rfc/rfc1738.txt)第2.2节的描述进行URL编码*。*CAS MUST在成功认证后将客户端重定向到此URL，这在第2.2.4节中详细讨论。在CAS服务器以非开放模式运行时(允许使用CAS服务器的service URL在CAS服务器中注册)，如果出现非授权service的URL时，CAS服务器MUST拒绝操作并输出消息。

注意：强烈RECOMMENDED通过服务管理工具对所有service URL进行过滤，以便只有授权和已知的客户端应用才能使用CAS服务器。将服务管理工具开放以允许对所有应用程序的宽松访问将潜在地增加服务攻击和其他安全漏洞的风险。

此外，RECOMMENDED仅允许https等安全协议用于客户端应用程序，以进一步加强身份验证客户端。

* *warn* [OPTIONAL] - 如果设置了此参数，则SSO MUST NOT透明的。在被认证到另一个service之前，客户端MUST被提示。
* *method* [OPTIONAL] - 发送回复时使用的方法。详情请参阅2.1.1节

### username/password认证的参数

除了第2.2.1节中规定的可选参数之外，以下HTTP请求参数MUST 在用作用户名/密码认证的凭证接受者时传递给*/login*。区分大小写。

* *username* [REQUIRED] - 试图登录的客户端的用户名
* *password* [REQUIRED] - 尝试登录的客户端用户名的密码
* *lt* [OPTIONAL] - 登录Ticket。这是作为2.1.3节讨论的登录表单的一部分提供的。登录Ticket本身在3.5节中讨论。
* *rememberMe* [OPTIONAL, CAS 3.0] -如果设置了此参数，CAS服务器可创建长期TCT(支持RememberMe功能)。由CAS服务器配置是否支持长期TCT。

注意：CAS服务器支持长期TCT（Remember-Me）时，MUST考虑账号的安全问题。例如包括共享的计算机使用。在CAS客户端系统上，可能需要处理不同的Remember-Me登录。详情请参阅第4.1节。

### 已信任认证的参数

对于信任验证没有REQUIRED的HTTP请求参数。 信任认证MAY基于HTTP请求的任何方面。

### 响应

当其作为凭证接收方运行时，*/login* MUST通过以下之一提供回复：

* 登录成功：将客户端重定向到由*service*参数指定的URL，其方式不会导致客户端的凭据被转发到*service*。 这个重定向MUST导致客户端向服务发出一个GET请求。 请求MUST包含一个有效的服务票据，作为HTTP请求参数*ticket*传递。有关更多信息，请参阅附录B。 如果未指定*service*，CAS MUST显示一条消息，通知客户端它已成功启动单一登录会话。
* 登录失败：作为凭证请求者返回/*login*。在这种情况下，RECOMMENDED CAS服务器向用户显示一条错误消息，说明登录失败的原因（例如密码错误，账户锁定等），并在适当的情况下为用户提供尝试重新登录的机会。

## */logout*

*/logout* 销毁与客户端的SSO session*。*TGC(第3.6节)被销毁后，随后的*/login*请求将不会获得Ticket，直到用户再次提供主要凭证(并因此建立新的SSO session)。

### 参数

以下HTTP请求参数MAY被指定为*/logout*。区分大小写，SHOULD由 */logout*来处理。

* *url* [OPTIONAL,CAS 1.0,CAS 2.0] – 如果指定了url参数，则指定的URL SHOULD显示在logout页面上，并且要有文字说明。
* *service* [OPTIONAL, CAS 3.0] - 如果指定了服务参数，则在由CAS服务器执行logout后，浏览器可能会自动重定向到由service指定的URL。CAS服务器的重定向实际执行取决于服务器配置。作为HTTP请求参数，service的值MUST按照[RFC 1738](https://tools.ietf.org/rfc/rfc1738.txt)第2.2节的描述进行URL编码。

注意： CAS 2.0规范中定义的*url*参数不再是CAS 3.0中的有效参数。CAS服务器必须忽略给定的*url*参数*。*CAS客户端MAY如上所述提供service参数，因为这确保参数在非开放模式下针对注册的服务URL进行验证。细节见2.3.2。

注意：强烈RECOMMENDED通过服务管理工具对所有service URL进行过滤，以便只有授权和已知的客户端应用才能使用CAS服务器。将服务管理工具开放以允许对所有应用程序的宽松访问将潜在地增加服务攻击和其他安全漏洞的风险。

此外，RECOMMENDED仅允许https等安全协议用于客户端应用程序，以进一步加强身份验证客户端。

###  响应

* */logout* [CAS 1.0, CAS 2.0]
	+ *如果*没有提供*url*参数，MUST显示一个页面，说明用户已经注销。
	+ 如果提供了*url*参数，那么*/logout* 页面SHOULD提供一个链接到对应的URL，如2.3.1节所述。
* */logout* [CAS 3.0]
	+ 如果没有提供service参数，MUST显示一个页面，说明用户已经注销*。*
	+ 如果提供了带有编码URL值的service请求参数，则CAS服务器在成功注销后将客户端重新定向给定的URL。

