CAS protocol

The CAS protocol is a simple and powerful ticket-based protocol developed exclusively for CAS. A complete protocol specification may be found here.

It involves one or many clients and one server. Clients are embedded in *CASified* applications (called "CAS services") whereas the CAS server is a standalone component:

The CAS server is responsible for authenticating users and granting accesses to applications

The CAS clients protect the CAS applications and retrieve the identity of the granted users from the CAS server.

The key concepts are:

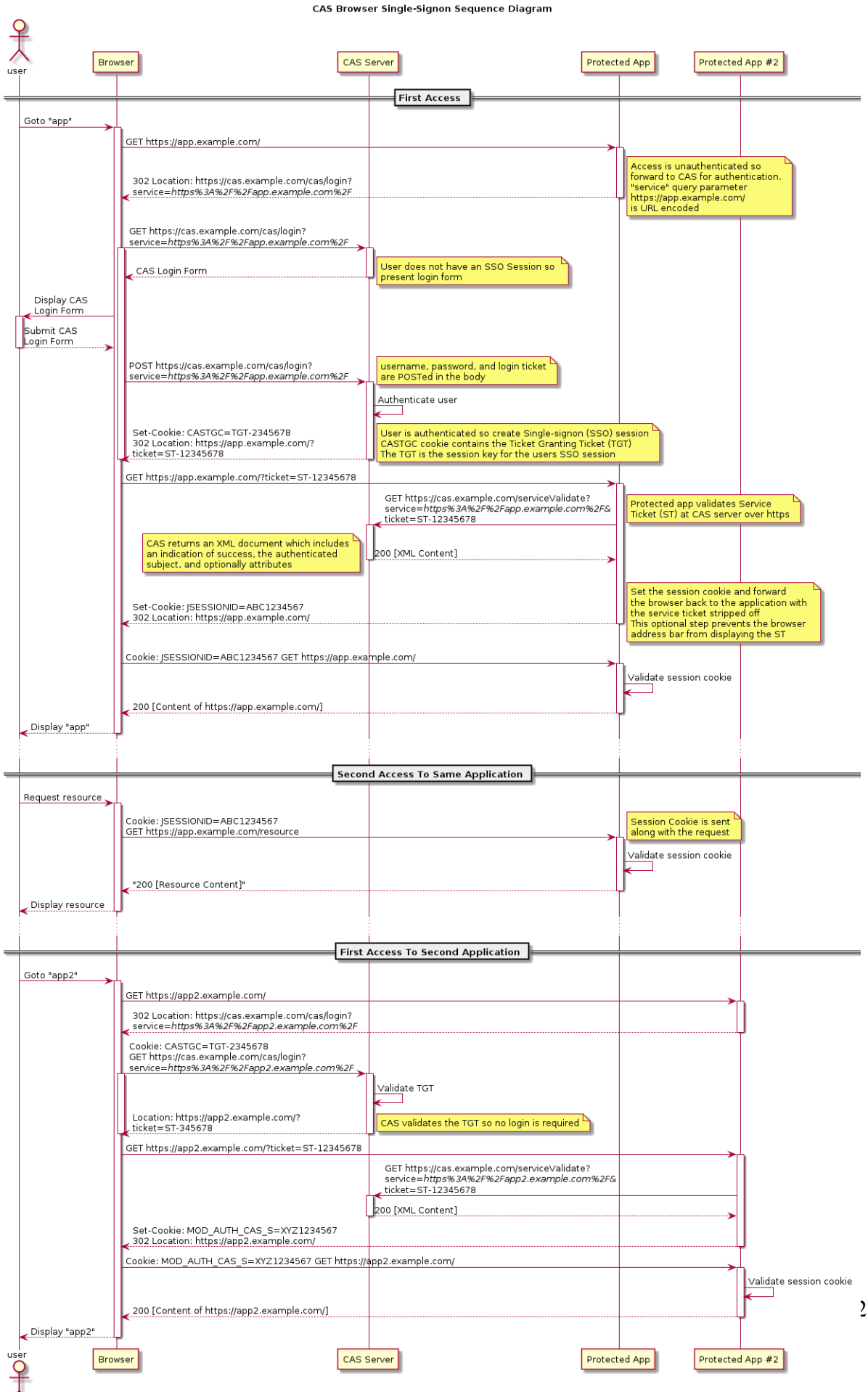
The TGT (Ticket Granting Ticket), stored in the CASTGC cookie, represents a SSO session for a user

The ST (Service Ticket), transmitted as a GET parameter in urls, stands for the access granted by the CAS server to the *CASified* application for a specific user.

