01.总体设计文档

1. 引言

1.1. 目的

简要介绍该说明书的目的，和使用该规格书的对象(如项目技术经理等)。

总设的输入：系统需求文档；在文档编写之前，画出一个设计的大体思路，经专家评定后，再动手写，也是一个输入。

总设的输出：模块的划分、模块间的接口、模块间的系统流程

需求文档里面有的需求，都要在总体设计里面有体现。回溯责任时，如果需求文档里面有，总体设计没体现，则是总体设计人员的责任。

本版本承接“一朵云”整体战略中的中小ISV SaaS化方向，本期需求以ISV 传统单体应用转SaaS化为主场景，通过帮助ISV快速将传统单体软件一键完成SaaS化、提供SaaS应用完备的访问安全能力、提供高性价比的自动化多租户数据库等能力，为垂直行业中小规模ISV在saas化转型中提供完整的能力支持。通过POC版本优先完成上述能力中的关键竞争力构建，快速进行种子用户产品验证。

1.2. 定义和缩写

定义或解释本文档中使用的所有专用名词和缩略语。

缩写名	描述	备注

SCP	信服云管理平台	
SaaSBoost	SaaS应用加速器，用于快速构建SaaS应用的平台能力，部署在SCP	
VPC	Virtual Private Cloud(虚拟专用网络)，托管云上逻辑隔离的网络环境。	
SASA	SaaSApplicationSecurityAndAccess(SaaS应用安全与访问系统)，包含AD、Atrust、IAM、PostgreSQL、SASA-API、Bifrost、DB-PROXY、WAF等组件，部署在ISV-VPC内。	
AD	应用交付，能够为ISV提供包括多租户应用统一访问入口、TLS卸载、服务器负载均衡的能力。	
Atrust	深信服零信任访问控制系统，是深信服基于零信任安全理念推出的一款以“流量身份化”和“动态自适应访问控制“为核心的创新安全产品。为ISV提供统一的应用隔离、安全访问等能力。	
SBP	SaaSBoostPlatform(SaaS应用加速器平台)，部署在ISV-VPC内，是承载SaaSBoost管理请求并配置SASA所有组件的服务，包含SASA-API、IAM、PostgreSQL、Bifrost等组件，	
SASA-API	SBP的API服务，部署在K3S宿主机内，外部由APISIX做访问安全与统一代理	
APISIX	是一个动态、实时、高性能的云原生 API 网关，提供负载均衡、身份验证、限流等功能，支持插件热加载和多种协议。	
IAM	Identity and Access Management(认证与访问管理)，提供统一的用户管理、身份认证、单点登录等功能。	
PostgreSQL	IAM的存储数据库，是一个功能强大、开源的对象关系型数据库管理系统（ORDBMS），以其高性能、稳定性和丰富的功能集著称，支持标准的 SQL 语言。	
Bifrost	通过监听ISV的单体应用数据库的Binlog，当用户信息增、删、改时，及时同步到IAM和Atrust中。	
DB-PROXY	提供单体应用访问数据库的统一代理。映射关系为 云主机IPS 统一前缀 。	
企业用户	ISV在SaaSBoost中创建的用户，对应ISV提供服务的企业的管理员。	
终端用户/普通用户	企业用户(企业管理员)在使用单体应用过程中创建出来的用户，记录在SBP和Atrust中，SaaSBoost不感知。	

1.3. 参考和引用

列出本文档编写过程中参考的其它文档或资料

如果现有规范无法覆盖需求，需要组织干系方讨论需求，提取新的规范接口并评审，参考流程：http://code.sangfor.org/sangfor/specification

交互地址：https://seerdesignmg.sangfor.com/file/1353648083359 23?fileOpenFrom=home&page_id=306%3A73015&devMode=true

需求地址：

1.4. 总设任务书

此总体设计任务书由部分开发经理/产品架构师填写下发，说明设计需要达到的要求

编号	目标项概述	对应的标准要求
1	性能参数要求	SaaSBoost服务API单个接口耗时。GET请求小于200ms，非GET请求小于1S。 SaaSBoost最大支持100个ISV，5000个企业用户，5000个应用。(1个ISV50个企业用户，50个应用) Atrust、IAM支持20000个用户(包含企业用户+终端用户)。账号密码认证要在3S内完成，企业用户&终端用户尝试登陆到弹出IAM认证页面耗时5S内，企业用户&终端用户输入密码到进入单体应用5S内(需要去除单体应用的回调接口的耗时)。 POC阶段：不对SaaSBoost的接口耗时有要求。支持5-10个ISV，每个ISV2个企业用户，每个企业用户5个终端用户。账号密码认证要在3S内完成，企业用户尝试登陆到弹出IAM认证页面耗时5S内，企业用户输入密码到进入单体应用5S内(需要去除单体应用的回调接口的耗时)。引入SASA后单体应用的API耗时增加不能超过50%。
2	资源开销要求	*说明设计需要满足的内存占用、CPU占用、磁盘占用、磁盘操作频繁程度、文件描述符、网络传输数据量资源等要求* SaaSBoost服务一般业务场景：cpu < 0.5 c，内存占用 < 500MB。数据保存在MySQL，同步至Mongo，满足业务指标即可，不对MySQL和Mongo磁盘占用做约束。 注：一般业务场景：每秒一次查询请求，同时每分钟一次应用创建或用户创建的请求。 IAM服务一般业务场景：cpu < 0.5c，内存占用 < 500MB；宿主机的CPU和内存占用均不能超过80%。不对PostgreSQL、Bifrost的CPU、内存、磁盘占用有单独要求。 注： 一般业务场景，每分钟3个认证请求+单点登录整个流程。

<table>
<tr>
<td>3</td>
<td>稳定性</td>
<td>描述设计需要达到的稳定性要求，比如崩溃恢复时间、最高压力时平均可持续运行时间、需要检测程序是否崩溃出错手段等要求

服务进程：SaaSBoost 设置在失败时自动重启；SASA-AP设置在失败时自动重启；IAM、PostgreSQL、Bifrost服务需要配置Readness(就绪探针)、Liveness(存活探针)探测程序，容器故障需要重新拉起。

SaaSBoost的service添加如下配置：
Restart=on-failure

<table>
<tr><th>容器名称</th><th>readinessProbe</th><th>livenessProbe</th></tr>
<tr><td>IAM</td><td>periodSeconds：10s

timeoutSeconds：15s</td><td>periodSeconds：30s

timeoutSeconds：15s</td></tr>
<tr><td>PostgreSQL</td><td>periodSeconds：10s

timeoutSeconds：15s</td><td>periodSeconds：30s

timeoutSeconds：15s</td></tr>
<tr><td>Bifrost</td><td>periodSeconds：10s

timeoutSeconds：15s</td><td>periodSeconds：30s

timeoutSeconds：15s</td></tr>
</table>

</td>
</tr>
</table>

4	架构要求	*描述总体架构设计要达到的要求，比如低耦合、可扩展等方面的具体要求，需要给出具体的要求和建议*
		
		• SaaSBoost用systemed管理，开机启动，phoenix框架
		  • 使用HTTP协议，统一使用APISIX代理，路由统一使用apisix help方式注册
		  • 定位为web服务，读写分离，写：MySQL；读：Mongo
		
		• AD在SCP上以开通应用交付的形式部署，承载所有SASA组件的外部流量，绑定弹性IP，进行统一的TLS卸载，并将流量统一转发给Atrust。
		
		• AtrustPOC阶段以综合网关形式部署，通过OpenAPI或者页面操作Atrust，对外暴露HTTP80接口，只承接AD转发过来的流量，并转发到其他SASA应用或客户的单体应用。
		
		• IAM部署在SKYOPS宿主机内，K3S管理，容器方式运行，无状态服务，满足云原生开发方式：
		  • 使用HTTP协议，为了保证控制面的性能，使用主机网络
		  • 内部服务之间服务发现使用K8S的service和endpoint
		
		• Postgresql部署在SKYOPS宿主机内，K3S管理，容器方式运行，满足云原生开发方式：
		  • 有状态服务，单节点部署，通过PVC挂载本地文件系统
		  • 内部服务之间服务发现使用K8S的service和endpoint
		• Bifrost部署在SKYOPS宿主机内，K3S管理，容器方式运行，满足云原生开发方式：
		  • 有状态服务，单节点部署，通过PVC挂载本地文件系统
		  • 内部服务之间服务发现使用K8S的service和endpoint
		
		• 低耦合方面
		  • SASA组件可以被SaaSBoost和SASA-API统一管理，SaaSBoost和SASA-API管理侧挂掉不应影响SASA组件的工作
		  • SaaSBoost与SASA组件采用标准的应用模型交互协议，不感知单体应用的业务逻辑

5	可维护性	*提供给维护人员的工具或手段* • SaaSBoost进程保证其只与业务有关，单独提供debug镜像携带各类调试工具如：网络io、抓包、trace等，使用rpdb进行排障 • SaaSBoost服务日志满足云BG日志规范，日志需要添加错误码，排障码，支持USR1信号触发DEBUG模式

6	可测试性	*说明系统给功能、压力、性能测试及自动化测试提供的机制和方法。比如提供统一的模块内部状态获取及调试的接口，制订调试接口规范。*
		
		• 可调试性
		  • SaaSBoost服务提供cli工具，以快速进行接口调试
		  • SaaSBoost服务使用标准的HTTP协议，支持curl快速调试，同时可以使用postman、Jmter等工具进行调试
		
		• 可操作性
		  • SaaSBoost服务提供API接口，采用swagger2.0和openapi 3.0声明的yaml格式定义
		
		• 可分解性
		  • 模块应该以微服务架构为主，通过标准API进行调用，方便mock测试
		  • 封装对环境的依赖，通过mock及环境模拟机制可以独立运行
		
		• 性能、压力测试
		  • 通过脚本模拟终端用户的创建，压测数据同步组件Bifrost的性能
		  • 可以通过JMeter进行SaaSBoostAPI压力测试、SASA-API、IAM、Atrust的压力测试

7	可扩展性	*说明系统在哪些方面需要保留可扩展的能力，并指明扩展方式* • IAMPOC阶段支持单体应用以Bcrypt、md5、md5plain加密方式形式储存密码，后续支持其他形式的加密方式 • Bifrost支持新增插件，用于异构数据结构的用户信息处理与同步。
8	兼容性	*允许升级配置的版本、允许对接的版本* • NA
9	升级	*对升级场景进行额外说明* • IAM服务和Bifrost服务部署在客户VPC网络中，采用灰度升级的形式，支持热升级，SaaSBoost中心端负责将新版本推送至harbor制品仓库，由IAM宿主机(Skyops)升级服务负责统一升级，POC阶段不要求。

10	安全性	*设备具备的防范用户恶意攻击的能力（如无防范，恶意攻击可能导致本设备宕机、拒绝服务）。* • 业务容器不允许以root权限启动 • SaaSBoost管理服务和部署在客户VPC内的SASA组件全部采用TLS加密，并通过互信机制进行认证 • SaaSBoost管理服务对外提供客户服务，所有请求需要通过平台互信认证与权限校验 • IAM和Bifrost组件POC阶段会暴露管理面到公网，必须经过互信认证和TLS加密。后续收敛，由SASA-API统一管理。
11	可重用性	*当前版本可采用的公用代码、可产出的公用代码要求版本经理和开发经理、预研部门一起识别和定义产出。* • NA
11	可分析定位能力	• 所有的异常均有排障码辅助快速定位问题。 • 不可出现系统异常等笼统性日志。
12	其它要求	IAM主要开发语言：go1.23

2. 产品介绍

2.1. 产品名称、型号、代号

ISV-SaaS化1.0

2.2. 版本规划

迭代时间	24.11.8	1230
迭代方向	POC版本	正式版
迭代目标	具备主场景saas化方案整体价值的基本能力，通过方案的完整性输出形成竞争力	NA
关键价值	（1）帮助交付传统单体应用的ISV一键完成初步多租户、多租户安全访问应用的业务saas化能力，初步具备saas化交付形态，商业化 （2）联动高性价比的多租户数据库、共享应用内部和应用访问安全组件（同VPC下）等形成高性价的SaaS方案吸引力 （3）高安全的应用防护组件能力，构建saas化应用上云改造驱动力 （4）资源隔离、访问隔离形成行业应用saas化首选方案	NA

2.3. 对应需求文档

列举本总体技术方案对应的需求文档

触点	客户场景
应用轻量SaaS化改造	面向目标客群ISV解决应用快速SaaS化能力，包括多租户能力、应用安全访问网关、以及关联应用承载面虚机、多租户数据库等能力
多租户数据库	在ISV SaaS化改造场景下提供自动化的多租户数据库中间件对接能力，从而帮助isv saas化后的业务 可以按租户实际使用情况灵活进行分库分表的策略配置
应用安全组件能力支持	针对将传统应用改造成轻量saas的isv，提供访问安全必须的组件，如waf、rasp等