注意：当CAS服务器以非开放模式运行时（允许的服务URL在CAS服务器内注册），CAS服务器MUST确保只有注册的[service]参数服务URL才能被接受用于重定向。以前的CAS 2.0规范中定义的*url*参数不再是CAS 3.0中的有效参数*。*CAS服务器必须忽略给定的*url*参数。

### 单点登出

CAS服务器MAY支持单点登出(SLO)*。*SLO意味着用户不仅从CAS服务器注销，还从所有访问的CAS客户端应用程序注销*。*

如果CAS服务器支持SLO，则每当TCT被用户明确过期时，CAS服务器MUST向该CAS会话期间向CAS提供的所有服务URL发送包含注销XML文档的HTTP POST请求(请参阅附录C) (例如在注销期间)*。*CAS不支持SLO POST请求的客户端MUST忽略这些请求*。*在TGT空闲超时时，CAS服务器MAY启动SLO请求。

#### 服务器行为

CAS服务器SHALL忽略对CAS客户端应用程序服务URL的单点登出POST请求中可能发生的所有错误*。*这可确保在发送POST请求时发生的任何错误都不会影响CAS服务器的性能和可用性(“fire and forget ”)。

#### 客户端行为

处理注销POST请求数据取决于CAS客户端*。*RECOMMENDED将用户从由SLO POST请求中发送的service ticket ID标识的应用程序中注销*。*如果客户端支持SLO POST请求处理，则客户端SHALL返回HTTP成功状态码。

## */validate* [CAS 1.0]

*/validate*检查服务票证的有效性。*/validate*是CAS 1.0协议的一部分，因此不处理代理验证。当代理票证传递给*/validate*时，CAS MUST响应票证验证失败。

### 参数

以下HTTP请求参数MAY被指定为*/validate*。区分大小写，MUST由*/validate*处理。

* *service* [REQUIRED] - 如第2.2.1节中所讨论的那样，为其发放Ticket的service的标识符*。*作为HTTP请求参数，service的值MUST按照[RFC 1738](https://tools.ietf.org/rfc/rfc1738.txt)第2.2节的描述进行URL编码。

注意：强烈RECOMMENDED通过服务管理工具对所有service URL进行过滤，以便只有授权和已知的客户端应用才能使用CAS服务器。将服务管理工具开放以允许对所有应用程序的宽松访问将潜在地增加服务攻击和其他安全漏洞的风险。

此外，RECOMMENDED仅允许https等安全协议用于客户端应用程序，以进一步加强身份验证客户端。

* *ticket* [REQUIRED] - */login*发出的service ticket*。*service ticket在第3.1节中描述。
* *renew* [OPTIONAL] -如果设置了此参数，则只有在service ticket是从用户的主凭证的呈现中发出时，Ticket验证才会成功*。*如果票据是从SSO session发出的，将会失败。

### 响应

*/validate*将返回以下两个响应之一：

Ticket验证成功: yes

Ticket验证失败: no

### */validate*的URL示例

简单的验证尝试:

https://cas.example.org/cas*/validate*?service=http%3A%2F%2Fwww.example.org%2Fservice&ticket=ST-1856339-aA5Yuvrxzpv8Tau1cYQ7

确保Service ticket是通过主要凭证的显示发出的：

https://cas.example.org/cas*/validate*?service=http%3A%2F%2Fwww.example.org%2Fservice&ticket=ST-1856339-aA5Yuvrxzpv8Tau1cYQ7&renew=true

## */serviceValidate* [CAS 2.0]

*/serviceValidate*检查service ticket的有效性并返回XML。*/serviceValidate*也MUST在请求时生成并发布proxy-granting tickets。*/serviceValidate*如果收到proxy-granting tickets，MUST NOT返回成功的身份验证。RECOMMENDED如果 */serviceValidate*接收到proxy-granting tickets，XML响应中的错误消息SHOULD解释验证失败，因为传递了代理票证给*/serviceValidate*。

### 参数

以下HTTP请求参数MAY被指定为*/serviceValidate*。区分大小写，MUST全部由*/serviceValidate*处理。

* *service* [REQUIRED] - 如第2.2.1节中所讨论的那样，为其发放Ticket的service的标识符*。*作为HTTP请求参数，service的值MUST按照[RFC 1738](https://tools.ietf.org/rfc/rfc1738.txt)第2.2节的描述进行URL编码。

注意：强烈RECOMMENDED通过服务管理工具对所有service URL进行过滤，以便只有授权和已知的客户端应用才能使用CAS服务器。将服务管理工具开放以允许对所有应用程序的宽松访问将潜在地增加服务攻击和其他安全漏洞的风险。