Versions

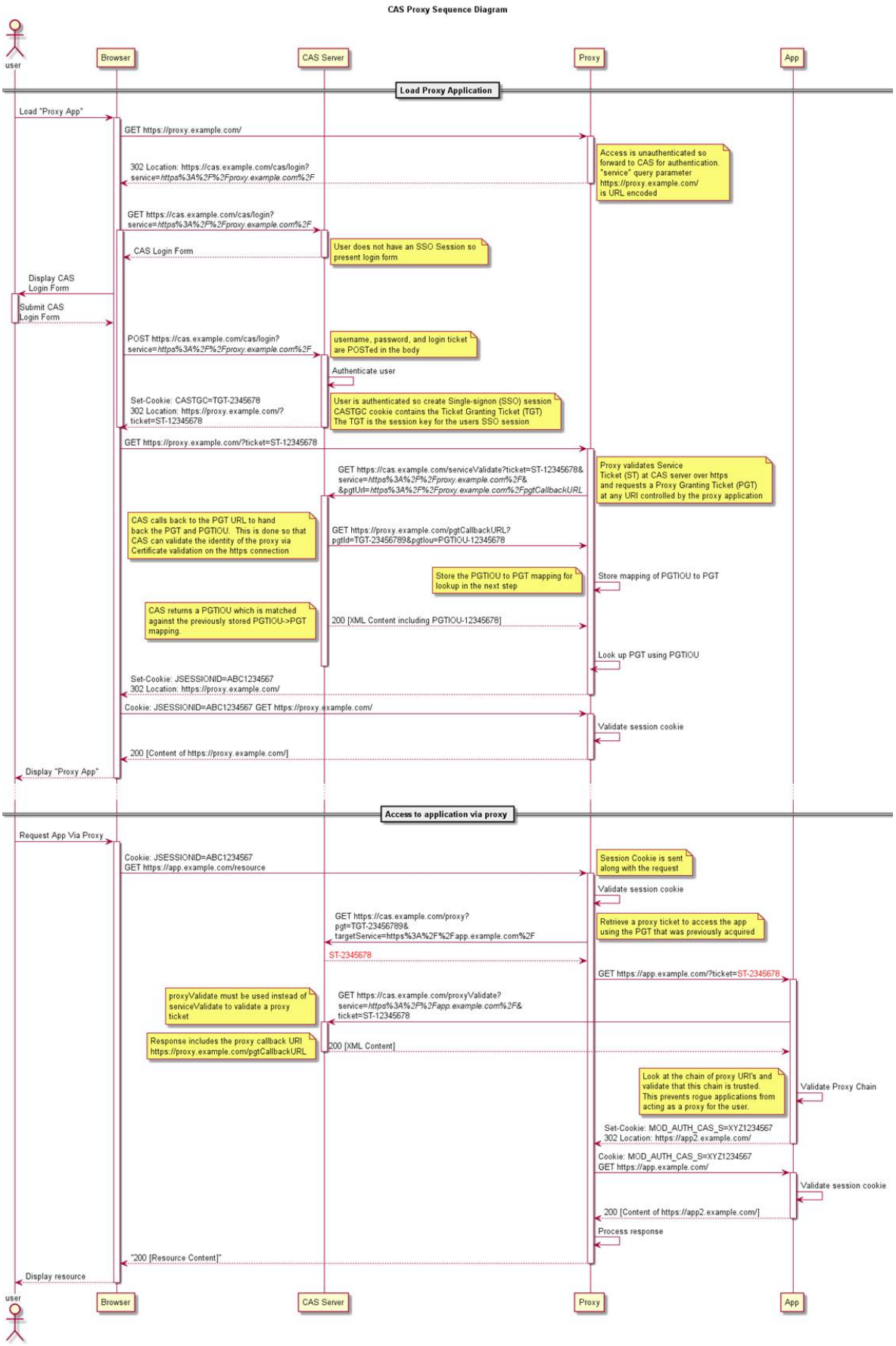
The current CAS protocol is the version 3.0. The draft version of the protocol is available as part of the CAS codebase, which is hereby implemented. It's mainly a capture of the most common enhancements built on top of the CAS protocol revision 2.0. Among all features, the most noticeable update between versions 2.0 and 3.0 is the ability to return the authentication/user attributes through the new `/p3/serviceValidate` response (in addition to the `/serviceValidate` endpoint, already existing for CAS 2.0 protocol).