2.4. 系统性能指标

类似于以下表格的系统性能指标

参考：

领域	指标	指标说明	备注	是否对外	迭代指标
见1.4章节性能参数要求					

2.5. 关键应用场景

描述系统需求里定义的整体用户场景，说明哪些是影响系统设计的关键用户场景。在方案选型时需保证方案能满足这些用户场景。

2.6. 关键技术及解决途径

描述系统所使用的关键技术，比如加速的流缓存算法，这里并不需要描述完整的关键技术，只需要概要说明，但要列举相关资料

3. 硬件方案

描述系统的硬件构成，描述支持的基本硬件平台以及所采用的其他功能芯片(比如压缩，加密等芯片。

4. 软件方案

4.1. 方案选型

4.1.1. SaaSboost 组件选型

对比aws、ali

4.1.2. IAM组件选型

对比keyclock+Zitadel+cloudpod

4.1.3. IAM认证方案选型

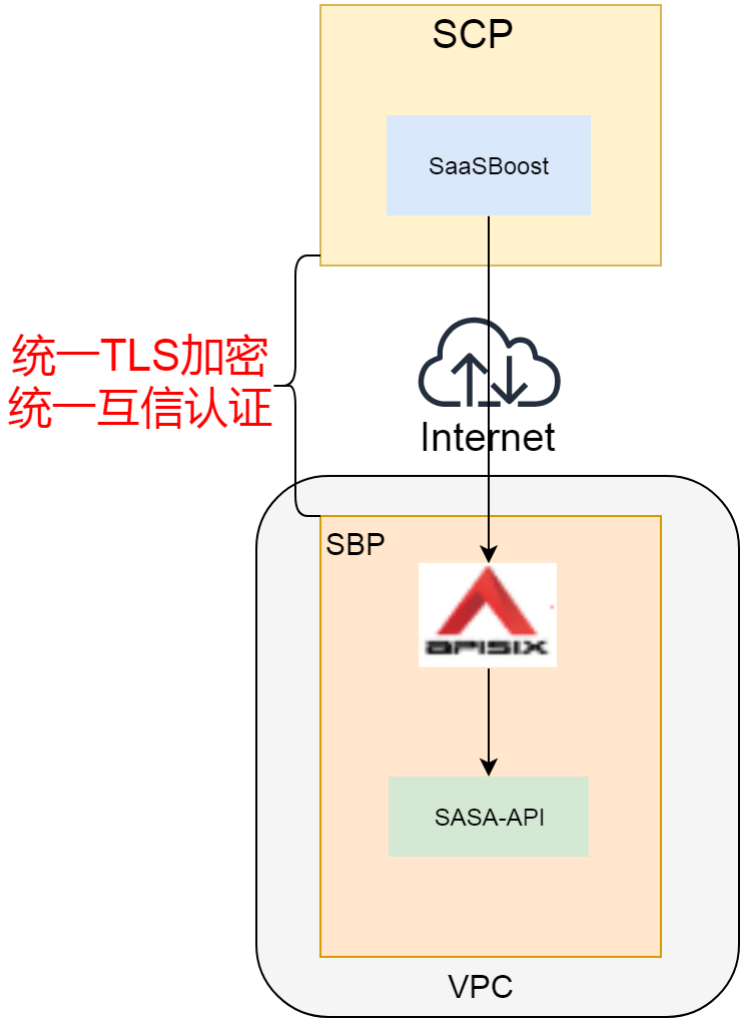
不同单体应用的密码加密方式可能是不同的，且不可反向解密，IAM要提供统一的密码认证方式，要支持常用的Bcrypt、Scrypt、md5、SHA256等。

序号	方案描述	优缺点	最终选择
方案一	一个用户配置一个密码加密方式，用户认证时先获取密码加密方式，然后再实例化hasher进行密码对比。工作量5d。	优点：性能较好，可以支持自定义参数的加密算法，比如指定Scrypt密钥的块大小、并行度和盐大小。 缺点：实现起来较为复杂，	
方案二	IAM支持各种加密算法的hasher，用户认证时循环对比，一种对比通过即为通过，全部不通过即为不通过。工作量0d。	优点：时间起来较为简单 缺点：不支持自定义参数的加密算法。	

POC阶段先实现方案二，后续往方案一演进。

4.1.4. SaaSBost与SBP互信方案(TLS层和应用层安全)

SBP密钥由APISIX颁发



密钥保存方案选型

序号	方案描述	优缺点	最终选择
方案一	SBP启动时生成一个统一的密钥，SaaSBoost升级脚本写入	优点：实现起来较为简单 缺点：安全问题，不同ISV的SBP密钥相同。密钥由SaaSBoost统一管理，ISV感知不到，且访问SBP需要校验源IP，只有密钥也无法访问。	

方案二	SBP部署后页面生成，SaaSBoost管理SBP的密钥、host等信息。	优点：安全性较高 缺点：需要SBP新开页面，且SaaSBoost要提供SBP管理的功能(需要与VPC绑定)，工作量较大10d。	

POC阶段先实现方案一，后续往方案二演进。

互信密钥算法方案选型

序号	备选方案名称	本方案的优点	本方案的风险和缺点	最终选择
方案一	key-auth	1、实现较为简单	1、对于安全性要求较高的场景，API Key 无法提供足够的安全保障 2、存在重放风险 3、api-key写死不够灵活	
方案二	jwt-auth		1、对于客户端来说，流程实现较为冗余，所有请求需要先调用颁发Token的接口，维护有效Token给客户端引入额外工作量 2、一旦颁发了 JWT，除非过期时间到达，否则无法撤销，可能会增加一定的安全风险 3、一般不用于服务间通信	
方案三	hmac-auth	1、lua实现，可以实现并发处理 2、可以有效防重放，有效防止数据篡改	1、客户端需要封装签名、服务端验证签名，实现起来稍许复杂，性能忽略不计 2、要保证客户端和服务端时间差异不能太大	

POC阶段先实现方案一，后续往方案三演进。

备注：如何配置以上三种认证算法

方案一：key-auth

Key-Auth 方案通常用于身份验证，其中客户端需要在请求中携带一个特定的密钥（key）作为身份验证凭据。请求在APISIX中针对该密钥进行验证，密钥信息储存在APISIX-Consumer中。在 HTTP客户端 中，可以通过添加自定义的头部来实现 Key-Auth 方案。

客户端示例：

```
package main

import (
"context"
"google.golang.org/grpc"
"google.golang.org/grpc/metadata"
"time"
)

func main() {
```

```
// 创建 gRPC 连接

conn, err := grpc.Dial("localhost:50051", grpc.WithInsecure())

if err != nil {

panic(err)

}

defer conn.Close()


// 创建一个新的上下文

md := metadata.Pairs("Authorization", "1234567890")

ctx := metadata.NewOutgoingContext(context.Background(), md)


// 创建 gRPC 客户端

client := httpclient(conn)


// 发起 gRPC 请求

response, err := client.YourMethod(ctx, &request)

if err != nil {

panic(err)

}


// 处理响应

// ...

}
```


APISIX配置示例：

①创建consumer

```
curl http://127.0.0.1:9180/apisix/admin/consumers \
-H "X-API-KEY: $admin_key" -X PUT -d '
{
"username": "test",
"plugins": {
    "key-auth": {
        "key": "1234567890"
    }
}
}'
```

②创建route

```
curl http://127.0.0.1:9180/apisix/admin/routes/test -H "X-API-KEY: $admin_key" -X PUT -d '
{
"methods": ["POST"],
"uri": "/sasa/*",

"plugins": {
"key-auth": {

    "header": "Authorization"
```

```
}
},
"upstream": {
"type": "roundrobin",
"nodes": {
    "sasa-api:26500": 1

},

"schema": "http"

}
}'
```

③请求示例：

```
curl http:10.132.48.88:9080/sasa/users  -H 'Authorization: 1234567890' -XPOST
```

方案二：jwt-auth

jwt-auth方案通常用于身份验证，被用于在用户登录后生成一个安全的令牌，客户端在后续的请求中携带该令牌以进行身份验证。其中客户端需要通过APISIX提供的public-api申请Token，后续请求携带该Token以完成身份认证。用于申请Token的密钥信息储存在APISIX-Consumer中。

APISIX配置示例：

①创建consumer

```
curl http://127.0.0.1:9180/apisix/admin/consumers \
-H "X-API-KEY: $admin_key" -X PUT -d '
{
"username": "test",
"plugins": {
"jwt-auth": {
    "key": "user-key",
    "secret": "my-secret-key"
}
}
}'
```

②创建route

```
curl http://127.0.0.1:9180/apisix/admin/routes/test -H "X-API-KEY: $admin_key" -X PUT -d '
{
"methods": ["POST"],
"uri": "/sasa/*",
"plugins": {
    "jwt-auth": {}

},
"upstream": {
"type": "roundrobin",
"nodes": {
    "sasa-api:26500": 1

},

"schema": "http"

}
}'
```

③创建签发 token 的 API

```
curl http://127.0.0.1:9180/apisix/admin/routes/jas \
-H "X-API-KEY: $admin_key" -X PUT -d '
{
"uri": "/apisix/plugin/jwt/sign",
"plugins": {
```

```
    "public-api": {}
}
}'
```

④获取Token

```
curl http://127.0.0.1:9080/apisix/plugin/jwt/sign?key=user-key -i
```

⑤请求示例

```
curl http:10.132.48.88:9080/sasa/users  -H 'Authorization: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJrZXkiOiJ1c2VyLWtleSIsImV4cCI6MTU2NDA1MDgxMX0.Us8zh_4VjJXF-TmR5f8cif8mBU7SuefPlpxhH0jbPVI' -XPOST
```

方案三：hmac-auth

HMAC（Hash-based Message Authentication Code）是一种基于哈希函数的消息认证码算法，用于验证消息的完整性和真实性。在网络通信中，HMAC 可以用于身份验证，确保消息在传输过程中没有被篡改。

HMAC 认证通常涉及以下几个步骤：

1. 生成密钥：双方事先共享一个密钥，用于生成和验证 HMAC。
2. 消息摘要：发送方使用密钥和消息内容计算出 HMAC，并将其附加到消息中。
3. 消息传输：发送方将消息和 HMAC 一起发送给接收方。
4. 验证：接收方使用相同的密钥、接收到的消息内容以及接收到的 HMAC 来重新计算 HMAC，并与接收到的 HMAC 进行比较，以验证消息的完整性和真实性。

HMAC 认证可以用于各种网络通信协议，包括 HTTP、WebSocket、gRPC 等。在实际应用中，HMAC 认证通常需要双方事先共享密钥，并且需要对消息内容进行适当的编码和处理，以确保在计算 HMAC 时不会出现歧义或安全漏洞。

APISIX原生支持hmac认证算法。

APISIX配置示例：

①创建consumer

```
curl http://127.0.0.1:9180/apisix/admin/consumers \
-H "X-API-KEY: $admin_key" -X PUT -d '
{
"username": "jack",
"plugins": {
    "hmac-auth": {
        "access_key": "user-key",
        "secret_key": "my-secret-key",
        "signed_headers": ["User-Agent", "Accept-Language", "x-custom-header"],

        "validate_request_body": true

        }
    }
}'
```

②创建route

```
curl http://127.0.0.1:9180/apisix/admin/routes/test -H "X-API-KEY: $admin_key" -X PUT -d '
{
"methods": ["POST"],
"uri": "/sasa/*",
"plugins": {
    "hmac-auth": {}

},
"upstream": {
    "type": "roundrobin",
    "nodes": {
        "sasa-api:26500": 1

    },
```

```
    "schema": "grpc"
}
}'
```

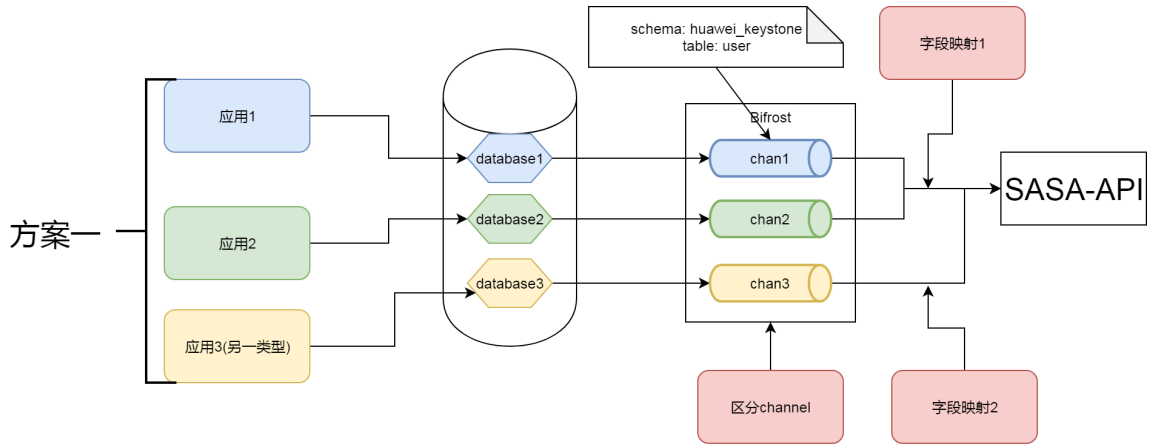
③生成签名

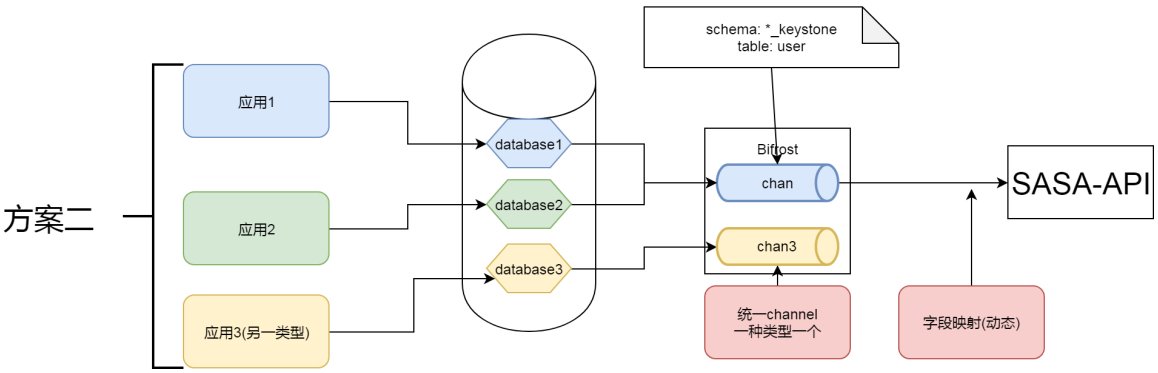
```
import base64
import hashlib
import hmac

secret = bytes('my-secret-key', 'utf-8')
message = bytes("""GET
/index.html
age=36&name=james
user-key
Tue, 19 Jan 2021 11:33:20 GMT
User-Agent:curl/7.29.0
x-custom-a:test
""", 'utf-8')

hash = hmac.new(secret, message, hashlib.sha256)


# to lowercase base64
print(base64.b64encode(hash.digest()))
```

