链滴

CAS 项目官方文档中文翻译

作者：PeterChen

原文链接：https://ld246.com/article/1600528811637

来源网站：链滴

<img src="https://ld246.com/images/img-loading.svg" alt="" data-src="https://b3logfile
com/bing/20190522.jpg?imageView2/1/w/960/h/540/interlace/1/q/100">
<h2 id="主题介绍">主题介绍</h2>
单点登录（Single Sign On），简称为 SSO，是比较流行的企业业务整合的解决方案之一。SSO
的定义是在多个应用系统中，用户只需要登录一次就可以访问所有相互信任的应用系统。随着互联网
术的发展，企业数字化的普及，单点登录已成为企业数字化建设的必备组件。目前市面上比较流行就
耶鲁大学开源的 CAS 单点登录系统，接下来的一系列文章主要是记录学习 CAS 项目过程中的输出。<
p>
<h2 id="官方文档">官方文档</h2>
CAS 项目源码地址：<a href="https://ld246.com/forward?goto=https%3A%2F%2Fgithub.
om%2Fapereo%2Fcas" target="_blank" rel="nofollow ugc">https://github.com/apereo/cas</
>
<h3 id="CAS-简介">CAS 简介</h3>
<h4 id="CAS-企业级单点登录系统">CAS 企业级单点登录系统</h4>
欢迎来到 Apereo 中心认证服务项目的首页，Apereo 中心认证服务项目通常被简称为 CAS，CA
是一个企业级的多语言支持的单点登录解决方案，她目标成为您身份验证和授权需求的综合平台。</
>
CAS 也是一个开源的文档友好的认证协议，该协议的主要实现是托管在这里的同名开源 Java 服
器组件，它支持大量额外的身份验证协议和特性。
以下内容包括 CAS 项目所提供的特性和技术的摘要：

可以作为 Spring Webflow/Spring Boot 等 Java 服务的组件使用；
支持的可插拔式的认证方式有：LDAP、Database、X.509、SPNEGO、JAAS、JWT、RADIUS
MongoDb 等等；
支持多种协议：CAS、SAML、WS-Federation、OAuth2、OpenID、OpenID Connect、RES
；
依托多个供应商支持多重身份认证：Duo Security、FIDO U2F、YuibiKey、Google Authentica
or、Authy、Acceptto 等等；
支持向外部提供者委托身份验证（第三方认证登录），例如：ADFS、Facebook、Twitter、SAM
2 IdPs 等等；
内置对密码管理、通知、使用条款和模拟的支持；
支持自定义属性发布，包括用户授权同意；
实时监控和跟踪应用程序行为、统计数据和日志；
支持使用特定的身份验证策略管理和注册客户端应用程序和服务；
跨平台终端支持（Java、.Net、PHP、Perl、Apache 等等）;
支持与 InCommon、Box、Office365、ServiceNow、Salesforce、Workday、WebAdvisor、
upal、Blackboard、Moodle、谷歌等应用进行集成。

贡献
想要了解更多如何为 CAS 项目贡献自己的代码，请访问：<a href="https://ld246.com/forwar
?goto=https%3A%2F%2Fgithub.com%2Fapereo%2Fcas%2Fblob%2Fmaster%2Fcas%2Fdevelo
er%2FContributor-Guidelines.html" target="_blank" rel="nofollow ugc">https://github.com/
pereo/cas/blob/master/cas/developer/Contributor-Guidelines.html
快速开始
为了计划和执行 CAS 部署，我们建议阅读以下文档：

<a href="https://ld246.com/forward?goto=https%3A%2F%2Fgithub.com%2Fapereo%2F
as%2Fblob%2Fmaster%2Fdocs%2Fcas-server-documentation%2Fplanning%2FArchitecture.h
ml" target="_blank" rel="nofollow ugc">架构
<a href="https://ld246.com/forward?goto=https%3A%2F%2Fgithub.com%2Fapereo%2F
as%2Fblob%2Fmaster%2Fdocs%2Fcas-server-documentation%2Fplanning%2FGetting-Started
html" target="_blank" rel="nofollow ugc">入门
<a href="https://ld246.com/forward?goto=https%3A%2F%2Fgithub.com%2Fapereo%2F
as%2Fblob%2Fmaster%2Fdocs%2Fcas-server-documentation%2Fplanning%2FInstallation-Re