此外，RECOMMENDED仅允许https等安全协议用于客户端应用程序，以进一步加强身份验证客户端。

* *ticket* [REQUIRED] - */login*发出的service ticket。service ticket在第3.1节中描述。
* *pgtUrl* [OPTIONAL] - 代理回调的URL。在2.5.4节中讨论。作为HTTP请求参数，“pgtUrl”值MUST按照[RFC 1738](https://tools.ietf.org/rfc/rfc1738.txt)第2.2节中的描述进行URL编码。
* *renew* [OPTIONAL] -如果设置了此参数，则只有在service ticket是从用户的主凭证的呈现中发出时，Ticket验证才会成功*。*如果票据是从SSO session发出的，将会失败。
* *format* [OPTIONAL] - 如果设置了此参数，则MUST根据参数值生成故障单验证响应*。*支持的值是XML和JSON*。*如果此参数未设置，则将使用默认的XML格式*。*如果CAS服务器不支持该参数值，则MUST按照第2.5.3节中的描述返回一个错误代码。

### 响应

*/serviceValidate* 将返回XML格式的CAS serviceResponse，如附录A中的XML架构中所述。下面是示例响应：

Ticket验证成功：

1. <cas:serviceResponse xmlns:cas="http://www.yale.edu/tp/cas">
2. <cas:authenticationSuccess>
3. <cas:user>username</cas:user>
4. <cas:proxyGrantingTicket>PGTIOU-84678-8a9d...</cas:proxyGrantingTicket>
5. </cas:authenticationSuccess>
6. </cas:serviceResponse>
7. {
8. "serviceResponse" : {
9. "authenticationSuccess" : {
10. "user" : "username",
11. "proxyGrantingTicket" : "PGTIOU-84678-8a9d..."
12. }
13. }
14. }

Ticket验证失败：

1. <cas:serviceResponse xmlns:cas="http://www.yale.edu/tp/cas">
2. <cas:authenticationFailure code="INVALID\_TICKET">
3. Ticket ST-1856339-aA5Yuvrxzpv8Tau1cYQ7 not recognized
4. </cas:authenticationFailure>
5. </cas:serviceResponse>
6. {
7. "serviceResponse" : {
8. "authenticationFailure" : {
9. "code" : "INVALID\_TICKET",
10. "description" : "Ticket ST-1856339-aA5Yuvrxzpv8Tau1cYQ7 not recognized"
11. }
12. }
13. }

代理响应请参见第2.6.2节。.

### 错误码

以下值可以用作认证失败响应的“code”属性*。*以下是所有CAS服务器MUST执行的最小错误代码集。实现MAY包括其他。

* *INVALID\_REQUEST* - 并非所有需要的请求参数都存在/请求参数不完整
* *INVALID\_TICKET\_SPEC* - 未能满足验证规范的要求
* *UNAUTHORIZED\_SERVICE\_PROXY* - 该service未被授权执行代理身份验证
* *INVALID\_PROXY\_CALLBACK* - 指定的代理回调无效。为代理身份验证指定的凭据不符合安全要求
* *INVALID\_TICKET* - 所提供的票证无效，或者票证不是来自初始登录，并且在验证时设置了更新。XML响应的<cas：authenticationFailure>块的主体SHOULD描述确切的细节。
* *INVALID\_SERVICE* -提供的Ticket有效，但指定的服务与Ticket关联的service不匹配*。*CAS服务器MUST使票无效并且不允许对相同票的验证。
* *INTERNAL\_ERROR* - 票证验证期间发生内部错误

对于所有的错误代码，RECOMMENDED CAS服务器在响应的XML<cas：authenticationFailure>块中提供更详细的消息

### 代理回调

如果服务希望将客户端的身份验证代理到后端服务，则它必须获取proxy-granting ticket(PGT)。通过代理回调URL处理此Ticket。该URL将唯一且安全地标识代理客户端身份验证的服务。然后后端服务可以决定是否接受基于代理服务标识回调URL的凭证。

代理回调机制的工作原理如下：

1. 请求proxy-granting ticket (PGT)的service在初始服务票证或代理票证验证时将HTTP请求参数“pgtUrl”指定为*/serviceValidate*(或*/proxyValidate*)。这是CAS连接到service的回调URL，以验证service的身份。该URL MUST是HTTPS，CAS MUST评估端点建立对等信任。建立信任至少涉及使用PKIX和容器信任来验证回调URL的证书的签名，链和过期窗口。PGT或相应PGTIOU的生成可能由于代理回叫URL不能满足最低安全要求(诸如在对等端之间建立信任失败或端点不响应等等)而失败。在失败时，不会发布PGT，并且第2.5.2节中所述的CAS服务响应MUST NOT包含<proxyGrantingTicket>块。此时，PGT的颁发将停止，service ticket验证将失败。否则，过程将正常进行到步骤2。
2. CAS使用HTTP GET请求将HTTP请求参数pgtId和pgtIou传递给pgtUrl端点。这些实体分别在3.3节和3.4节中讨论。如果代理回调url指定了参数，那么MUST保留这些参数*。*CAS还MUST通过验证来自GET请求的响应HTTP状态码来确保端点可达，如步骤＃3所详述。如果代理服务未能通过身份验证，或者端点以不可接受的状态码进行响应，则代理身份验证MUST失败，并且CAS必须按照第2.5.3节中的描述使用适当的错误代码进行响应。
3. 如果HTTP GET返回200(OK)的HTTP状态代码，则CAS MUST响应*/serviceValidate*(或*/proxyValidate*)请求，并在服务响应(第2.5.2节)中包含<cas：proxyGrantingTicket>块的PGTIOU(第3.4节)。如果HTTP GET返回除HTTP 3xx重定向之外的任何其他状态码，CAS MUST响应*/serviceValidate*(或*/proxyValidate*)请求，并且服务响应MUST NOT包含<cas：proxyGrantingTicket>块*。*CAS MAY遵循由pgtUrl发布的任何HTTP重定向。但是，在<proxy>块中验证时提供的标识回调URL MUST与pgtUrl参数最初传递给*/serviceValidate*(或*/proxyValidate*)的URL相同。
4. 该服务在CAS响应中收到PGT IOU，以及来自代理回调的PGT和PGT IOU都将使用PGT IOU关联PGT与验证响应。然后服务将使用PGT获取代理票据，如第2.7节所述。