④请求示例

```
curl http:10.132.48.88:9080/sasa/users -XPOST -H "X-HMAC-SIGNATURE: 8XV1GB7Tq23OJcoz6wjqTs4ZLxr9DiLoY4PxzScWGYg=" \
-H "X-HMAC-ALGORITHM: hmac-sha256" \
-H "X-HMAC-ACCESS-KEY: user-key" \
-H "Date: Tue, 19 Jan 2021 11:33:20 GMT" \
-H "X-HMAC-SIGNED-HEADERS: User-Agent;x-custom-a" \
-H "x-custom-a: test"
```

4.1.5. Bifrost同步配置如何下发





以下方案均需要Bifrost的HTTP插件支持字段映射、自定义头部、自定义URL、自定义请求体的功能，工作量3天

序号	方案描述	优缺点	最终选择
方案一	一个应用配置一个通道，SaaSBoost新增应用时，自动下发配置到Bifrost中，使用baseauth的形式调用API，或者进到Bifrost手动配置。	优点： 缺点：编码工作量新增2或0天	
方案二	一种类型的应用统一一个通道。Bifrost监听binlog的schema字段需要支持正则	优点：一种类型的应用只需要配置一次通道即可 缺点：编码工作量新增5天	

POC阶段选用方案一的手动配置的方式(每新增一个应用配置一次)，后续再支持自动化

4.1.6. SASA-API如何保存IAM的密钥信息

序号	方案描述	优缺点	最终选择
方案一	IAM启动时生成一个统一的密钥，sasa-api升级脚本写入	优点：实现起来较为简单 缺点：安全问题，不同ISV的IAM密钥相同。密钥由SASA-API统一管理，ISV感知不到，且后续会将IAM收敛暴露面。	
方案二	技服进行IAM部署时生成，通过sasa cli写入	优点：安全性稍高 缺点：需要部署时技服手动操作	

选用方案一

4.1.7. SASA-API如何保存Atrust的密钥信息

选型背景：Atrust密钥信息只能在UI生成

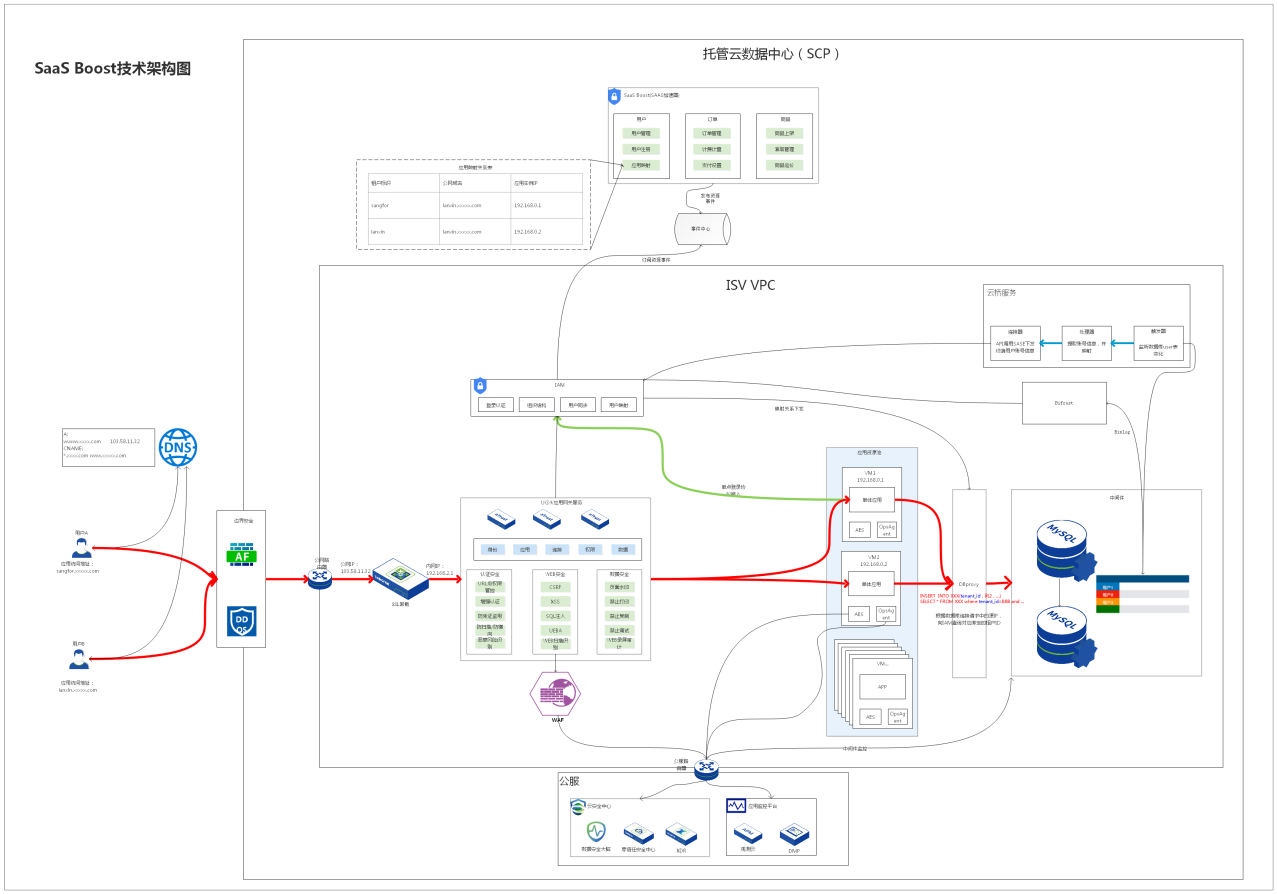
序号	方案描述	优缺点	最终选择
1	登录Atrust生成一个AppID和APPSecret之后，创建新的VMA，SASA-API升级脚本写入，每个Atrust的AppId和AppSecret都一样	优点：实现起来较为简单 缺点：新建了Atrust镜像，后续升级无法处理	
2	技服进行Atrust部署时手动生成，配置到SASA-API中，可以通过UI或脚本实现	优点：利于后续演进 缺点：需要部署时技服手动操作，需要提供API或者CLi，存在工作量	

POC阶段选用方案二，后续考虑自动化和动态生成

4.2. 软件总体架构

系统的静态视图，重点是系统的软件模块构成，包括模块名、功能说明以及模块之间的层次(依赖)关系。在系统设计之初(在分析系统流程图之前),我们根据需求和经验初步划分软件功能模块，以及模块的依赖关系和通信模型，然后分析 系统流程图，不断的修正系统的模块划分、模块的依赖关系和通信模型，最终得出模块的接口说明文档，而模块以及模块的接口说明文档，是概要设计阶段的工作入口和依据。

4.2.1.总体部署架构图



架构约束：

1、SaaSBoost部署在SCP，拥有用户管理和应用管理等能力，并负责管理该数据中心下的所有VPC内的SASA (SaaSApplicationSecurityAndAccess)组件

2、SASA包含AD、Atrust、WAF、AES、DB-Proxy、DB等组件

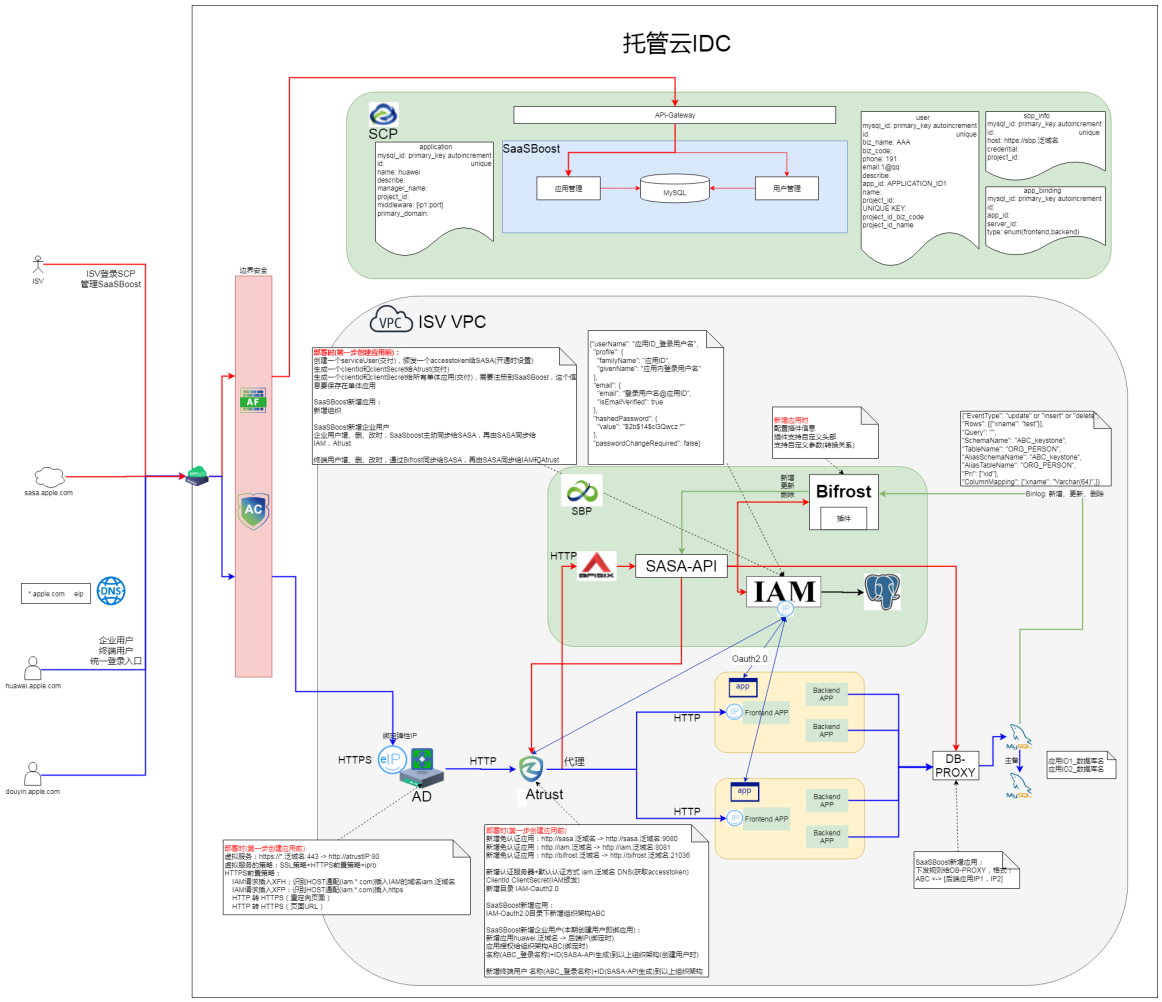
3、AD绑定弹性公网IP，对外暴露HTTPS443端口，负责统一的SSL卸载，并可以将所有请求转化为HTTP转发给Atrust，并能进行特定的头部改写

4、Atrust可以将访问安全能力赋予ISV的所有单体应用，可以将不同企业用户的请求代理到对应的单体应用，防止跨应用访问，可以默认跳转IAM进行认证

5、IAM部署在单独宿主机中，可以作为集中用户管理中台，支持所有单体应用的用户都通过统一的IAM进行认证登录，支持作为Atrust的认证服务器和所有单体应用的认证服务器

6、DB-Proxy作为单体应用的数据库访问网关，支持承载不同的单体应用的数据库请求

4.2.2.技术架构图



架构约束：

1、SaaSBoost部署在SCP，拥有用户管理和应用管理等能力，并负责管理该数据中心下的所有VPC内的SBP组件，进而管理所有SASA(SaaSApplicationSecurityAndAccess)组件