uirements.html" target="_blank" rel="nofollow ugc">安装要求
安装步骤
博客

技术支持
CAS 的开发由以下工具、项目和服务提供支持：

IntelliJIDEA
Eclipse
Spring Boot
YourKit

YourKit 通过其全功能的 Java Profiler 支持开放源码项目。YourKit, LLC 是 YourKit Java Profile 和 YourKit。net Profiler 的创建者，这是用于分析 Java 和。net 应用程序的创新和智能工具。
<h3 id="CAS项目规划">CAS 项目规划</h3>
<h4 id="架构">架构</h4>
架构图
<img src="https://ld246.com/images/img-loading.svg" alt="casarchitecture.png" data-sr="https://b3logfile.com/file/2020/09/casarchitecture-48a8ba31.png?imageView2/2/interlace//format/jpg">
CAS 组件
CAS 由服务端和客户端两部分组成，这两部分通过多种协议进行通信交互实现单点登录。
CAS Server（服务端）
CAS 服务器是基于 Spring 框架上的 Java 应用，其主要职责是通过发出和验证票据，对用户进身份验证并授予对启用了 CAS 的服务(通常称为 CAS 客户端)的访问权。当服务器向成功登录的用户出票据授予票据(TGT)时，将创建 SSO 会话。应用户的请求，使用 TGT 作为令牌通过浏览器重定向服务发出服务票据(ST)。ST 随后在 CAS 服务器上通过反向通道通信进行验证。这些交互在 CAS 协议档中有详细的描述。
CAS Clients（客户端）
术语“CAS 客户端”在通常的使用中有两个不同的含义。CAS 客户端一方面指任何启用了 CAS 应用程序，可以通过受支持的协议与服务器通信。CAS 客户端另一方面也指代一个软件包，这个软件可以与各种软件平台和应用程序集成，以便通过某些身份验证协议(如 CAS、SAML、OAuth)与 CAS 务器通信。支持多个软件平台和产品的 CAS 客户端已经开发完成。
支持平台：

Apache httpd Server (<a href="https://ld246.com/forward?goto=https%3A%2F%2Fgithubcom%2FJasig%2Fmod_auth_cas" target="_blank" rel="nofollow ugc">mod_auth_cas module/a>)
Java (Java CAS Client)
.NET (.NET CAS Client)
PHP (phpCAS)
Perl (PerlCAS)
Python (pycas)
Ruby (rubycas-client)

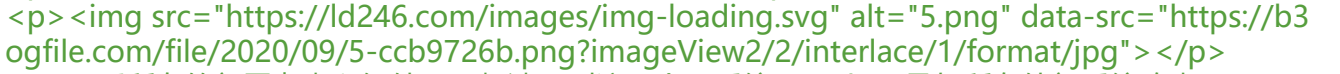
应用示例：

Canvas

Atlassian Confluence
Atlassian JIRA
Drupal
Liferay
uPortal
...

当术语“CAS 客户端”在本手册中没有进一步的限定时，它指的是集成组件，如 Java CAS 客户，而不是依赖于 CAS 服务器(客户端)的应用程序。
支持协议
客户端通过几种受支持的协议中的任何一种与服务器通信。所有受支持的协议在概念上是相似的但是有些协议具有使其适合特定应用程序或用例的特性或特征。例如，CAS 协议支持委托(代理)身份证，SAML 协议支持属性释放和单点签出。
支持的协议有：

CAS (versions 1, 2, and 3)
SAML 1.1 and 2
OpenID Connect
OpenID
OAuth 2.0
WS Federation

软件组件
从三个分层的子系统来描述 CAS 服务器是有帮助的:


几乎所有的部署考虑和组件配置都涉及到这三个子系统。Web 层是与所有外部系统(包括 CAS 户端)通信的端点。Web 层将业务请求委托给票务子系统（Ticketing），以生成用于 CAS 客户端访的票据。SSO 会话依赖认证子系统（Authentication）成功认证身份后生成票据。
CAS 使用了 Spring 框架的许多特性，特别是 Spring MVC 和 Spring Webflow。以 Spring 为础的 CAS 为开发人员提供了一个完整的、可扩展的框架，通过实现 CAS 和 Spring API 扩展点来定或扩展 CAS 非常简单。熟悉 Spring 的一般知识有助于理解 CAS 组件之间的相互作用。能够使用基于xml 的配置是安装、定制和扩展 CAS 的必备基础。能够使用 XML，特别是 Spring IOC 容器是安装AS 的先决条件。
<h4 id="入门">入门</h4>
本文档提供了有关如何开始进行 CAS 服务器部署的高级指南。本指南的唯一重点是描述 CAS 部人员必须遵循并采用的过程，以实现成功且可持续的体系结构和部署。
<h5 id="收集用例">收集用例</h5>
在部署之前，记录，分类和分析所需的用例和需求非常重要。一旦您有了一些想法，请与 CAS 社区讨论分享这些想法， 以了解可能已经解决了您今天面临的相同问题的共同趋势，实践和模式。

要点提示
通常情况下请避免设计或采用会严重改变 CAS 内部组件，给配置的管理和维护带来沉重负担的例和工作流程，或者避免重新发明 CAS 软件及其支持的协议，所有新增选项都会增加维护成本和麻。
<h5 id="研究架构">研究架构</h5>
了解什么是 CAS，并且可以做什么。这将帮助您建立基础，以了解 CAS 可能已经实现了哪些用和需求。查看 CAS 体系结构的基础知识， 以了解哪些选和选择可用于部署和应用程序集成。
同样，研究 CAS 支持的协议和规范列表也同重要。
<h5 id="关注博客">关注博客</h5>
有时，博客文章会出现在 Apereo 博客上 ，当您考虑需求和估功能时，这些博客文章可能会变得有用。通常建议您关注该博客，并尽可能多地了解项目新闻和公，并且不要回避在整个 CAS 部署中撰写和贡献自己的博客文章，经验和更新。
<h5 id="准备环境">准备环境</h5>
根据配置组件的选择，可能会有其他要求，例如 LDAP 数据词典、数据库和缓存架构。但是，在多数情况下，对于选择具有明确硬件和软件依存关系的组件的部署者来说，要求应该是显而易见的。任何其他要求都不明显的情况下，有关组件配置的讨论应提及操作系统、软件、硬件和其他要求。</>
Java
CAS 的核心是基于 Java 的 Web 应用程序。在部署之前，您将需要安装 JDK <code>11</code>。
Oracle JDK License
Oracle 更新了提供 Oracle JDK 的许可条款。针对 Oracle Java SE 的新的 Oracle 技术网络许可议与以前 JDK 版本所使用的许可有本质上的不同。下载和使用本产品前，请仔细阅读新条款。
该许可的关键部分如下:
除开发、测试、原型设计和演示应用程序外，您不得:将程序用于任何数据处理或任何商业、生产内部业务目的。
不要下载或使用 Oracle JDK，除非你打算付费。建议使用 OpenJDK 进行构建。
Servlet 容器要求
没有正式支持 CAS 的 servlet 容器，但是最常用的是 Apache Tocat。对特定 servlet 容器的支持取决于社区成员的专业知识。
有关更多信息，请参见本指南
构建工具
CAS 提供了一种叫做 WAR overlays 的方式以提供一种简单而灵活的部署解决方案。尽管它当然需要很高的前期学习成本，但从长远来，它可以带来许多好处。
提示:您不需要另外安装 Graldle，CAS 项目内置了该工具。
Git(可选)
尽管不是严格要求，但强烈建议您为 CAS 部署安装 <a href="https://ld246.com/forward?got=https%3A%2F%2Fgit-scm.com%2Fdownloads" target="_blank" rel="nofollow ugc">Git</a，并在源代码控制存储库中管理所有 CAS 工件，配置文件，构建脚本和设置。