### 属性[CAS 3.0]

[CAS 3.0] 响应文档MAY包含用于附加认证和/或用户属性的可选元素*。*有关详细信息，请参阅[附录A]。

### serviceValidate的URL实例

简单的验证尝试：

https://cas.example.org/cas*/serviceValidate*?service=http%3A%2F%2Fwww.example.org%2Fservice&ticket=ST-1856339-aA5Yuvrxzpv8Tau1cYQ7

简单的验证尝试：确保服务票证是通过主要凭证的呈现发布的：

https://cas.example.org/cas*/serviceValidate*?service=http%3A%2F%2Fwww.example.org%2Fservice&ticket=ST-1856339-aA5Yuvrxzpv8Tau1cYQ7&renew=true

传递代理的回调URL：

https://cas.example.org/cas*/serviceValidate*?service=http%3A%2F%2Fwww.example.org%2Fservice&ticket=ST-1856339-aA5Yuvrxzpv8Tau1cYQ7&pgtUrl=https://www.example.org%2Fservice%2FproxyCallback

### 使用自定义属性的响应示例

1. <cas:serviceResponse xmlns:cas="http://www.yale.edu/tp/cas">
2. <cas:authenticationSuccess>
3. <cas:user>username</cas:user>
4. <cas:attributes>
5. <cas:firstname>John</cas:firstname>
6. <cas:lastname>Doe</cas:lastname>
7. <cas:title>Mr.</cas:title>
8. <cas:email>jdoe@example.org</cas:email>
9. <cas:affiliation>staff</cas:affiliation>
10. <cas:affiliation>faculty</cas:affiliation>
11. </cas:attributes>
12. <cas:proxyGrantingTicket>PGTIOU-84678-8a9d...</cas:proxyGrantingTicket>
13. </cas:authenticationSuccess>
14. </cas:serviceResponse>
15. {
16. "serviceResponse" : {
17. "authenticationSuccess" : {
18. "user" : "username",
19. "proxyGrantingTicket" : "PGTIOU-84678-8a9d...",
20. "proxies" : [ "https://proxy1/pgtUrl", "https://proxy2/pgtUrl" ],
21. "attributes" : {
22. "firstName" : "John",
23. "affiliation" : [ "staff", "faculty" ],
24. "title" : "Mr.",
25. "email" : "jdoe@example.orgmailto:jdoe@example.org",
26. "lastname" : "Doe"
27. }
28. }
29. }
30. }

## */proxyValidate* [CAS 2.0]

*/proxyValidate* MUST执行与/serviceValidate相同的验证任务并另外验证代理票证(proxy tickets)。*/proxyValidate* MUST能够验证service ticket和代理票据。详情请参见第2.5.4节。

### 参数

*/proxyValidate*与*/serviceValidate*具有相同的参数要求。参见第2.5.1节。

### 响应

*/proxyValidate*将返回XML格式的CAS serviceResponse，如附录A中的XML架构中所述。下面是示例响应：

Ticket验证成功响应：

1. <cas:serviceResponse xmlns:cas="http://www.yale.edu/tp/cas">
2. <cas:authenticationSuccess>
3. <cas:user>username</cas:user>
4. <cas:proxyGrantingTicket>PGTIOU-84678-8a9d...</cas:proxyGrantingTicket>
5. <cas:proxies>
6. <cas:proxy>https://proxy2/pgtUrl</cas:proxy>
7. <cas:proxy>https://proxy1/pgtUrl</cas:proxy>
8. </cas:proxies>
9. </cas:authenticationSuccess>
10. </cas:serviceResponse>
11. {
12. "serviceResponse" : {
13. "authenticationSuccess" : {
14. "user" : "username",
15. "proxyGrantingTicket" : "PGTIOU-84678-8a9d...",
16. "proxies" : [ "https://proxy1/pgtUrl", "https://proxy2/pgtUrl" ]
17. }
18. }
19. }