2、SASA组件包含AD、Atrust、WAF、AES、SBP、DB-Proxy、DB等组件

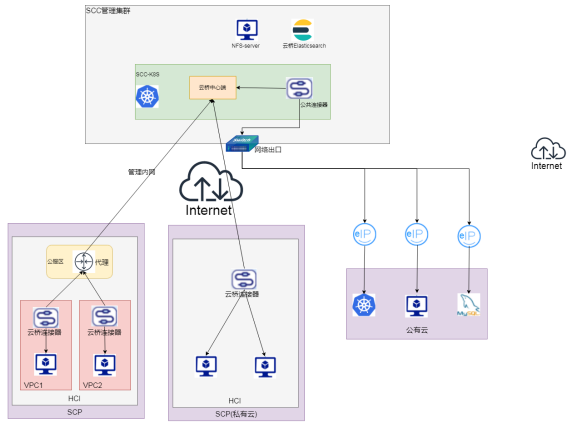
3、AD绑定弹性公网IP，对外暴露HTTPS443端口，负责统一的SSL卸载，并可以将所有请求转化为HTTP转发给Atrust，并能进行特定的头部改写

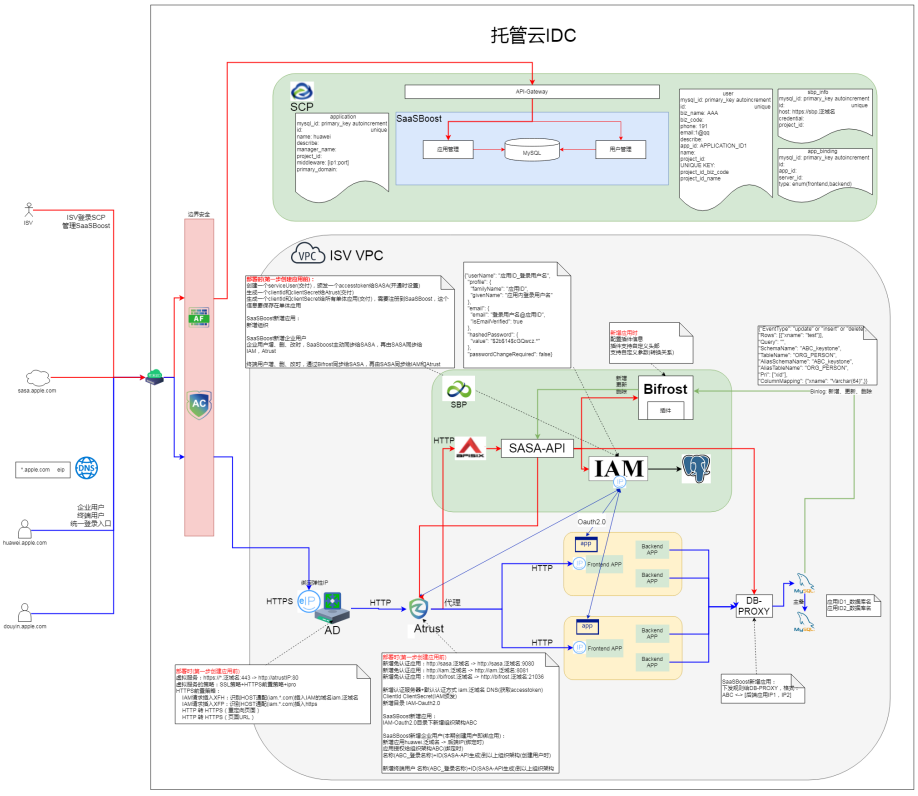
4、Atrust可以将访问安全能力赋予ISV的所有单体应用，可以将不同企业用户的请求代理到对应的单体应用，防止跨应用访问，可以默认跳转IAM进行认证

5、SBP部署了K3S，包含APISIX、SASA-API、IAM、Bifrost、PostgreSQL等组件，APISIX作为唯一对外暴露面

6、IAM docker形式部署，可以作为集中用户管理中台，支持所有单体应用的用户都通过统一的IAM进行认证登录，支持作为Atrust的认证服务器和所有单体应用的认证服务器

7、DB-Proxy作为单体应用的数据库访问网关，支持承载不同的单体应用的数据库请求

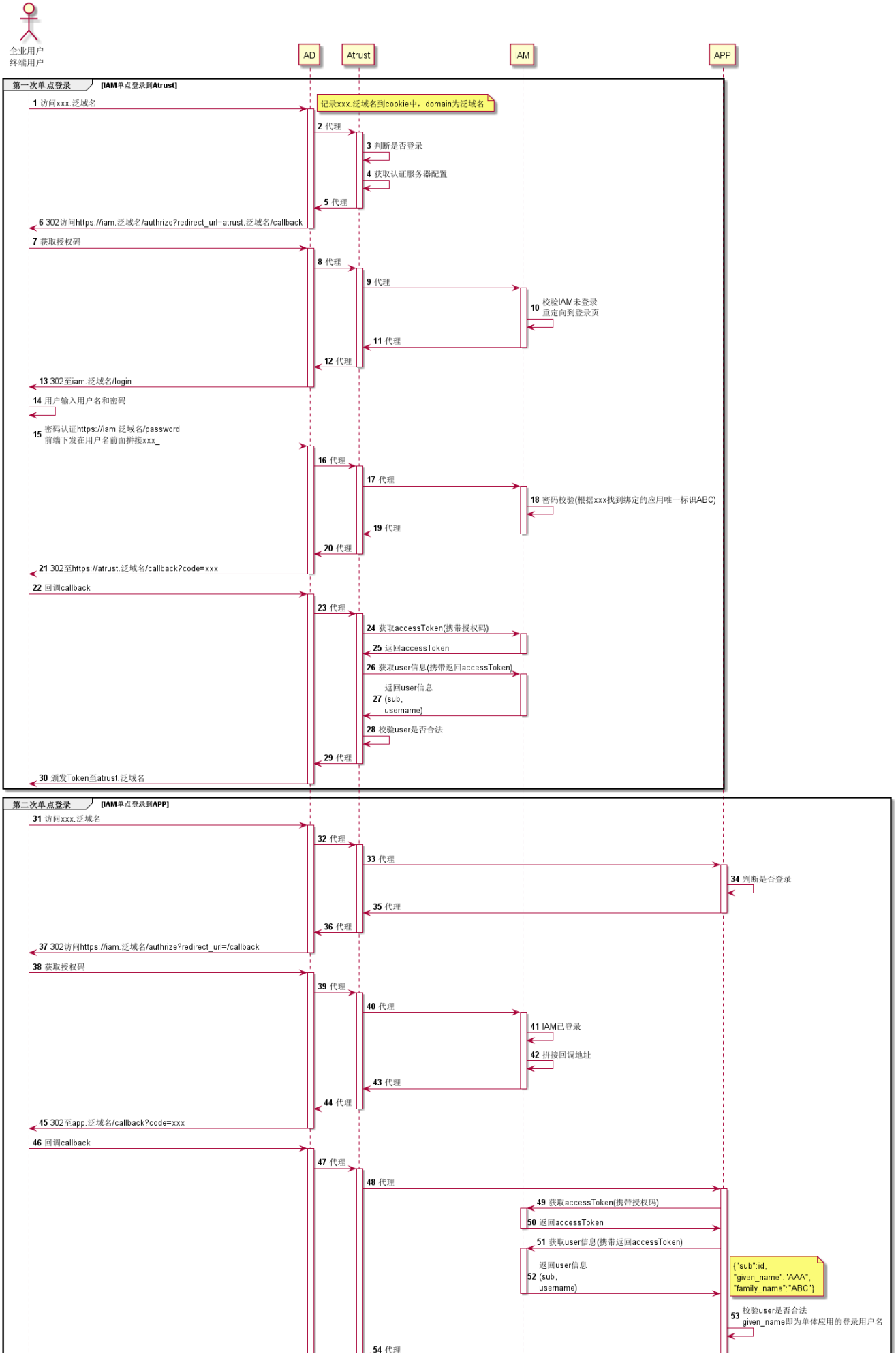


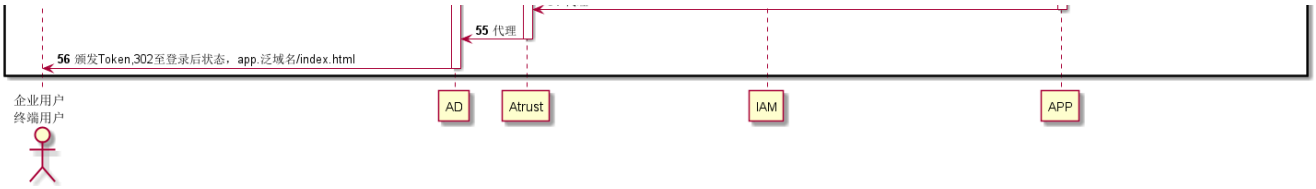


4.3 系统流程

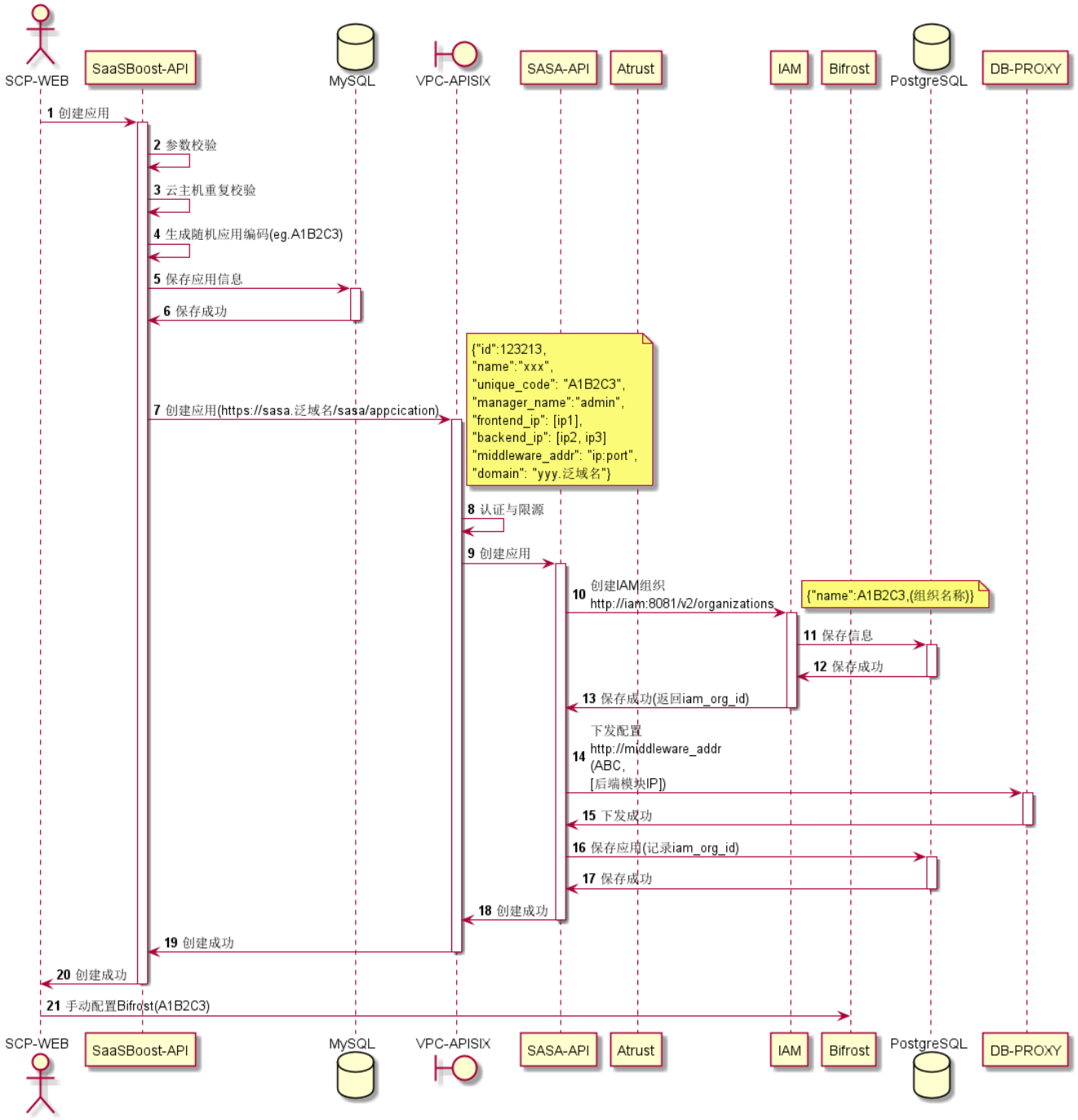
系统的动态视图，重点描述系统功能的实现过程。通过各模块的交互，实现所有系统功能。通过分析系统流程图，导出各模块的接口。依据用例或系统功能需求编写系统流程图。每个用例需要对应一个系统流程。每个功能需求点都要描述其实现方案（模块）或者流程。
对于规模较大的版本，整体架构只需要定义到子系统一级，系统流程只需要描述子系统之间的交互流程和接口即可。针对每个子系统 的设计在子系统概要设计中完成。