操作系统
尽管基于 Linux 的安装通常比 Windows 更普遍，但对操作系统没有特别的偏好。
网络连接
在任何基于 Maven / Gradle 的项目的构建阶段，通常都需要 Internet 连接，包括用于安装 CAS 的建议的 WAR 覆盖图。构建过程通过搜索包含在本地下载和安装的工件（大多数情况下为 jar 文件的在线存储库来解决依赖关系。
硬件要求
社区上的传闻似乎表明，CAS 部署至少应在具有 8GB 内存的双核 3.00Ghz 处理器上运行良好。果日志保存在服务器本身上，则还需要足够的磁盘空间（最好是 SSD）来容纳 CAS 生成的日志。</p

请记住，以上要求是建议。您可能会或多或少地完全满意，这取决于您的部署和求量。从最低限度开始，并准备根据需要调整和增强容量。
<h5 id="部署CAS">部署 CAS</h5>
建议使用 WAR Overlay 方法在本构建和部署 CAS 。这种方法不需要采用者显式下载任何版本的 CAS，而是利用覆盖机来组合 CAS 原始工件和本地自定义，以进一步简化将来的升级和维护。
注意：请勿直接克隆或下载 CAS 代码库。仅在您希望为项目的发展做出贡时才需要。
这是非常重要的，你尝试做其他事情之前得到的功能基础工作。避免立即行临时更改以自定义部署。坚持使用 CAS 提供的默认设置和设置，并一次更改一点。跟踪源代码管理中的过程和应用的更改，并在进行修改时标记更改。
<h5 id="定制开发">定制开发</h5>
如果你需要的功能正好映射到 CAS 已有的功能，请浏览文档以找到最接近的匹配项并引用。同，重要的是要尽可能遵守 CAS 基准：

避免对软件内部进行临时更改。
避免对核心配置组件（例如 Spring 和 Spring Webflow）进行手动更改。
如果遇到问题，请避免对部署进行一次性错误修复。

如前所述，所有修改策略都会导致头痛和成本。
相反，请尝试热身以下建议：

错误修复和小改进属于 CAS 的核心软件。不是您的部署。尽一切努力报告问题，提供修补程序补丁，并与 CAS 社区一劳永逸地解决问题。
某些内部 CAS 组件很难进行扩充和修改。在大多数情况下，此方法是有目的地执行的，以使您离危险和不必要的复杂更改。如果您遇到需要并记住其配置和实现需要修改软件核心内部的功能或用，请与 CAS 社区进行讨论，并尝试将增强功能直接构建到 CAS 软件中，而不是处理它像雪花一样。

总而言之，仅在部署配置真正真正地完全满足您的需求时才对其进行更改。否则，请尝试归纳并馈以降低维护成本。从长远来看，不遵守这一战略可能会导致灾难性的后果。
<h5 id="故障排除">故障排除</h5>
该故障排除指南可帮助您，当您遇到些问题时，它通常会尝试描述故障排除和诊断的有用的方法策略。您也可以从 CAS 社区寻求帮助。
<h4 id="安全指南">安全指南</h4>
CAS 是安全软件，可为基于 Web 的应用程序提供基于 Web 的安全单点登录。单一登录在安全和便利性方面提供了双赢：减少了单一信任证书代理对密码的暴露，同时透明地提供了对多种服务的问而无需重复登录。CAS 的使用通常会改善安全性环境，但是要实现适当的安全性，应考虑几个 CAS

配置，策略和部署方面的问题。
报告安全问题
安全团队要求您不要创建公开可见的问题或帖子以讨论您认为的安全漏洞要正确报告问题并了解如何生成响应，请参阅本指南。
<h5 id="公告内容">公告内容</h5>

2020 年 7 月 24 日漏洞披露<a>
2020 年 2 月 8 日漏洞露
2019 年 12 月 20 日漏披露
2019 年 11 月 21 日漏洞披
2019 年 9 月 27 日漏洞露
2018 年 9 月 26 日漏洞露
2017 年 3 月 6 漏洞披露
2016 年 10 月 24 日漏披露
2016 年 4 月 8 日洞披露