注意：当验证通过多个代理进行时，代理遍历的顺序MUST反映在块中*。*最近访问的代理MUST是列出的第一个代理，并且所有其他代理MUST在新代理添加时向下移动*。*在上面的例子中，首先访问由<https：// proxy1 / pgtUrl>标识的服务，并且该服务代理了由<https：// proxy2 / pgtUrl>标识的服务的身份验证。

Ticket验证失败响应：

1. <cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>
2. <cas:authenticationFailure code="INVALID\_TICKET">
3. ticket PT-1856376-1HMgO86Z2ZKeByc5XdYD not recognized
4. </cas:authenticationFailure>
5. </cas:serviceResponse>
6. {
7. "serviceResponse" : {
8. "authenticationFailure" : {
9. "code" : "INVALID\_TICKET",
10. "description" : "Ticket PT-1856339-aA5Yuvrxzpv8Tau1cYQ7 not recognized"
11. }
12. }
13. }

### 错误码

见 2.5.3

### */proxyValidate*的URL实例

*/proxyValidate*接受与*/serviceValidate*相同的参数*。*有关使用示例，请参见第2.5.5节，用“proxyValidate”替换“serviceValidate”。

## */proxy* [CAS 2.0]

*/proxy*为已获取proxy-granting tickets的服务提供代理票证，并将代理身份验证到后端服务。

### 参数

以下HTTP请求参数MUST指定给*/proxy*。区分大小写。

* *pgt* [REQUIRED] - 在服务票证或代理票证验证期间由服务获取的代理授权票证。
* *targetService* [REQUIRED] - 后端服务的服务标识符。请注意，并非所有后端服务都是Web服务，因此服务标识符不会始终是URL。但是，此处指定的服务标识符MUST与验证代理票证时指定给*/proxyValidate*的服务参数匹配。

### 响应

/*proxyValidate*将返回XML格式的CAS serviceResponse，如附录A中的XML架构中所述。下面是示例响应：

请求成功响应：

1. <cas:serviceResponse xmlns:cas="http://www.yale.edu/tp/cas">
2. <cas:proxySuccess>
3. <cas:proxyTicket>PT-1856392-b98xZrQN4p90ASrw96c8</cas:proxyTicket>
4. </cas:proxySuccess>
5. </cas:serviceResponse>

请求失败响应：

1. <cas:serviceResponse xmlns:cas="http://www.yale.edu/tp/cas">
2. <cas:proxyFailure code="INVALID\_REQUEST">
3. 'pgt' and 'targetService' parameters are both required
4. </cas:proxyFailure>
5. </cas:serviceResponse>
6. {
7. "serviceResponse" : {
8. "authenticationFailure" : {
9. "code" : "INVALID\_REQUEST",
10. "description" : "'pgt' and 'targetService' parameters are both required"
11. }
12. }
13. }

### 错误码

以下值MAY用作认证失败响应的代码属性*。*以下是所有CAS服务器MUST执行的最小错误代码集*。*实现MAY包括其他。

* *INVALID\_REQUEST* - 并非所有需要的请求参数都存在
* *UNAUTHORIZED\_SERVICE* - 服务未经授权执行代理请求
* *INTERNAL\_ERROR* - 票证验证期间发生内部错误

对于所有的错误代码，RECOMMENDED CAS服务器在响应的XML<cas：authenticationFailure>块中提供更详细的消息

### */proxy*的URL实例

简单的代理请求：

https://server/cas/proxy?targetService=http%3A%2F%2Fwww.service.com&pgt=PGT-490649-W81Y9Sa2vTM7hda7xNTkezTbVge4CUsybAr

### service ticket生命周期影响

CAS服务器实现MAY在代理票据生成时更新父服务票据(ST)生存期。

## */p3/serviceValidate* [CAS 3.0]

*/p3/serviceValidate* MUST执行与*/serviceValidate*相同的验证任务，并在CAS响应中返回用户属性*。*有关详细信息，请参阅第2.5节和第2.5.7节。

### 参数

*/p3/serviceValidate*具有与*/serviceValidate*相同的参数要求*。*参见第2.5.1节。

## */p3/proxyValidate* [CAS 3.0]

*/p3/proxyValidate* MUST执行与/p3/serviceValidate相同的验证任务并另外验证代理票证*。*见2.8节。

### 参数

*/p3/proxyValidate*与*/p3/serviceValidate*具有相同的参数要求*。*见2.8.1节。

# **CAS 实体**

## service ticket

service ticket是一个不透明的字符串，客户端使用该字符串作为凭证来获得对service的访问权限*。*service ticket是客户端如第2.2节所述的那样通过向*/login*提交凭证和service标识符后，从CAS服务器获取的。