Web flow diagram


CAS Browser Single-Signon Sequence Diagram
user
Browser
CAS Server
Protected App
Protected App #2
First Access
Goto "app"
GET https://app.example.com/
Access is unauthenticated so forward to CAS for authentication. "service" query parameter https://app.example.com/ is URL encoded
302 Location: https://cas.example.com/cas/login? service=https%3A%2F%2Fapp.example.com%2F
GET https://cas.example.com/cas/login? service=https%3A%2F%2Fapp.example.com%2F
User does not have an SSO Session so present login form
CAS Login Form
Display CAS Login Form
Submit CAS Login Form
POST https://cas.example.com/cas/login? service=https%3A%2F%2Fapp.example.com%2F
username, password, and login ticket are POSTed in the body
Authenticate user
User is authenticated so create Single-signon (SSO) session
CASTGC cookie contains the Ticket Granting Ticket (TGT)
The TGT is the session key for the users SSO session
Set-Cookie: CASTGC=TGT-2345678
302 Location: https://app.example.com/? ticket=ST-12345678
GET https://app.example.com/?ticket=ST-12345678
GET https://cas.example.com/serviceValidate? service=https%3A%2F%2Fapp.example.com%2F& ticket=ST-12345678
Protected app validates Service Ticket (ST) at CAS server over https
CAS returns an XML document which includes an indication of success, the authenticated subject, and optionally attributes
200 [XML Content]
Set the session cookie and forward the browser back to the application with the service ticket stripped off
This optional step prevents the browser address bar from displaying the ST
Set-Cookie: JSESSIONID=ABC1234567
302 Location: https://app.example.com/
Cookie: JSESSIONID=ABC1234567 GET https://app.example.com/
Validate session cookie
200 [Content of https://app.example.com/]
Display "app"
Second Access To Same Application
Request resource
Cookie: JSESSIONID=ABC1234567
GET https://app.example.com/resource
Session Cookie is sent along with the request
Validate session cookie
"200 [Resource Content]"
Display resource
First Access To Second Application
Goto "app2"
GET https://app2.example.com/
302 Location: https://cas.example.com/cas/login? service=https%3A%2F%2Fapp2.example.com%2F
Cookie: CASTGC=TGT-2345678
GET https://cas.example.com/cas/login? service=https%3A%2F%2Fapp2.example.com%2F
Validate TGT
Location: https://app2.example.com/? ticket=ST-345678
CAS validates the TGT so no login is required
GET https://app2.example.com/?ticket=ST-12345678
GET https://cas.example.com/serviceValidate? service=https%3A%2F%2Fapp2.example.com%2F& ticket=ST-12345678
200 [XML Content]
Set-Cookie: MOD_AUTH_CAS_S=XYZ1234567
302 Location: https://app2.example.com/
Cookie: MOD_AUTH_CAS_S=XYZ1234567 GET https://app2.example.com/
Validate session cookie
200 [Content of https://app2.example.com/]
Display "app2"
user
Browser
CAS Server
Protected App
Protected App #2

Proxy web flow diagram

One of the most powerful feature of the CAS protocol is the ability for a CAS service to act as a proxy for another CAS service, transmitting the user identity.