<h5 id="注意事项">注意事项</h5>
要考虑的基础结构安全性问题可能包括以下内容：
安全传输（https）
与 CAS 服务器的所有通信都必须通过安全通道（即 TLSv1）进行。此要求有两个主要依据：</p

身份验证过程需要传输安全凭证。
授予 CAS 票证的票证是不记名令牌。

由于任何一种数据的泄露都会允许模拟攻击，因此保护 CAS 客户端和 CAS 服务器之间的通信通至关重要。
实际上，这意味着所有 CAS URL 必须使用 HTTPS，但这也意味着从 CAS 服务器到应用程序的所有连接都必须使用 HTTPS：

当生成的服务票证通过“服务” URL 发送回应用程序时
调用代理回调 URL 时。

要查看 CAS 属性的相关列表并调整此行为，请<a href="https://ld246.com/forward?goto=ht

ps%3A%2F%2Fgithub.com%2Fapereo%2Fcas%2Fblob%2Fmaster%2Fdocs%2Fcas-server-documentation%2Fconfiguration%2FConfiguration-Properties.html%23http-client" target="_blank" rel="nofollow ugc">查看本指南。
系统间安全
CAS 通常需要连接到其他系统，例如 LDAP 目录，数据库和缓存服务。我们通常建议在可能的况下使用到这些系统的安全传输（SSL / TLS，IPSec），但是可能会有补偿性控制措施使安全传输变不必要。具有严格访问控制的专用网络和公司网络是常见的例外，但是仍然建议安全传输。客户端认验证可能是 LDAP 带来足够安全性的另一个好的解决方案。
如前所述，必须确保与其他系统的连接。但是，如果将 CAS 服务器部署在多个节点上，则 CAS 务器本身也是如此。如果基于缓存的票证注册表在单个 CAS 服务器上运行时没有任何安全问题，则网络不受保护的情况下，使用多个节点时同步可能会成为安全问题。
如果没有适当地保护，任何磁盘存储也很容易受到攻击。可以关闭 EhCache 到磁盘的溢出以增保护，而应将高级加密数据机制用于数据库磁盘存储。
<h5 id="安全功能">安全功能</h5>
CAS 支持许多功能，可用于实施各种安全策略。通过 CAS 配置和 CAS 客户端集成提供以下功。请注意，许多功能开箱即用，而其他功能则需要显式设置。
强制认证
许多 CAS 客户端和受支持的协议都支持强制身份验证的概念，用户必须重新进行身份验证才能问特定服务。CAS 协议通过 renew 参数支持强制认证。强制身份验证为 SSO 会话主的身份提供了额外的保证，因为用户必须在访问之前验证其凭据。强制认证适用于需要或要求更高安性的服务。通常，强制认证是基于每个服务配置的，但是服务管理工具会根据集中式安全策略为实施制认证提供一些支持。强制身份验证可以与多因素身份验证结合使用实现任意特定于服务的访问控制略的功能。
被动认证
一些 CAS 协议支持被动身份验证，其中在请求时匿名授予对受 CAS 保护的服务的访问权限。CAv2 和 CASv3 协议通过网关功能支持此功能。被动身份验证补充了强制身份验证；在制身份验证需要身份验证才能访问服务的情况下，被动身份验证允许服务访问（尽管是匿名的），而需身份验证。
代理认证
代理身份验证或委托身份验证使得 CAS 变得更加强大，这是非常重要且能够提高安全性的功能代理身份验证受 CASv2 和 CASv3 协议支持，并由服务代表用户请求的代理票证进行中转。因此，服代理了用户的身份验证。代理身份验证通常用于服务无法直接与用户交互的情况，并且可以替代将最用户凭据重播到服务的方法。
但是，代理票证存在风险，因为接受代理票证的服务负责验证代理链（已委托最终用户的身份验以通过其到达票证验证服务的服务列表）。服务可以通过针对/ serviceValidate 验证端点验证票证来择完全不接受代理票证（并避免承担验证代理链的责任），但是经验表明，对此很容易感到困惑，并置为无意中使用了/ proxyValidate 端点，而不是检查票证验证响应中出现的所有代理链。因此，代身份验证需要仔细配置以进行适当的安全控制。如果不需要代理身份验证，建议在 CAS 服务器上禁代理身份验证组件。
从历史上看，任何服务都可以获取授权代理票证，并可以从中获得访问任何其他服务的代理票证换句话说，安全模型是分散的而不是集中的。服务管理工具通过公开可以基于每个服务启用或禁用的理身份验证标志，提供对代理身份验证的一些集中控制。默认情况下，不授予注册的服务代理身份验功能。
凭证缓存和重播
ClearPass 扩展提供了一种机制来捕捉主认证证书，并缓存他们（加密保存），并据需要访问遗留服务按需重播。虽然 建议使用代理身份验证代替密码重播，但可能需要将旧服务与 CAS 集成。有关详细信息，请见 ClearPass 文档。
服务管理
服务管理工具提供了许多特定于服务的配置控件，这些控件会影响安全策略并为集中式安全策略