### service ticket 属性

* service ticket仅对于在生成时指定为*/login*的服务标识符有效。service标识符SHOULD NOT是service ticket的一部分。
* service ticket MUST仅适用于一次ticket确认尝试。无论验证是否成功，CAS MUST使ticket失效，同一ticket的所有未来验证尝试都失败。
* CAS SHOULD在发布后的合理时间内将未经验证的service ticket过期。如果某个service提供了过期的service ticket进行验证，则CAS MUST响应验证失败。
* RECOMMENDED验证响应包含解释验证失败原因的描述性信息。
* RECOMMENDED service ticket到期前的有效期不超过五分钟。本地安全和CAS使用情况考虑MAY确定未经验证的服务请求的最佳使用期限。
* service ticket MUST包含足够的安全随机数据，以便票据不被猜测。
* service ticket MUST以字符ST-开头。
* service MUST能够接受长达32个字符的service ticket。RECOMMENDED service支持长度不超过256个字符的service ticket。

## proxy ticket

proxy ticket是一种不透明的字符串，service用它作为凭证代表客户获得对后端服务的访问权限*。*proxy ticket通过service提供的有效代理授权票证(第3.3节)以及与其连接的后端服务的服务标识符从CAS获得。

### proxy ticket 属性

* proxy ticket仅在为其生成时指定给*/proxy*的service标识符有效。service标识符SHOULD NOT是代理票证的一部分。
* proxy ticket MUST仅适用于一次票证验证尝试。无论验证是否成功，CAS MUST使票证失效，导致同一票证的所有未来验证尝试都失败。
* CAS SHOULD在发布后的合理时间内将未经验证的proxy ticket过期。如果某个service提供了验证过期的proxy ticket，CAS MUST响应验证失败响应。
* RECOMMENDED验证响应包含解释验证失败原因的描述性信息。
* RECOMMENDED proxy ticket在到期之前有效的时间不超过五分钟。本地安全和CAS使用情况考虑可以确定未经验证的proxy ticket的最佳使用寿命。
* proxy ticket MUST包含足够的安全随机数据，以便票据不被猜测。
* proxy ticket SHOULD以字符PT-开头。
* 后端服务MUST能够接受长度最多为32个字符的proxy ticket。
* RECOMMENDED后端服务支持长度不超过256个字符的proxy ticket。

## proxy-granting ticket

proxy-granting ticket (PGT)是一种不透明的字符串，服务使用该字符串来获取用于代表客户端获得对后端服务的访问权的proxy ticket*。*代理授予票证是在验证服务票证或代理票证后从CAS获取的*。*代理授权票发行在第2.5.4节中有详细描述。

### proxy-granting ticket 属性

* proxy-granting ticket MAY由服务使用以获得多个代理票*。*代理授予票不是一次性使用票。
* proxy-granting ticket MUST在代理身份验证的客户端注销CAS时过期。
* proxy-granting ticket MUST包含足够的安全随机数据，以便通过暴力攻击在合理的时间段内无法猜测票证。
* proxy-granting ticket SHOULD以字符PGT-开头。
* service MUST能够处理长达64个字符的proxy-granting ticket。
* RECOMMENDED service支持长度不超过256个字符的proxy-granting ticket。

## proxy-granting ticket IOU

proxy-granting ticket IOU是放置在由*/serviceValidate*和*/proxyValidate*提供的响应中的一个不透明字符串，用于将service ticket或proxy ticket验证与特定的proxy-granting ticket相关联*。*有关此过程的完整说明，请参见第2.5.4节。

### proxy-granting ticket IOU属性

* proxy-granting ticket IOU SHOULD NOT包含对其关联的代理授予票据的任何引用*。*给定一个特定的PGTIOU，MUST NOT在合理的时间段内通过算法方法推导出相应的PGT。
* proxy-granting ticket IOU MUST包含足够安全的随机数据，以便通过暴力攻击在合理的时间段内无法猜测票证。
* proxy-granting ticket IOU SHOULD以字符PGTIOU-开头。
* service MUST能够处理长达64个字符的PGTIOU*。*RECOMMENDED service支持最长256个字符的PGTIOU。。

## login ticket

login ticket是一个可选字符串，可以通过*/login*作为凭证请求者提供，并作为用户名/密码认证的凭证接受者传递*/login。*其目的是防止由于Web浏览器中的错误而导致的证书重放。

### login ticket 属性

* */login*发出的login ticket MUST是概率独一无二的。
* login ticket MUST仅适用于一次认证尝试*。*无论身份验证是否成功，CAS MUST使login ticket失效，导致所有未来对该登录票证实例的身份验证尝试失败。
* login ticket SHOULD以字符LT-开头。

## ticket-granting cookie

ticket-granting cookie是CAS在建立单一登录会话时设置的HTTP cookie *。*这个cookie保持客户端的登录状态，并且当它有效时，客户端可以将其呈现给CAS以代替主要凭证*。*service可以通过2.1.1，2.4.1和2.5.1节中描述的*renew*参数选择退出单一登录。