4.3.1.单点登录

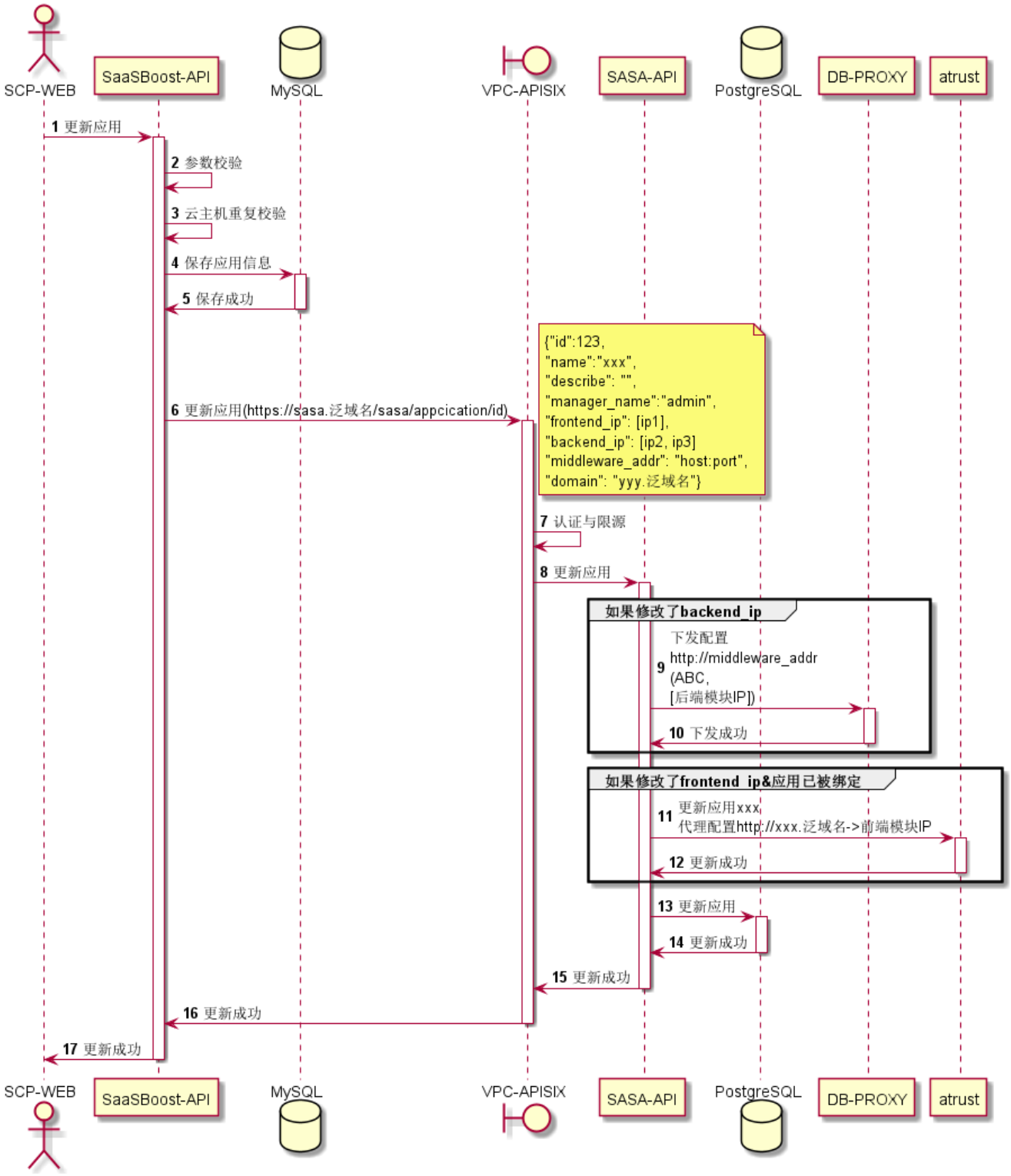

企业用户
终端用户
AD
Atrust
IAM
APP
第一次单点登录
[IAM单点登录到Atrust]
1 访问xxx.泛域名
记录xxx.泛域名到cookie中，domain为泛域名
2 代理
3 判断是否登录
4 获取认证服务器配置
5 代理
6 302访问https://iam.泛域名/authrize?redirect_url=atrust.泛域名/callback
7 获取授权码
8 代理
9 代理
10 校验IAM未登录
重定向到登录页
11 代理
12 代理
13 302至iam.泛域名/login
14 用户输入用户名和密码
15 密码认证https://iam.泛域名/password
前端下发在用户名前面拼接xxx_
16 代理
17 代理
18 密码校验(根据xxx找到绑定的应用唯一标识ABC)
19 代理
20 代理
21 302至https://atrust.泛域名/callback?code=xxx
22 回调callback
23 代理
24 获取accessToken(携带授权码)
25 返回accessToken
26 获取user信息(携带返回accessToken)
返回user信息
27 (sub,
username)
28 校验user是否合法
29 代理
30 颁发Token至atrust.泛域名
第二次单点登录
[IAM单点登录到APP]
31 访问xxx.泛域名
32 代理
33 代理
34 判断是否登录
35 代理
36 代理
37 302访问https://iam.泛域名/authrize?redirect_url=/callback
38 获取授权码
39 代理
40 代理
41 IAM已登录
42 拼接回调地址
43 代理
44 代理
45 302至app.泛域名/callback?code=xxx
46 回调callback
47 代理
48 代理
49 获取accessToken(携带授权码)
50 返回accessToken
51 获取user信息(携带返回accessToken)
返回user信息
52 (sub,
username)
{"sub":id,
"given_name":"AAA",
"family_name":"ABC"}
53 校验user是否合法
given_name即为单体应用的登录用户名
54 代理



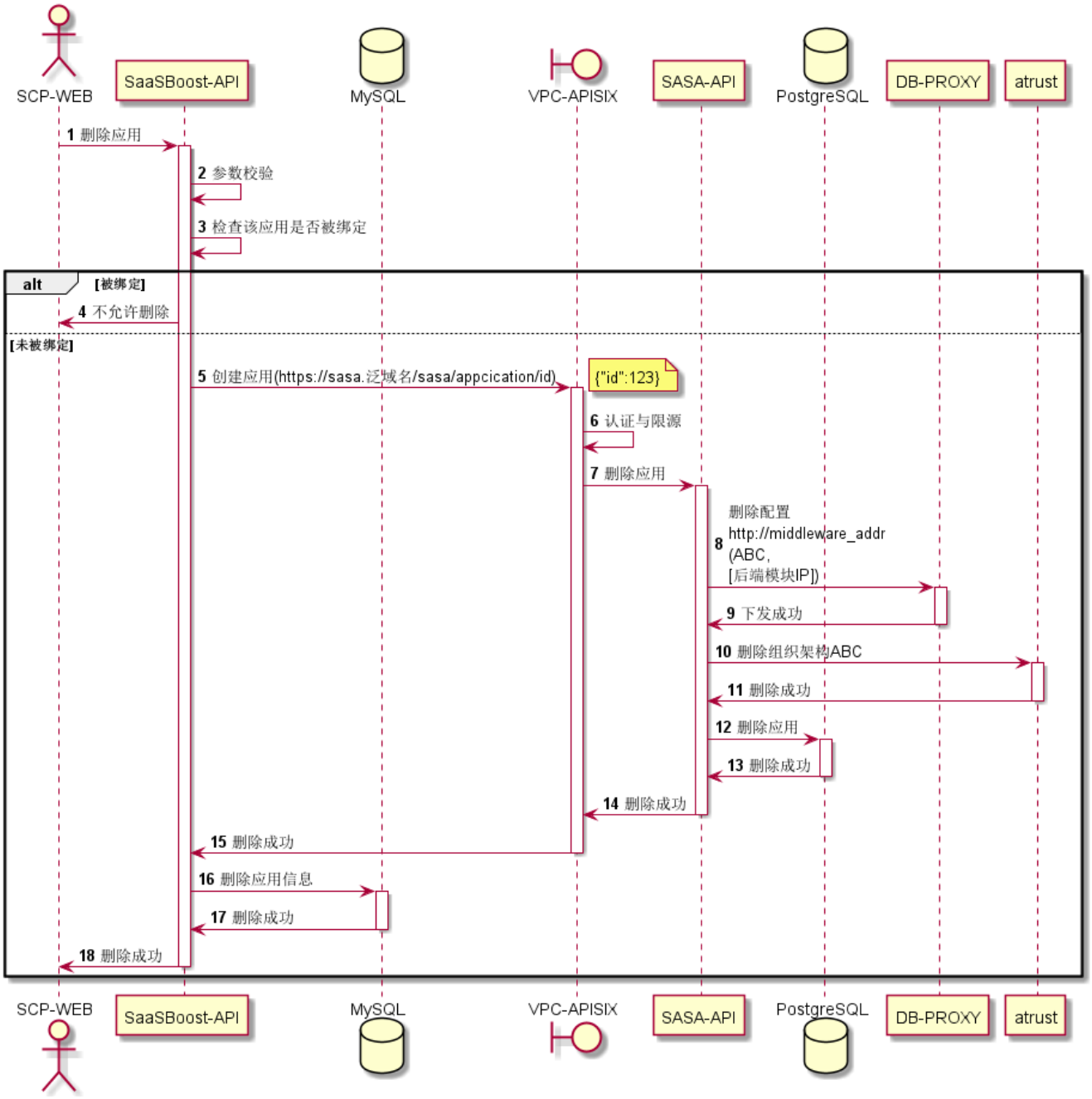
4.3.2.创建应用



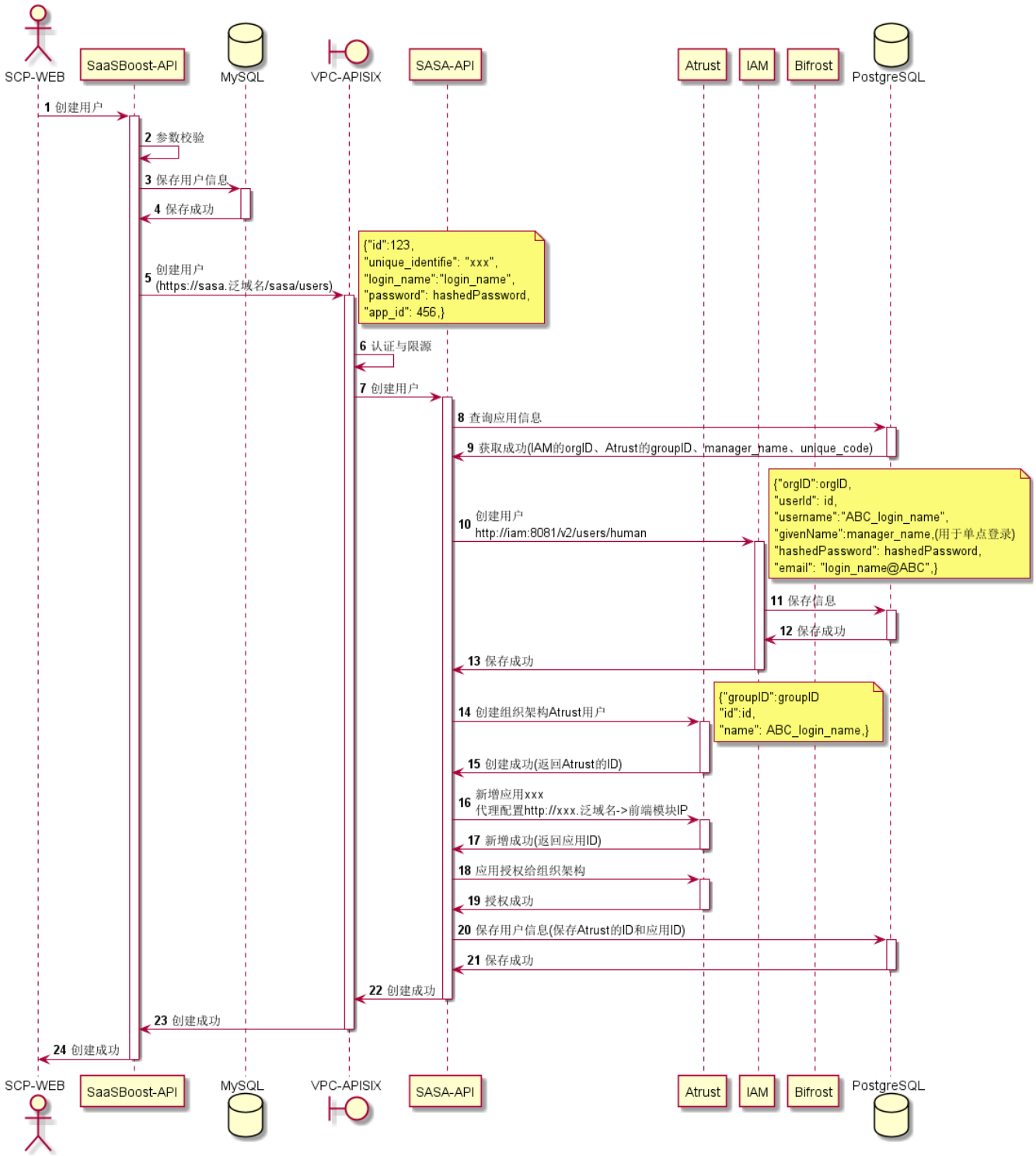
4.3.3.更新应用


SCP-WEB
SaaSBoost-API
MySQL
VPC-APISIX
SASA-API
PostgreSQL
DB-PROXY
atrust
1 更新应用
2 参数校验
3 云主机重复校验
4 保存应用信息
5 保存成功
{"id":123,
"name":"xxx",
"describe": "",
"manager_name":"admin",
"frontend_ip": [ip1],
"backend_ip": [ip2, ip3]
"middleware_addr": "host:port",
"domain": "yyy.泛域名"}
6 更新应用(https://sasa.泛域名/sasa/appcication/id)
7 认证与限源
8 更新应用
如果修改了backend_ip
下发配置
9 http://middleware_addr
(ABC,
[后端模块IP])
10 下发成功
如果修改了frontend_ip&应用已被绑定
11 更新应用xxx,
代理配置http://xxx.泛域名->前端模块IP
12 更新成功
13 更新应用
14 更新成功
15 更新成功
16 更新成功
17 更新成功
SCP-WEB
SaaSBoost-API
MySQL
VPC-APISIX
SASA-API
PostgreSQL
DB-PROXY
atrust