供一些支持。（请注意，CAS 一直以来都支持分散式安全策略模型。）服务管理控件的一些要点：</
>

授权服务
强制认证
属性发布
代理验证控制
主题控制
服务授权控制
多因素服务访问策略

服务管理工具由包含一个或多个注册服务的服务注册表组成，每个服务都指定上述管理控件。可通过静态配置文件，Web 用户界面或两者来控制服务注册表。有关更多信息，请参见服务管理部分。
授权服务
作为安全性最佳实践，强烈建议将服务管理工具限制为仅包括被授权使用 AS 的已知应用程序列表。使管理界面对所有应用程序保持开放可能会给安全攻击创造机会。
SSO cookie 加密
<p>授予票证的 cookie 是 CAS 在建立单点登录会话时设置的 HTTP cookie。默认情况下，cookie 是通过 CAS 属性中定义的设置进行加密和签名的。尽管提供了用于初始部署的样本数据，但<strong必须根据您的特定环境重新生成这些密钥。请<a href="https://ld246.com/forward?goto=https%3A%2F%2Fgithub.com%2Fapereo%2Fcas%2Fblob%2Fmaster%2Fdocs%2Fcas-server-ocumentation%2Finstallation%2FConfiguring-SSO.html" target="_blank" rel="nofollow ugc"参阅本指南以获取更多信息。</p>
密码重置安全链接
帐户密码重置请求通过安全链接处理，该链接发送到用户的注册电子邮件地址。该链接仅在定义时间窗口内可用，并且请求已由 CAS 正确签名和加密。尽管提供了用于初始部署的样本数据，但<strng>必须根据您的特定环境重新生成这些密钥。
请参阅本指南以获取更多信息
协议票据加密
由 CAS 发行并与其他应用程序共享的协议票据（例如服务票据）可以选择进行签名/加密过程。使 CAS 服务器将始终交叉检查票证的有效性和过期策略，也可能会强制执行此检查，以确保传输到他应用程序的票证不会被篡改并保持真实性。尽管提供了用于初始部署的样本数据，但必须/strong>根据您的特定环境重新生成这些密钥。
请注意根据所使用的加密方法和算法，对生成的票证进行加密和签名会增生成的票证的长度。并非所有的 CAS 客户都具备处理冗长的机票字符串的能力，并且可能会让您不兴。在启用此功能之前，请先评估现有的集成，并考虑您的部署是否真正需要此功能。
要查看 CAS 属性的相关列表，请查看本指南。
票务注册表加密
对于集群式 CAS 部署，可能需要安全的票证复制，以确保 CAS 生成的票证不会在运输过程中被改。CAS 通过允许对票证进行本机加密和签名来解决此问题。尽管提供了用于初始部署的样本数据，必须根据您的特定环境重新生成这些密钥。请参阅本指南以获取更多信息。
管理页面安全

CAS 提供了许多针对系统管理员和部署人员的 Web 界面。这些屏幕以及许多 REST 端点使 CAS 部署者无需借助本机命令行界面即可管理和重新配置 CAS 行为。不用说，这些端点和屏幕必须受到护，并且仅允许经过授权的方正确访问。请参阅指南以获取更多信息。
票证过期策略
票证过期策略是实施安全策略的主要机制。凭单到期策略允许控制 CAS SSO 会话行为的一些重方面：

SSO 会话时长（滑动到期时间，绝对值）
票务重用

有关各种到期策略和配置说明的详细讨论，请参见“ 配置票证组件” 部分。
单一登出
单一注销或单一注销（SLO）是一项功能，通过该功能可以通知 CAS 服务 CAS SSO 会话的终止并期望服务终止 SSO 会话所有者的访问。尽管单点注销可以提高安全性，但从根本上说，这是一种力而为的功能，并且实际上可能不会终止对 SSO 会话期间消耗的所有服务的访问。以下补偿性控制施可用于改善与单一签出缺陷相关的风险：

需要对敏感服务进行强制身份验证
减少应用程序会话超时
减少 SSO 会话持续时间

SLO 可以通过两种方式发生：从 CAS 服务器（后通道注销）和/或从浏览器（前通道注销）。对反向通道注销，SLO 进程依赖于 <code>SimpleHttpClient</code> 具有线程池的类：必须定义其小才能正确处理所有注销请求。其他尚未处理的注销请求在发送之前被临时存储在队列中：其大小定为线程池全局容量的 20%，并且可以调整。两种大小都是 CAS 系统的关键设置，它们的值决不能超过 CAS 服务器的实际容量。
登录限制
CAS 支持策略驱动的功能，以限制连续失败的身份验证尝试，以帮助防止暴力破解和拒绝服务攻。该功能在后端身份验证存储缺少等效功能的环境中很有用。如果在基础系统中可以使用此支持，我鼓励使用它而不是 CAS 功能。理由是在底层系统中启用支持可在包括 CAS 在内的所有相关系统中提该功能。有关 更多信息，请参见 登录限制配置部分。
凭证保护
要了解如何通过加密保护敏感的 CAS 设置，请查看本指南。
CAS 安全过滤器
CAS 项目提供了许多简单的通用全筛选器，适用于就地修补 Java CAS 服务器和 Java CAS 客户端部署，这些部署容易受到基于些请求参数的不良 CAS 协议输入攻击的攻击。过滤器配置为清除身份验证请求参数，如果请求不符合 CAS 协议（例如，参数重复多次，包含多个值，包含不可接受的值等），则拒绝请求。
据强烈建议所有 CAS 部署进行评估，并包括该配置，如果需要防止在将 CS 容器和环境都无法阻止恶意和严重配置的请求的情况下协议的攻击。
CORS