### ticket-granting cookie 属性

* 如果相应TGT的长期支持未激活(4.1.1)，在客户端浏览器会话结束时， SHALL将ticket-granting cookie设置为过期。
* CAS SHALL将Cookie路径设置为尽可能限制*。*例如，如果CAS服务器设置在路径/cas下，则Cookie路径SHALL设置为/cas。
* ticket-granting cookie的价值SHALL包含足够的安全随机数据，以便票据授予cookie在合理的时间段内不被猜测。
* ticket-granting cookie的名字SHOULD以TGC-字符开头。
* ticket-granting cookie的值SHOULD遵循与票证授予票据相同的规则*。*通常，票证授予cookie的值MAY包含票证授予票证本身作为经过身份验证的单一登录会话的表示形式。

## ticket and ticket-granting cookie character set

除上述要求外，所有CAS的ticket和ticket and ticket-granting cookie的值MUST仅包含集合{A-Z，a-z，0-9}中的字符和连字符 - 。

## ticket-granting ticket

ticket-granting ticket (TGT)是由CAS服务器生成的一个不透明的字符串，该字符串在*/login*时通过成功的身份验证事件发出*。*此票证可能与ticket-granting cookie相关联，该票据代表单点登录会话的状态，并具有有效期限，并作为发布service ticket，proxy-granting ticket等的基础和基线。

### ticket-granting ticket 属性

* ticket-granting ticket MAY被服务使用以获得多个服务票*。*票证授予票不是一次性使用票，并且与有效期限和到期政策相关联。
* 当正在管理身份验证的客户端注销CAS时，ticket-granting ticket MUST到期。
* ticket-granting ticket MUST包含足够安全的随机数据，以便通过暴力攻击在合理的时间段内不能猜测票证。
* ticket-granting ticketSHOULD以TGT-字符开头。
* RECOMMENDED将ticket-granting ticket与其他外部资源共享时加密，以尽量减少安全漏洞，因为它们与票证授予cookie相关联并代表身份验证会话。

# **Optional Features**

## **Long-Term Tickets - Remember-Me [CAS 3.0]**

CAS服务器MAY支持长期票证授予票证(称为“Remember Me”功能)*。*如果CAS服务器支持此功能，只要CAS服务器中的Long-Term Ticket Granting Ticket未过期且浏览器TGC Cookie有效，就可以对CAS服务器执行周期性的非交互式重新登录。

### 启用 Remember-Me (Login Page)

* CAS服务器MUST在登录页面上提供复选框以允许记住我功能。
* 默认情况下，该复选框MUST未被选中。
* MUST是用户选择启用记住我的登录或不。参见第2.2.2节。

### Security implications

启用记住我MAY会有安全隐患*。*由于CAS认证与浏览器绑定，并且当有效的Long-Term Ticket Granting Ticket存在且浏览器提供的CAS cookie有效时，用户不会进行交互式登录，因此MUST特别注意CAS客户端，以便正确处理Remember-Me登录*。*MUST是CAS客户负责决定是否以及何时记住我的CAS登录可能被特殊处理*。*见4.1.3。

### CAS Validation Response Attributes

由于只有CAS客户端MUST决定如何处理Remember-Me登录(请参阅4.2.1)，CAS服务器必须向CAS客户端提供有关Remember-Me登录的信息*。*在这种情况下，MUST由CAS服务器支持的所有故障单验证方法提供此信息(请参阅第2.5,2.6和2.8节)。

* 在serviceValidate XML响应中(请参阅附录A)，Remember-Me登录必须由longTermAuthenticationRequestTokenUsed属性指示*。*此外，可以使用isFromNewLogin属性来决定是否存在安全隐患。
* 在SAML验证响应中，Remember-Me MUST由

longTermAuthenticationRequestTokenUsed属性指示。

### CAS Client requirements

如果CAS客户端需要处理特殊的Remember-Me登录(例如拒绝访问记住的登录中的CAS客户端应用程序的敏感区域)，则CAS客户端MUST NOT使用/验证CAS验证URL，因为此URL不支持CAS 验证响应文档中的属性。

### Long-Term ticket-granting cookie properties

当CAS Server创建Long-Term Ticket Granting Ticket时，Ticket-granting cookie不得在3.6.1中定义的客户端浏览器会话结束时过期*。*相反，Ticket-granting cookie将在定义的Long-Term Ticket Granting Ticket生命周期中过期。

Long-Term Ticket Granting Ticket的生命周期价值定义取决于CAS服务器实施者*。*长期机票授予机票的有效期MAY NOT超过3个月。

## */samlValidate* [CAS 3.0]

*/samlValidate*通过HTTP POST提供的SAML 1.1请求文档检查服务故障单的有效性*。*MUST返回SAML(安全访问标记语言)7 1.1响应文档*。*这允许释放经认证的NetID的附加信息(属性)*。*安全声明标记语言(SAML)描述了一个文档和协议框架，通过该框架可以交换安全声明(如关于先前认证行为的声明)。