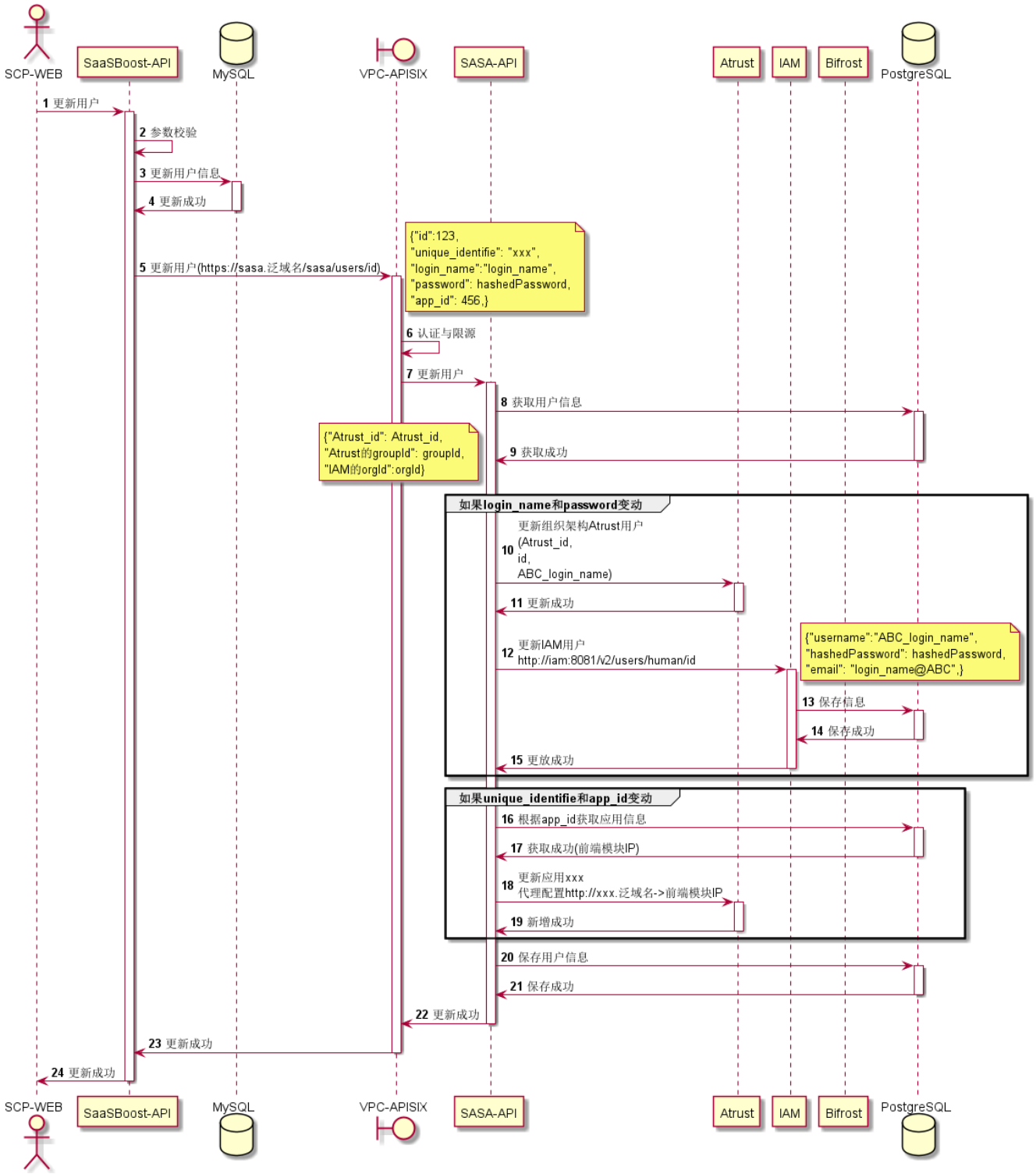
4.3.4.删除应用


SCP-WEB
SaaSBoost-API
MySQL
VPC-APISIX
SASA-API
PostgreSQL
DB-PROXY
atrust
1 删除应用
2 参数校验
3 检查该应用是否被绑定
alt [被绑定]
4 不允许删除
[未被绑定]
5 创建应用(https://sasa.泛域名/sasa/appcication/id)
{"id":123}
6 认证与限源
7 删除应用
8 删除配置
http://middleware_addr
(ABC,
[后端模块IP])
9 下发成功
10 删除组织架构ABC
11 删除成功
12 删除应用
13 删除成功
14 删除成功
15 删除成功
16 删除应用信息
17 删除成功
18 删除成功
SCP-WEB
SaaSBoost-API
MySQL
VPC-APISIX
SASA-API
PostgreSQL
DB-PROXY
atrust

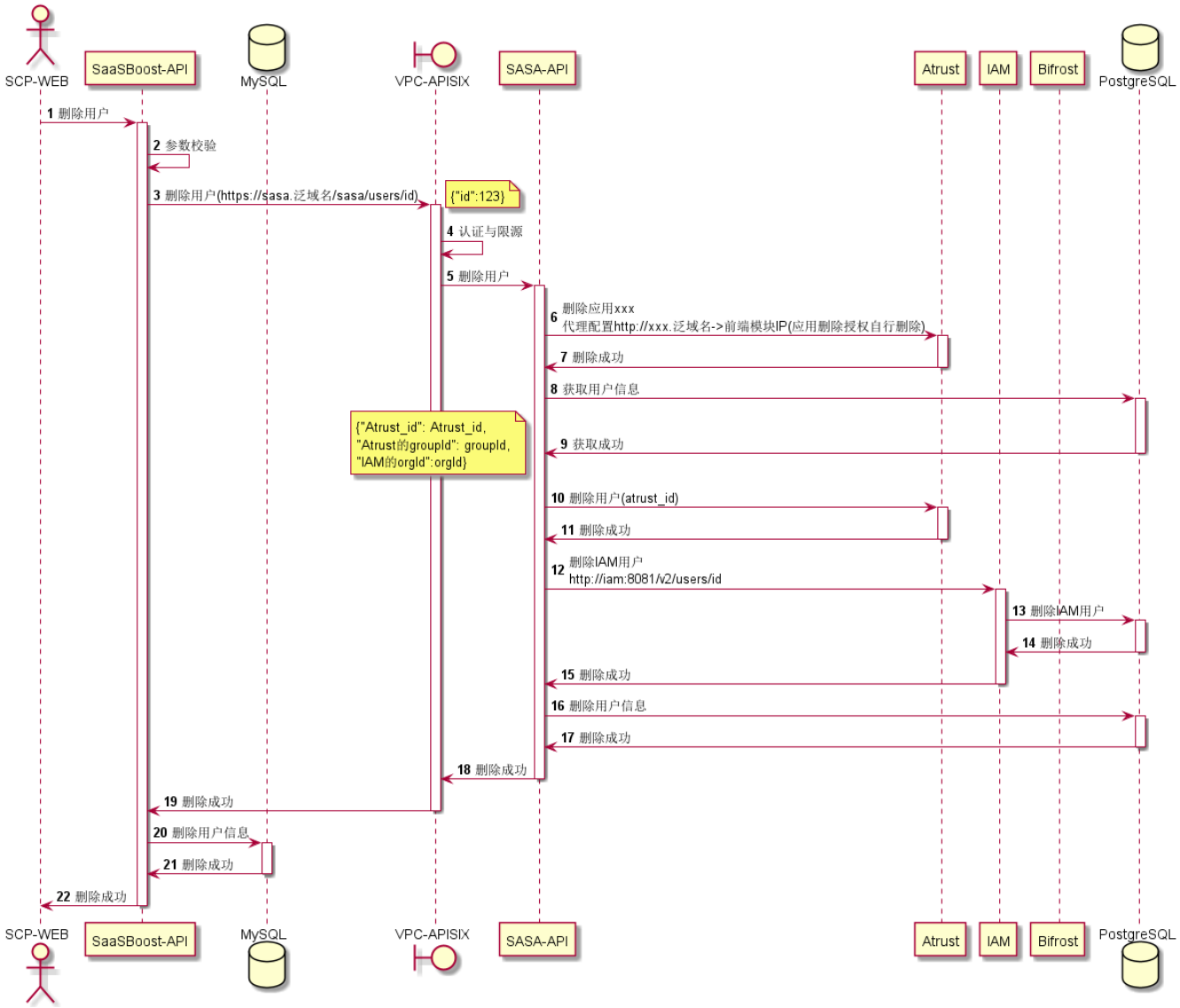
4.3.5.创建用户


SCP-WEB
SaaSBoost-API
MySQL
VPC-APISIX
SASA-API
Atrust
IAM
Bifrost
PostgreSQL
1 创建用户
2 参数校验
3 保存用户信息
4 保存成功
5 创建用户
(https://sasa.泛域名/sasa/users)
{"id":123,
"unique_identifie": "xxx",
"login_name":"login_name",
"password": hashedPassword,
"app_id": 456,}
6 认证与限源
7 创建用户
8 查询应用信息
9 获取成功(IAM的orgID、Atrust的groupID、manager_name、unique_code)
{"orgID":orgID,
"userId": id,
"username":"ABC_login_name",
"givenName":manager_name,(用于单点登录)
"hashedPassword": hashedPassword,
"email": "login_name@ABC",}
10 创建用户
http://iam:8081/v2/users/human
11 保存信息
12 保存成功
13 保存成功
{"groupID":groupID
"id":id,
"name": ABC_login_name,}
14 创建组织架构Atrust用户
15 创建成功(返回Atrust的ID)
16 新增应用xxx
代理配置http://xxx.泛域名->前端模块IP
17 新增成功(返回应用ID)
18 应用授权给组织架构
19 授权成功
20 保存用户信息(保存Atrust的ID和应用ID)
21 保存成功
22 创建成功
23 创建成功
24 创建成功