CAS 为启用 HTTP 访问控制（CORS）提供了一流的支持。CORS 的一种应用是，当资源从与第个资源本身所服务的域不同的域中请求资源时，该资源发出跨域 HTTP 请求。通过 XHR / Ajax 请求问启用了 CAS 的应用程序，这应该会帮助更多。
要查看 CAS 属性的相关列表并调整此行为，请查看本指南。
安全响应头
作为 CAS 安全筛选器的一部分，CAS 项目自动提供必要的配置，以将 HTTP 安全标头插入 Web 响应中，以防止受到 HSTS，XSS，X-FRAME 和其他攻击。这些设置当前默认为启用。要查看 CAS 性的相关列表并调整此行为，请查看本指南。
要查看并了解有关这些选项的更多信息，请访问本指南。
Spring Webflow
CAS 项目使用 Spring Webflow 来管理和协调认证过程。CAS 使用的 Webflow 的会话状态由户端管理，然后在身份验证过程的各个状态中传递和跟踪。必须保护和加密此状态，以防止会话劫持虽然 CAS 提供了开箱即用的默认加密设置，但强烈建议在生产部署之前对所有 CAS 部署进行评估，并重新生成此配置以防受到攻击。
长期认证
通常在 CAS 登录表单上选择（通常通过复选框）长期身份验证功能，通常称为“记住我”，以免长时间重新进行身份验证。长期身份验证允许用户以降低的安全性为代价选择其他便利。安全性降的程度取决于用于建立 CAS SSO 会话的设备的特征。由单个用户拥有或操作的设备建立的长期 SSO 话比标准 CAS SSO 会话安全性稍差。唯一真正关心的是使用寿命的延长，以及 CAS 合格票的曝光率加。另一方面，从共享设备建立长期 CAS SSO 会话可能会大大降低安全性。来自先前 SSO 会话的伪影响由其他用户建立的后续 SSO 会话的可能性，即使面对一次注销，也可能会增加假冒的可能性。然没有可行的缓解措施来改善共享设备上的长期 SSO 会话的安全性，但对用户进行内在风险教育可提高整体安全性。
重要的是要注意，强制身份验证取代了长期身份验证，因此，如果将服务配置为强制身份验证，即使在长期会话的情况下，也需要身份验证才能访问服务。
在安装过程中，必须通过配置和 UI 定制显式启用长期身份验证支持 。因此，部署者选提供长期的身份验证支持，并且在可用时，用户可以通过在 CAS 登录表单上进行选择来选择使用它。/p>
警告
在已建立的 SSO 会话期间，CAS 支持服务访问的可选通知。默认情况下，CAS 透明地请求服务问所需的票证，并将其提供给目标服务以进行验证，从而在成功验证后就允许访问该服务。在大多数况下，这几乎是立即发生的，并且用户不知道访问启用了 CAS 的服务所需的 CAS 身份验证过程。意到此过程可能会带来一些安全方面的好处，并且 CAS 支持警告用户可以在 CAS 登录幕上选择的“标记”，以提供在访问服务之前显示的插页式通知页面。默认情况下，通知页面为用户供了进行 CAS 身份验证或退出目标服务而中止的选项。
<h4 id="升级指南">升级指南</h4>
通常，建议采用者尝试使他们的 CAS 部署与可用的最新 CAS 版本保持一致。特别是，具有 <code>PATCH</code> 或 <code>SECURITY</code> 性质的发行版应立即应用，因为它们是其相应版本的直接替代品。有关更多信息，请参见 CAS <a href="https://ld246.com/forward?goto=http%3A%2F%2Fgithub.com%2Fapereo%2Fcas%2Fblob%2Fmaster%2Fdocs%2Fdeveloper%2FRel

ase-Policy.html" target="_blank" rel="nofollow ugc">发布策略。
CAS 升级的总体目标可能是：

升级是否解决了严重的安全漏洞或烦人的问题？我的 CAS 部署受该漏洞和/或错误影响吗？
升级是否提供了对实现本地用例有用的功能？
升级是否提供了在我的叠加层中本地承载的功能，例如，通过消除这些本地更改，我可以直接从 AS 中受益，并最终得到一个较小的，更易于维护的 WAR overlays？

本文档试图以很高的层次描述升级给定 CS 部署所需的范围和工作 。我们没有描述审查和调整所需的所有步骤/更改（这是不可能的），是描述了可以执行升级的策略
<h5 id="变更记录">变更记录</h5>
尝试升级之前，请查看 CAS 更改日志，以确定要升级到的版本中包含哪些更改/修复，以及这些更改/修复是否适用于您的环境和 CAS 部署如果您使用的是较旧的 CAS 版本，并且遇到了似乎是错误的错误，则可以通过查看更改日志来找到盖层的替代品，以解决此问题。
<h5 id="讨论问题">讨论问题</h5>
查看了更改日志后，如果您看不到可以修复/调整您所想行为的改进，请在相应的 CAS 邮件列表讨论该问题。讨论的结果将是范围/努力评估，以确定解决方案的可行性以及将在其中进行修复的目版本。
<h5 id="范围审查">范围审查</h5>
一旦确定了理想的 CAS 版本，在尝试升级之前，请查看 CAS 发行政策。这将使了解新版本可能带来的更改以及升级所需的精力。
<h5 id="评估开发环境">评估开发环境</h5>
作为最佳实践，建议您通过 WAR overlay 方法部署 CAS 。如果有的话，这里的任务将是确定叠加层已触摸和修改的文件数。对应用更改内容和原因进行分类，然后使用 CAS 更改日志对这些更改进行交叉检查。可能是，由于升级，默认况下，CAS 会默认提供叠加层中存在的许多本地更改，这将使您在本地放弃许多这些改进。
您所做的更改通常是：