### Parameters

以下HTTP请求参数必须指定给*/samlValidate*。区分大小写。

* *TARGET* [REQUIRED] - 后端服务的URL编码服务标识符*。*请注意，作为HTTP请求参数，此URL值MUST按照[RFC 1738](https://tools.ietf.org/rfc/rfc1738.txt)的第2.2节中的描述进行URL编码*。*此处指定的服务标识符必须与提供给*/login*的服务参数匹配*。*参见第2.1.1节*。*TARGET服务SHALL使用HTTPS*。*MUST NOT将SAML属性发布到非SSL网站。

### HTTP Request Method and Body

请求*/samlValidate* MUST是HTTP POST请求*。*请求主体MUST是文档类型为“text / xml”的有效SAML 1.0或1.1请求XML文档。

### SAML request values

* RequestID [REQUIRED] - 请求的唯一标识符
* IssueInstant [REQUIRED] - 请求的时间戳
* samlp：AssertionArtifact [REQUIRED] - 在登录时作为响应参数获取的有效CAS服务票据*。*参见2.2.4节。

### */samlValidate* 的POST请求实例

1. POST /cas/samlValidate?TARGET=
2. Host: cas.example.com
3. Content-Length: 491
4. Content-Type: text/xml
5. <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
6. <SOAP-ENV:Header/>
7. <SOAP-ENV:Body>
8. <samlp:Request xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol" MajorVersion="1" MinorVersion="1" RequestID="\_192.168.16.51.1024506224022" IssueInstant="2002-06-19T17:03:44.022Z">
9. <samlp:AssertionArtifact>ST-1-u4hrm3td92cLxpCvrjylcas.example.com</samlp:AssertionArtifact>
10. </samlp:Request>
11. </SOAP-ENV:Body>
12. </SOAP-ENV:Envelope>

### SAML response

CAS服务器对*/samlValidate*请求的响应*。*MUST是SAML 1.1响应。

SAML 1.1验证响应：

1. <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
2. <SOAP-ENV:Header />
3. <SOAP-ENV:Body>
4. <Response xmlns="urn:oasis:names:tc:SAML:1.0:protocol" xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
5. xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
6. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" IssueInstant="2008-12-10T14:12:14.817Z"
7. MajorVersion="1" MinorVersion="1" Recipient="https://eiger.iad.vt.edu/dat/home.do"
8. ResponseID="\_5c94b5431c540365e5a70b2874b75996">
9. <Status>
10. <StatusCode Value="samlp:Success">
11. </StatusCode>
12. </Status>
13. <Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion" AssertionID="\_e5c23ff7a3889e12fa01802a47331653"
14. IssueInstant="2008-12-10T14:12:14.817Z" Issuer="localhost" MajorVersion="1"
15. MinorVersion="1">
16. <Conditions NotBefore="2008-12-10T14:12:14.817Z" NotOnOrAfter="2008-12-10T14:12:44.817Z">
17. <AudienceRestrictionCondition>
18. <Audience>
19. https://some-service.example.com/app/
20. </Audience>
21. </AudienceRestrictionCondition>
22. </Conditions>
23. <AttributeStatement>
24. <Subject>
25. <NameIdentifier>johnq</NameIdentifier>
26. <SubjectConfirmation>
27. <ConfirmationMethod>
28. urn:oasis:names:tc:SAML:1.0:cm:artifact
29. </ConfirmationMethod>
30. </SubjectConfirmation>
31. </Subject>
32. <Attribute AttributeName="uid" AttributeNamespace="http://www.ja-sig.org/products/cas/">
33. <AttributeValue>12345</AttributeValue>
34. </Attribute>
35. <Attribute AttributeName="groupMembership" AttributeNamespace="http://www.ja-sig.org/products/cas/">
36. <AttributeValue>
37. uugid=middleware.staff,ou=Groups,dc=vt,dc=edu
38. </AttributeValue>
39. </Attribute>
40. <Attribute AttributeName="eduPersonAffiliation" AttributeNamespace="http://www.ja-sig.org/products/cas/">
41. <AttributeValue>staff</AttributeValue>
42. </Attribute>
43. <Attribute AttributeName="accountState" AttributeNamespace="http://www.ja-sig.org/products/cas/">
44. <AttributeValue>ACTIVE</AttributeValue>
45. </Attribute>
46. </AttributeStatement>
47. <AuthenticationStatement AuthenticationInstant="2008-12-10T14:12:14.741Z"
48. AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
49. <Subject>
50. <NameIdentifier>johnq</NameIdentifier>
51. <SubjectConfirmation>
52. <ConfirmationMethod>
53. urn:oasis:names:tc:SAML:1.0:cm:artifact
54. </ConfirmationMethod>
55. </SubjectConfirmation>
56. </Subject>
57. </AuthenticationStatement>
58. </Assertion>
59. </Response>
60. </SOAP-ENV:Body>
61. </SOAP-ENV:Envelope>

#### 1 SAML CAS response attributes

SAML响应中可能会提供以下附加属性：

* longTermAuthenticationRequestTokenUsed - 如果CAS服务器支持Long-Term Ticket Granting Ticket (Remember-Me)(请参阅第4.1节)，则SAML响应MUST包含此属性以指示记录到CAS客户端的登录。