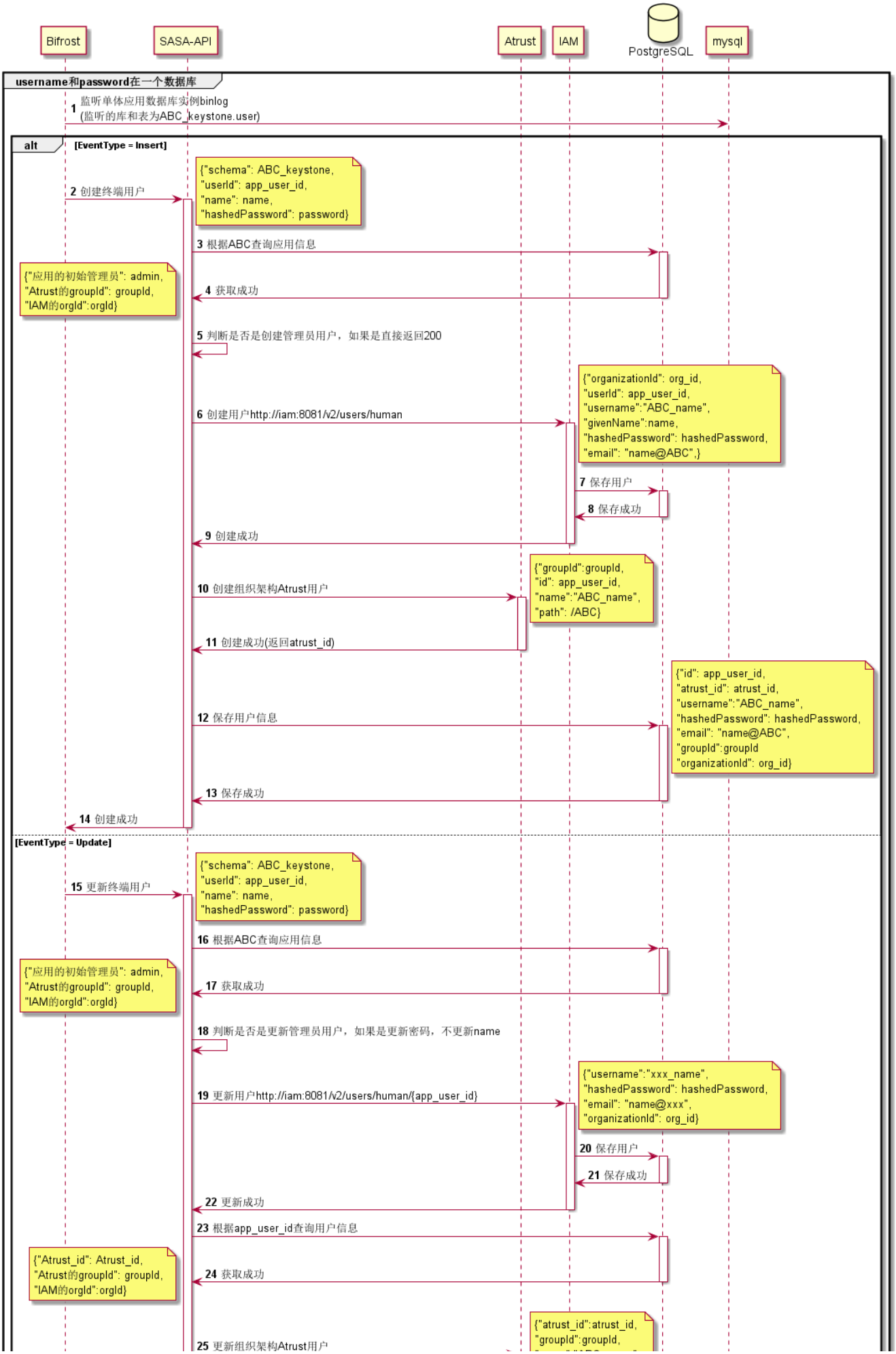
4.3.6.更新用户

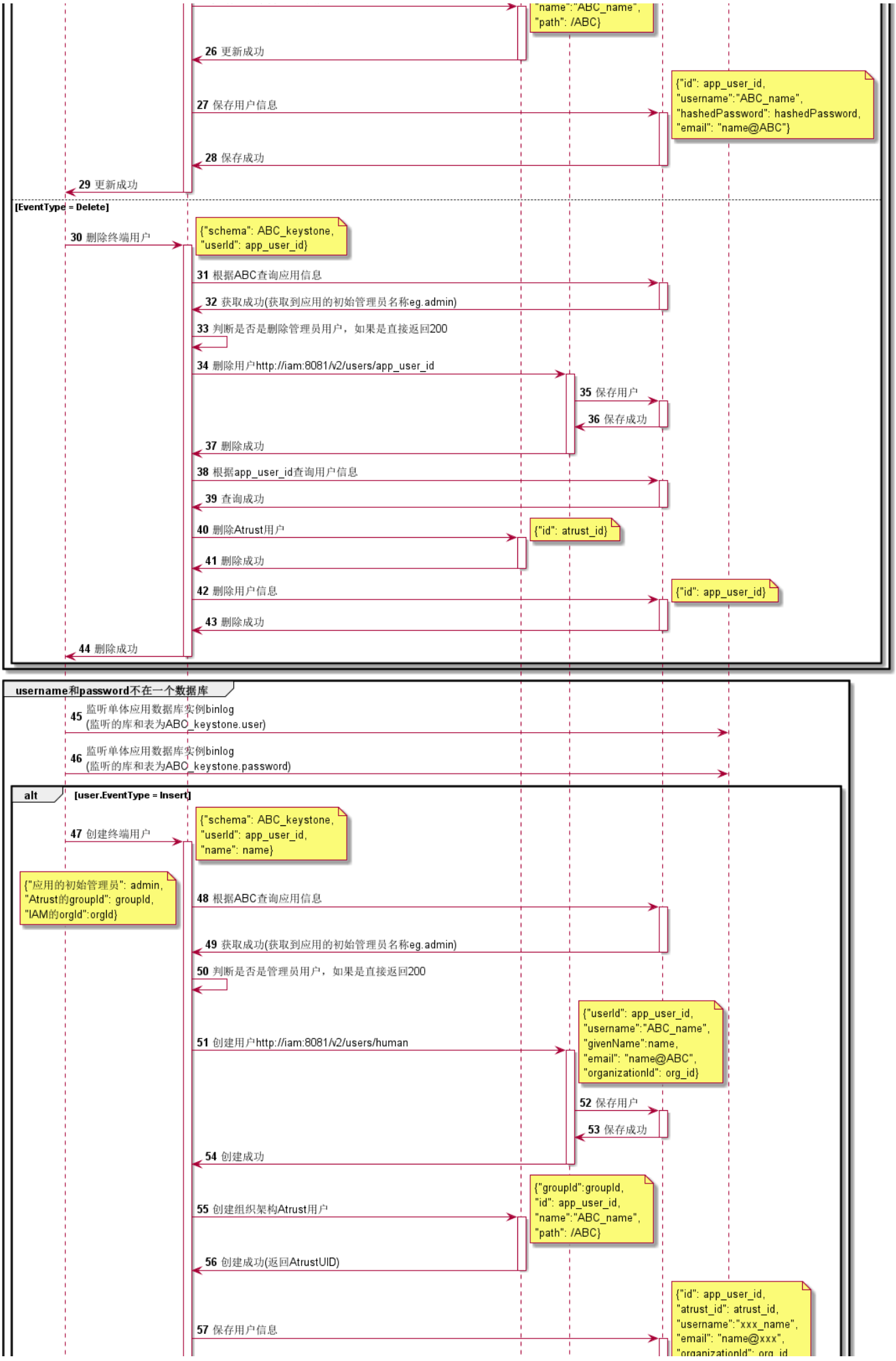

SCP-WEB
SaaSBoost-API
MySQL
VPC-APISIX
SASA-API
Atrust
IAM
Bifrost
PostgreSQL
1 更新用户
2 参数校验
3 更新用户信息
4 更新成功
5 更新用户(https://sasa.泛域名/sasa/users/id)
{"id":123,
"unique_identifie": "xxx",
"login_name":"login_name",
"password": hashedPassword,
"app_id": 456,}
6 认证与限源
7 更新用户
8 获取用户信息
9 获取成功
{"Atrust_id": Atrust_id,
"Atrust的groupId": groupId,
"IAM的orgId":orgId}
如果login_name和password变动
更新组织架构Atrust用户
10 (Atrust_id,
id,
ABC_login_name)
11 更新成功
12 更新IAM用户
http://iam:8081/v2/users/human/id
{"username":"ABC_login_name",
"hashedPassword": hashedPassword,
"email": "login_name@ABC",}
13 保存信息
14 保存成功
15 更放成功
如果unique_identifie和app_id变动
16 根据app_id获取应用信息
17 获取成功(前端模块IP)
18 更新应用xxx
代理配置http://xxx.泛域名->前端模块IP
19 新增成功
20 保存用户信息
21 保存成功
22 更新成功
23 更新成功
24 更新成功

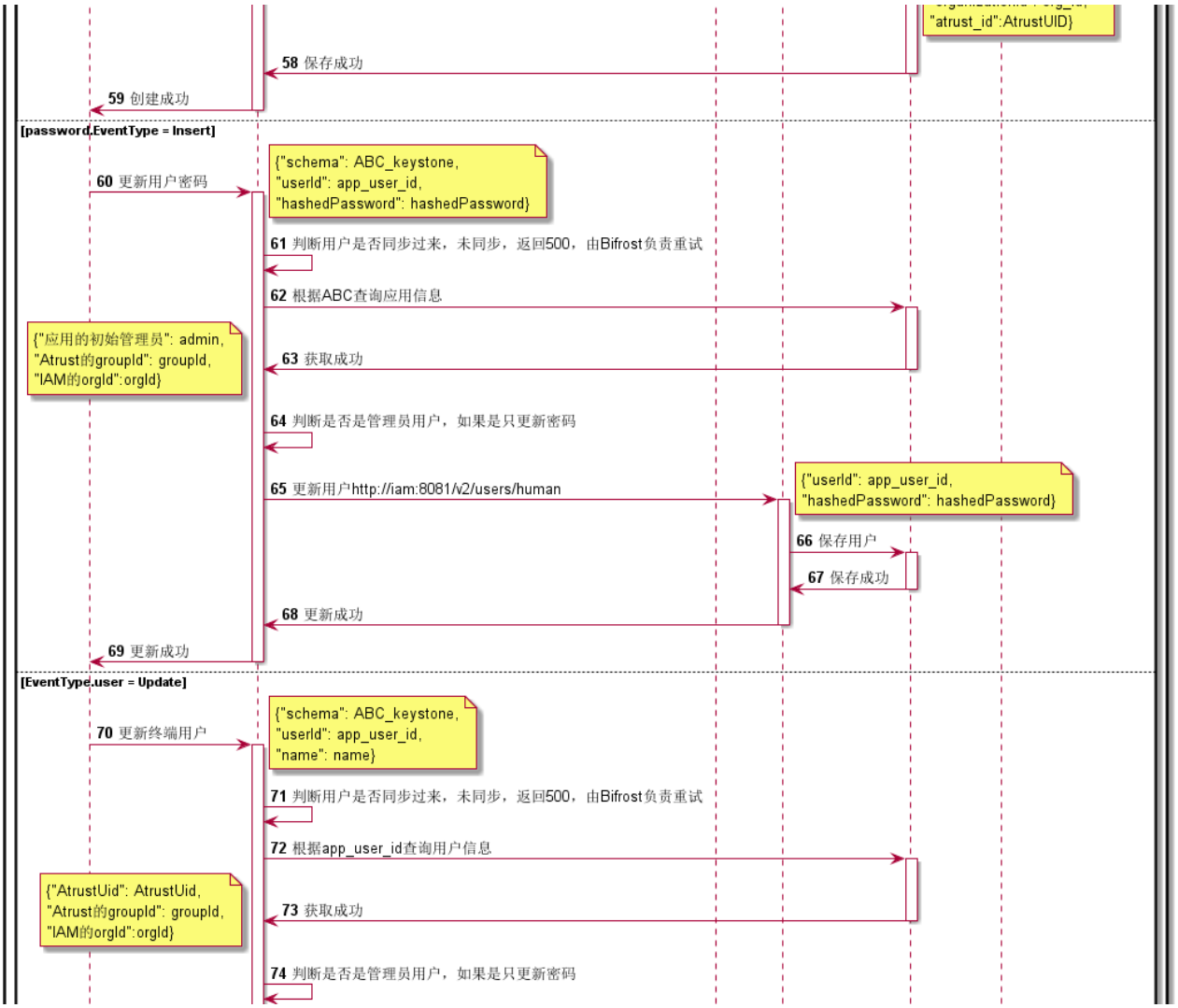
4.3.7.删除用户



4.3.8.Bifrost同步


Bifrost
SASA-API
Atrust
IAM
PostgreSQL
mysql
username和password在一个数据库
1 监听单体应用数据库实例binlog
(监听的库和表为ABC_keystone.user)
alt
[EventType = Insert]
{"schema": ABC_keystone,
"userId": app_user_id,
"name": name,
"hashedPassword": password}
2 创建终端用户
3 根据ABC查询应用信息
{"应用的初始管理员": admin,
"Atrust的groupId": groupId,
"IAM的orgId":orgId}
4 获取成功
5 判断是否是创建管理员用户，如果是直接返回200
{"organizationId": org_id,
"userId": app_user_id,
"username":"ABC_name",
"givenName":name,
"hashedPassword": hashedPassword,
"email": "name@ABC",}
6 创建用户http://iam:8081/v2/users/human
7 保存用户
8 保存成功
9 创建成功
{"groupId":groupId,
"id": app_user_id,
"name":"ABC_name",
"path": /ABC}
10 创建组织架构Atrust用户
11 创建成功(返回atrust_id)
{"id": app_user_id,
"atrust_id": atrust_id,
"username":"ABC_name",
"hashedPassword": hashedPassword,
"email": "name@ABC",
"groupId":groupId
"organizationId": org_id}
12 保存用户信息
13 保存成功
14 创建成功
[EventType = Update]
{"schema": ABC_keystone,
"userId": app_user_id,
"name": name,
"hashedPassword": password}
15 更新终端用户
16 根据ABC查询应用信息
{"应用的初始管理员": admin,
"Atrust的groupId": groupId,
"IAM的orgId":orgId}
17 获取成功
18 判断是否是更新管理员用户，如果是更新密码，不更新name
{"username":"xxx_name",
"hashedPassword": hashedPassword,
"email": "name@xxx",
"organizationId": org_id}
19 更新用户http://iam:8081/v2/users/human/{app_user_id}
20 保存用户
21 保存成功
22 更新成功
23 根据app_user_id查询用户信息
{"Atrust_id": Atrust_id,
"Atrust的groupId": groupId,
"IAM的orgId":orgId}
24 获取成功
{"atrust_id":atrust_id,
"groupId":groupId,
25 更新组织架构Atrust用户


"name":"ABC_name",
"path": /ABC}
26 更新成功
{"id": app_user_id,
"username":"ABC_name",
"hashedPassword": hashedPassword,
"email": "name@ABC"}
27 保存用户信息
28 保存成功
29 更新成功
[EventType = Delete]
{"schema": ABC_keystone,
"userId": app_user_id}
30 删除终端用户
31 根据ABC查询应用信息
32 获取成功(获取到应用的初始管理员名称eg.admin)
33 判断是否是删除管理员用户，如果是直接返回200
34 删除用户http://iam:8081/v2/users/app_user_id
35 保存用户
36 保存成功
37 删除成功
38 根据app_user_id查询用户信息
39 查询成功
40 删除Atrust用户
{"id": atrust_id}
41 删除成功
42 删除用户信息
{"id": app_user_id}
43 删除成功
44 删除成功
username和password不在一个数据库
45 监听单体应用数据库实例binlog
(监听的库和表为ABC_keystone.user)
46 监听单体应用数据库实例binlog
(监听的库和表为ABC_keystone.password)
alt
[user.EventType = Insert]
{"schema": ABC_keystone,
"userId": app_user_id,
"name": name}
47 创建终端用户
{"应用的初始管理员": admin,
"Atrust的groupId": groupId,
"IAM的orgId":orgId}
48 根据ABC查询应用信息
49 获取成功(获取到应用的初始管理员名称eg.admin)
50 判断是否是管理员用户，如果是直接返回200
{"userId": app_user_id,
"username":"ABC_name",
"givenName":name,
"email": "name@ABC",
"organizationId": org_id}
51 创建用户http://iam:8081/v2/users/human
52 保存用户
53 保存成功
54 创建成功
{"groupId":groupId,
"id": app_user_id,
"name":"ABC_name",
"path": /ABC}
55 创建组织架构Atrust用户
56 创建成功(返回AtrustUID)
{"id": app_user_id,
"atrust_id": atrust_id,
"username":"xxx_name",
"email": "name@xxx",
"organizationId": org_id}
57 保存用户信息