身份验证方案和策略（即 LDAP，JDBC 等）
在 CAS 属性文件中控制 CAS 行为的设置
用户界面更改可能包括 CSS 和 JavaScript
属性解析和发布策略
注册并授权使用 CAS 的服务

<h5 id="准备开发环境">准备开发环境</h5>
确保您已准备好用于配置和测试的单独开发环境。不管升级有多小，您都需要确保在您的环境中它进行了良好的测试，然后再进行切换。评估新升级的软件依赖关系和平台要求（例如 Java 等），确保在尝试安装之前正确安装和配置了所有组件。
<h5 id="清理配置">清理配置</h5>
我们建议您首先以针对您要升级到的版本的单独的干净 CAS WAR overlay 开始。这具有保证新 CAS 部署无需任何本地更改即可正常运的优势。一次构建和部署干净的 CAS 覆盖图，以确保您的构建/部署过程正常运行。
<h5 id="应用更改">应用更改</h5>

浏览在本地叠加层中找到的更改目录。将这些文件与原始版本进行比较和比较。您可以通过以下式找出两个版本之间的差异：

如果您一次构建了干净的 CAS 覆盖图，则通常会在 CAS 覆盖图的 <code>target</code> 或 <ode>build/libs</code> 目录中自动获得原始版本。在正确的路径中找到正确的文件并进行比较。</l>
直接转到项目源存储库，找到适当的分支并比较文。

不用说，您将需要：

不错的差异工具，例如 KDiff3，<a href="https://ld246com/forward?goto=http%3A%2F%2Fwinmerge.org%2F" target="_blank" rel="nofollow ugc"WinDiff，Beyond Compare 等。
体面的智能文本编辑器（例如 Sublime，Atom）或功能完善的 IDE（例如 IntelliJ IDEA）</a。

<h5 id="文件变更">文件变更</h5>
请记住要记录本地 WAR overlay 中存在的其余更改，以便下次执行相同的过程时，您可以了解AR overlay 的外观方式。
<h4 id="发行策略">发行策略</h4>
本章文档主要说明发行版本的工程师应采取的简化的 CAS 服务器发布版本的步骤。
<h5 id="Sonatype-账户设置">Sonatype 账户设置</h5>
您将需要注册一个 Sonatype 账户，并且必须要获得 org.apereo 创建 JIRA 的授权才能将 releas 版本发布到软件包中，拥有账号和授权后，还需要一名当前已属于项目成员的人为您做担保。
<h5 id="GPG设置">GPG 设置</h5>
您需要生成自己的 PGP 签名来对发布工件进行签名，然后再将其上传到中央存储库，为了创建pen PGP 签名，您需要生成一个密钥对，你需要向 build 提供以下信息：

公钥 ID（keyId 的最后 8 个符号，您可以通过 <code>gpg -K</code> 命令来获取它）
包含您的私钥的密钥环文件的绝对路径，从 gpg 2.1 开始，您需要使用以下命令导出密钥：

<pre><code class="highlight-chroma">gpg --keyring secring.gpg --export-secret-keys > ~/ .gnupg/secring.gpg
</code></pre>

使用密码保护您的私钥；
需要在 <code>~/.gradle/gradle.properties</code> 文件中添加以下设置：

<pre><code class="highlight-chroma">signing.keyId=7A24P9QB
signing.password
P@$$w0rd
signing.secretKeyRngFile=/Users/example/.gnupg/secring.gpg
</code></pre>
有关如何使用 Gradle 签名插件对工件进行签名的其他说明，<a href="https://ld246.com/forw

rd?goto=https%3A%2F%2Fdocs.gradle.org%2Fcurrent%2Fuserguide%2Fsigning_plugin.html" arget="_blank" rel="nofollow ugc">请参见此处。
<h5 id="环境设置">环境设置</h5>

设置您的 SSH 密钥，并确保 GitHub 中也引用了该 SSH 密钥；
使用 <code>$GRADLE_OPTS</code> 设置调整以初始化 JVM 堆大小；
在 <code>~/.gradle/gradle.properties</code> 添加以下配置：

<pre><code class="highlight-chroma">org.gradle.daemon = false
org.gradle.parallel = false
</code></pre>

签出 CAS 项目：<code>git clone git@github.com:apereo/cas.git cas-server</code>
确保安装了最新版 JDK 11：java -version;

<h5 id="准备发布">准备发布</h5>
应用以下步骤准备发布环境。根据是否应创建新的发行分支，需要考虑一些变体。
创建分支
<pre><code class="highlight-chroma"># 将$ BRANCH替换为CAS版本（即5.3.x）
git checkout -b $ RANCH
</code></pre>
提醒
您应该只对主要或次要版本（例如 <code>4.2.x</code>，<code>5.0.x</code>）。如果已经存要发布的版本的远程跟踪分支，则应使用 <code>git checkout</code> 签出该分支，否则跳过此骤，继续进行下一部分进行构建和发布。
<h5 id="GitHub-操作">GitHub 操作</h5>
提醒
创建新分支后，仅应针对主要版本或次要版本执行此操作。