CAS Proxy Sequence Diagram
user
Browser
CAS Server
Proxy
App
Load Proxy Application
Load "Proxy App"
GET https://proxy.example.com/
Access is unauthenticated so forward to CAS for authentication. "service" query parameter https://proxy.example.com/ is URL encoded
302 Location: https://cas.example.com/cas/login?service=https%3A%2F%2Fproxy.example.com%2F
GET https://cas.example.com/cas/login?service=https%3A%2F%2Fproxy.example.com%2F
CAS Login Form
User does not have an SSO Session so present login form
Display CAS Login Form
Submit CAS Login Form
POST https://cas.example.com/cas/login?service=https%3A%2F%2Fproxy.example.com%2F
username, password, and login ticket are POSTed in the body
Authenticate user
Set-Cookie: CASTGC=TGT-2345678
302 Location: https://proxy.example.com/?ticket=ST-12345678
User is authenticated so create Single-signon (SSO) session CASTGC cookie contains the Ticket Granting Ticket (TGT) The TGT is the session key for the users SSO session
GET https://proxy.example.com/?ticket=ST-12345678
GET https://cas.example.com/serviceValidate?ticket=ST-12345678&service=https%3A%2F%2Fproxy.example.com%2F&&pgtUrl=https%3A%2F%2Fproxy.example.com%2FpgtCallbackURL
Proxy validates Service Ticket (ST) at CAS server over https and requests a Proxy Granting Ticket (PGT) at any URI controlled by the proxy application
CAS calls back to the PGT URL to hand back the PGT and PGTIOU. This is done so that CAS can validate the identity of the proxy via Certificate validation on the https connection
GET https://proxy.example.com/pgtCallbackURL?pgtId=TGT-23456789&pgtIou=PGTIOU-12345678
Store the PGTIOU to PGT mapping for lookup in the next step
Store mapping of PGTIOU to PGT
CAS returns a PGTIOU which is matched against the previously stored PGTIOU->PGT mapping.
200 [XML Content including PGTIOU-12345678]
Look up PGT using PGTIOU
Set-Cookie: JSESSIONID=ABC1234567
302 Location: https://proxy.example.com/
Cookie: JSESSIONID=ABC1234567 GET https://proxy.example.com/
Validate session cookie
200 [Content of https://proxy.example.com/]
Display "Proxy App"
Access to application via proxy
Request App Via Proxy
Cookie: JSESSIONID=ABC1234567
GET https://app.example.com/resource
Session Cookie is sent along with the request
Validate session cookie
GET https://cas.example.com/proxy?pgt=TGT-23456789&targetService=https%3A%2F%2Fapp.example.com%2F
Retrieve a proxy ticket to access the app using the PGT that was previously acquired
ST-2345678
GET https://app.example.com/?ticket=ST-2345678
proxyValidate must be used instead of serviceValidate to validate a proxy ticket
GET https://cas.example.com/proxyValidate?service=https%3A%2F%2Fapp.example.com%2F&ticket=ST-12345678
Response includes the proxy callback URI https://proxy.example.com/pgtCallbackURL
200 [XML Content]
Look at the chain of proxy URI's and validate that this chain is trusted. This prevents rogue applications from acting as a proxy for the user.
Validate Proxy Chain
Set-Cookie: MOD_AUTH_CAS_S=XYZ1234567
302 Location: https://app2.example.com/
Cookie: MOD_AUTH_CAS_S=XYZ1234567
GET https://app.example.com/
Validate session cookie
200 [Content of https://app2.example.com/]
Process response
"200 [Resource Content]"
Display resource

Other protocols

Even if the primary goal of the CAS server is to implement the CAS protocol, other protocols are also supported as extensions:

- OpenID
- OAuth
- SAML

Delegated Authentication

Using the CAS protocol, the CAS server can also be configured to delegate the authentication to another CAS server.

Source: http://jasig.github.io/cas/development/protocol/CAS-Protocol.html