更改 <code>.github/workflows/cas-build.yml</code> 为触发并仅构建新创建的发行分支。
检查 <code>ci</code> 文件夹下的所有 CI Shell 脚本，确保没有指向 <code>development</code> 或 <code>master</code> 分支。这尤其适用于如何将 CAS 文档发布到 <code>gh-pages</code> 的分支。
在 CI 中禁用报告新依赖性版本或使用 Renovate 等更新依赖性的作业。

不要忘了 commit 您所有的修改并 push 您的修改，并创建一个新的远程分支进行跟踪管理。
<h5 id="执行发布">执行发布</h5>

在项目的 gradle.properties 中修改项目的版本号为发布版本的版本号，并且删除 <code>-SNAPSHOT</code>，例如改为（6.0.0-RC1）

使用如下命令构建并发布项目：
<pre><code class="highlight-chroma">./release.sh
</code></pre>

登录：https://oss.sonatype.org.

在页面的左侧找到 CAS 发行的版本组件然后在列表的下方点击“Staged Repositories”;

通过工具栏的“close”按钮关闭仓库并添加描述信息，这一步或许会耗费您几分钟时间，但是些动作能够确保您所有的操作都是正确的；

通过工具栏的“Release”按钮发布仓库并添加描述，这一步仅仅是确认仓库是否成功被关闭；

<h5 id="释放版本">释放版本</h5>

使用发布的版本号创建一个标签（tag）,提交（commit）并推送（push）到远程仓库；
您还应该切换回主开发分支(即主开发分支)，并遵循以下步骤:

在项目的 <code>gradle.properties</code> 文件中修改项目的版本号为下一个开发版本号，如 5.0.0-SNAPSHOT;
推送（push）您的修改到远程仓库；

<h5 id="内务处理">内务处理</h5>

关闭项目版本的里程碑；
找到映射到已发布标记的版本并更新描述；提醒
当您更新发布版本的描述时，尽量保持一致，并遵循与以前版本相同的布局。
当发布项目的 RC 版本时，将该发布标记为预发布；
向 @cas-announce、@cas-user 和 @cas-dev 邮件列表发送通知消息，并链接到新的发布页。

<h5 id="更新Overlays">更新 Overlays</h5>
更新以下 Overlay 项目以指向新发布的 CAS 版本。对于下面的每个 Overlay 项目，您可能需要当前的主分支移到维护分支，特别是处理主/次版本发布时，以及如果发布过程让您创建了一个新的支时：

CAS WAR Overlay

<h5 id="更新文档">更新文档</h5>
请记住
只有在创建新分支时，主要或次要版本时才应该这样做：

配置文档以指向此处 <code>current</code> 的最新可用版本。
配置文档以将新版本包括在<a href="https://ld246.com/forward?goto=https%3A%2F%2Fgi

hub.com%2Fapereo%2Fcas%2Fblob%2Fgh-pages%2F_layouts%2Fdefault.html" target="_blan" rel="nofollow ugc">可用版本列表中。
更文档并添加新发布的版本。
如有必要，更新项目<code>README.md</code> 页面以列出新版本。
更新构建过程，以包括有关如何构建新版本的所有需信息。
更新发行说明并删除所有以前的条目。

<h5 id="更新维护记录文档">更新维护记录文档</h5>
更新维护策略以记录发布时间表和 EOL 时间线。仅当处理主要或次要版本时，此任务才有意义。/p>
<h5 id="更新演示">更新演示</h5>
（可选）现在许多 CAS 演示都在 Heroku 上运行，并在代码库内的专用分支中进行了跟踪，需在发布相关时进行更新。
<h4 id="发布时间表">发布时间表</h4>
请查阅：https://github.com/apereo/ca/milestones
<h4 id="发行说明">发行说明</h4>
<h4 id="维护方案">维护方案</h4>
<h3 id="安装部署">安装部署</h3>
<h4 id="基本要求">基本要求</h4>
<h4 id="WAR-Overlays-构建方法">WAR Overlays 构建方法</h4>
<h4 id="命令行工具">命令行工具</h4>
<h4 id="Docker-部署方案">Docker 部署方案</h4>
<h4 id="Servlet容器部署方案">Servlet 容器部署方案</h4>
<h4 id="操作系统服务">操作系统服务</h4>
<h4 id="问题处理指南">问题处理指南</h4>
<h3 id="配置">配置</h3>
<h4 id="概览">概览</h4>
<h4 id="配置服务器">配置服务器</h4>
<h4 id="常用属性">常用属性</h4>
<h4 id="属性和设置项">属性和设置项</h4>
<h4 id="安全配置">安全配置</h4>
<h4 id="元数据配置">元数据配置</h4>
<h4 id="扩展配置">扩展配置</h4>
<h4 id="重新加载更改">重新加载更改</h4>
<h4 id="集群部署">集群部署</h4>
<h4 id="配置发现">配置发现</h4>
<h4 id="配置表达式">配置表达式</h4>
<h3 id="认证方式">认证方式</h3>
<h4 id="总览">总览</h4>
<h4 id="认证方式-">认证方式</h4>
<h5 id="LDAP">LDAP</h5>

<h6 id="密码策略">密码策略</h6>
<h5 id="数据库">数据库</h5>
<h6 id="密码策略-">密码策略</h6>
<h2 id="学习笔记">学习笔记</h2>