4.4. 数据结构

描述模块间交互所使用的数据结构，比如数据库，共享内存，文件，等等。统一的消息格式可在此描述，只涉及少数模块的消息格式也可在接口描述时说明。

4.4.1. 关键数据结构

创建组织及管理员用户(IAM)

URL：http://iam.apple.com:8081/v2/organizations

BODY：

```
{
  "name": "huawei",()
  "admins": [
    {
      "human": {
        "userId": "2889087628237",(SaaSBoostID)
        "username": "huawei_AAA",
        "profile": {
                        "givenName": "huawei",
                        "familyName": "AAA"
        },
        "email": {
          "email": "AAA@huawei",
          "isVerified": true
        },
        "hashedPassword": {
          "hash": "$2a$12$lJ08fqVr8bFJilRVnDT9QeULI7YW.
nT3iwUv6dyg0aCrfm3UY8XR2",
          "changeRequired": true
        }
      }
    }
  ]
}
```

删除组织(IAM)

URL：http://iam.apple.com:8081/admin/v1/orgs/:orgId

BODY：NA

创建用户(IAM)

URL：http://iam.apple.com:8081/v2/users/human

BODY：

```
{
  "userId": "2889087628237", (IDSaaSBoostID)
  "username": "huawei_AAA",(_)
  "profile": {
    "givenName": "huawei",()
    "familyName": "AAA"()
  },
  "organization": {()
    "orgId": "huawei",
    "orgDomain": "string"
  },
  "phone": {
    "phone": "+41791234567",
    "isVerified": true
  },
  "email": {
    "email": "AAA@huawei",(@)
    "isVerified": true
  },
  "hashedPassword": { ()
    "hash": "$2a$14$qJW2Gwwj68kwtn6VsdRrq.66G5v7Sr1O6ZO7B8mZbjL.fh1
/pVX8e",
    "changeRequired": false
  },
  "password": { ()
    "password": "Secr3tP4ssw0rd!",
    "changeRequired": false
  }
}
```

更新用户

URL：http://iam.apple.com:8081/v2/users/human/{user_id}

BODY：

```
{
        "userName": "gzs-bcrypt9",
        "profile": {
                "givenName": "huawei",
                "familyName": "AAA"
        },
        "email": {
                "email": "minnie6@mouse.com",
                "isEmailVerified": true
        },
        "hashedPassword": {
                "hash": "$2a$14$qJW2Gwwj68kwtn6VsdRrq.
66G5v7Sr1O6ZO7B8mZbjL.fh1/pVX8e",
                "changeRequired": false
        }
}
```

删除用户

URL：http://iam.apple.com:8081/v2/users/id

BODY：NA

4.4.2. 流程关键数据结构

4.5 模块设计

4.5.1.SaaSBoost服务模块

4.5.1.1.模块系统需求

4.5.1.2.模块设计约束

4.5.1.3.接口列表

4.5.2.SASA-API模块

4.5.2.1.模块系统需求

4.5.2.2.模块设计约束

4.5.2.3.接口列表

4.5.3.IAM模块

4.5.2.1.模块系统需求

4.5.2.2.模块设计约束

4.5.2.3.接口列表

4.5.4.Bifrost模块

4.5.3.1.模块系统需求

4.5.3.2.模块设计约束

4.5.3.3.接口列表

4.5.5.DB-PROXY模块

4.5.4.1.模块系统需求

4.5.4.2.模块设计约束

4.5.4.3.接口列表

4.6 风险分析

基于模块的设计目标，分析哪些点可能存在风险。基于模块的运行场景，分析各个环节可能遇到的问题，面临的风险。

风险分析可采用头脑风暴方法搜集或使用思维导图分析。风险确定后，需要对风险进行排序，确定风险最大，最需要先解决的问题。分析完成后，按风险从大到小排列填写下表。

序号	风险标题	风险说明	下一步思路
1			
2			
3			
4			
5			

5. 关键特性设计

5.1. 业务出错处理

用一览表的方式说明每种可能的出错情况出现时，系统输出信息的形式、含意及处理方法。

系统流程	出错步骤	出错可能原因	错误处理方式	影响（网络中断、网络抖动、性能下降、配置丢失）
通用问题				
客户问题				

5.2. 避免业务长时间中断处理

一些可能导致用户的业务长时间中断的问题或者隐患。是“系统出错处理设计”的一个子集。

业务中断错误编号	受影响客户业务	可能导致的异常及其原因	恢复和应对措施	可测试性保证
1	客户自身的某个业务服务异常退出	OOM或自身BUG等		

5.3. 硬件故障的可靠性保障措施

描述怎么检查系统中出现的硬件故障，以及出现故障时的业务保障措施。

5.4. 补救措施

说明故障出现后可能采取的变通措施，包括：

a. 后备技术：说明准备采用的后备技术，当原始系统数据万一丢失时启用的副本的建立和启动的技术，例如周期性地把磁盘信息记录到磁带上去就是对于磁盘媒体的一种后备技术；

b. 降效技术：说明准备采用的后备技术，使用另一个效率稍低的系统或方法来求得所需结果的某些部分，例如一个自动系统的降效技术可以是手工操作和数据的人工记录；

c. 恢复及再启动技术：说明将使用的恢复再启动技术，使软件从故障点恢复执行或使软件从头开始重新运行的方法。

错误编号	措施类型	补救措施

5.5. 维护设计

说明为了系统维护的方便而在程序内部设计中作出的安排，包括在程序中专门安排用于系统的检查与维护的检测点和专用模块。

5.6. 安全性设计

说明系统需求规格说明书中描述的安全性需求如何实现。如果是由某个模块独立实现，仅需说明具体由哪个模块实现，这个需求应该明确纳入该模块的设计目标书。

5.6.1. 威胁建模分析

威胁建模分析，目的是分析该版本总体设计中涉及到的各个模块实体、数据存储、进程服务、开放端口、交互协议等是否存在相关的安全威胁风险，把风险识别出来并填入如下表格，

具体威胁建模分析方法，采用微软的STRIP方法来分析，具体方法请参考如下地址，看不懂的可以找产品线安全接口人咨询，或发邮件给RDSec提供咨询培训。

元素	威胁分类	威胁概述	威胁场景名称	威胁评级	威胁分析	消减方案	当前状态	测试方法(攻击方法)
连接器服务/流程引擎	篡改T，欺骗S	攻击者可能会在连接器和流程引擎之间插入自身，使得连接器和流程引擎双方错以为正在直接通信，但是攻击者监控和篡改了双方的所有数据	中间人攻击	中	缺乏加密通信，如果通信双方没有采用加密通信协议，通信内容就会以明文形式传输，容易被攻击者窃取和篡改； 如果通信双方没有进行有效的身份验证，攻击者可以冒充其中一方的身份，使得双方认为他们正在直接通信，但实际上所有的通信都经过了攻击者的监控和篡改；	1.使用加密通信协议：通信双方应该使用TLS1.3加密通信协议，确保通信内容经过加密 2.沿用托管云中的身份验证机制：通信双方应该使用AKSK认证等身份验证机制 3、签名中有防重放的机制	已处理	
连接器服务/集成管理服务	篡改T	在创建流程时，攻击者通过在模型参数中输入字符串时注入恶意的SQL指令攻击。在设计不良的程序当中忽略了检查，那么这些注入进去的指令就会被数据库误认为是正常的SQL指令而执行，因此遭到数据泄漏或者数据的破坏。其主要原因是程序没有细致地过滤用户输入的数据，致使非法数据侵入系统。	SQL注入	高	用户输入的数据未进行参数校验导致产生SQL注入； 用户输入验证的不充分，如果应用程序没有对用户输入进行充分的验证和过滤，攻击者更容易成功注入恶意的SQL代码。	1.输入验证和过滤：应用程序应该对用户输入进行充分的验证和过滤，确保输入的数据符合预期的格式和范围 2.参数化查询，不直接将用户的输入作为SQL代码的一部分 3.最小权限原则，减少被攻击的潜在风险	已处理	通过postman测试请求，构造请求参数包含 or '1'='1' sql注入数据，判断是否检测出这个注入问题存在

连接器服务密码管理	信息泄露I	租户的密码信息存储租户VPC中的sqlite数据库中，攻击者通过其他的攻击手段进入租户的sqlite中，其中存储的用户密码信息将会泄露，攻击者可以使用这些用户密码信息肆意非法访问租户的其他资源或服务	数据泄露、密码泄露	中	1.密码以明文或弱加密形式存储在数据库中，使得攻击者可以轻易获取 2.数据库中存储了过多和密码直接绑定的详细信息，映射关系清晰	1.密码存储使用密文存储落库，SCC加密，租户的连接器服务解密并使用，即使被攻破，拿到的也是密码的密文 2.只存密码密文和映射key，不存储别的用户信息，不指明这个密码是什么服务或中间件的密码，不提供中间件或服务的访问地址，即使密文被解密，得到的也只是一段没有意义的字符串	已处理	无
连接器服务/流程引擎	欺骗S	攻击者模拟连接器满足触发条件时创建流程实例，根据流程ID创建别人的流程实例；连接器向流程引擎汇报任务的时候根据jobkey,随意的汇报其他用户的job完成状态； 也即：攻击者可以根据某个资源ID，在没有进行有效身份校验的系统中，未经授权地访问其他用户的资源	越权操作	中	1.系统中缺乏有效的身份校验机制，使得攻击者可以绕过访问控制，直接使用资源ID进行访问 2.攻击者可以通过猜测或枚举的方式尝试访问其他用户的资源	1.资源ID必须是复杂、不可预测的，增加猜测和枚举的难度 2.强制的身份验证方式(AKSK)结合验签算法openapi_auth:(aksk用于验证用户的身份和授予资源访问权限；验签用于验证数据的完整性) 3、互信验证 4、归属性校验(防止操作其他租户的资源)	已处理	构造gRPC或HTTP请求，访问其他租户的流程，如启动其他租户的流程，创建流程实例；结果是到达apisix就会被拦截和报错，无法做到实际操作非法资源
连接器服务/流程引擎	流量攻击	攻击者劫持租户的VPC，无限制的访问SCC中心端，导致中心端不可用	流量攻击DDOS	中	1.系统没有对连接器的访问做限制，访问控制限制不当	1.限制连接器在一段时间内(1min)的最大访问次数，超过次数则限制访问 2、平台有限流的机制	已处理	通过Jmeter并发请求，超过限流设置的请求会被服务端拒绝，状态码:429
AK/SK泄露	信息泄露I	获取云服务台地址，窃取AK/SK	窃取API密钥	高	1、窃取API密钥	1、SK加密存储 2、AKSK可以访问的API我们会权限校验进行约束，无法访问没有权限的API 3、AKSK有白名单机制，会设置信任的客户端IP，即使泄露也不能在任意网络位置使用 4、平台可以快速切断任意AKSK的访问能力	已处理	1、在不信任的网路位置访问 2、访问没有权限的API
接口调用	篡改T	接口调用参数防注入	SQL注入	高		针对参数做严格类型校验	已处理	页面创建流程的时候使用特殊字符参数校验会报错
创建流程：输入密码	信息泄露I	日志中泄漏敏感信息	日志中泄漏敏感信息	中	日志中会输出用户密码，不安全	日志对密码进行星号*代替	已处理	创建流程的时候，选择需要输入密码的应用，然后查看日志
租户调用其他租户的OpenAPI	提权E	租户A请求租户B的URL	水平越权	高	租户可能会调用其他租户的OpenAPI	进行URL的归属性校验	已处理	租户A去调用另一个租户的OpenAPI

5.6.2. 安全设计

根据上一节威胁建模识别出来的风险和建议采取的安全措施，设计相关的安全机制，包括整体的安全架构设计和各个模块需要设计的安全机制，在此章节进行详细描述，比如通信协议要加密，就需要设计相关的加密方案。

5.6.1.1. 整体安全架构设计

5.6.3. 预使用组件版本合规性情况

使用黑鸭扫描器对预计使用的组件版本进行合规性扫描（https://200.200.0.207/ 账号权限以及使用方式，请与张笑尘联系），并将扫描结果上传至本处，原则上不允许使用提示告警的组件，如果需要使用，需经过法务胡海斌审批通过，并将分析结果上传至该处，降低后续关联修复的巨额成本。（黑鸭扫描报告同时含有合规性、漏洞情况）

OpenHub给出Zeebe的漏洞报告--无报告的漏洞

5.6.4. 预使用组件版本漏洞情况

使用黑鸭扫描器对预计使用的组件版本进行安全性扫描（https://200.200.0.207/ 账号权限以及使用方式，请与张笑尘联系），并将扫描结果上传至本处，原则上不允许使用含有漏洞的组件版本，如果需要使用，需经过安全经理、主管审批通过，并将分析结果上传至该处，降低后续关联修复的巨额成本。（黑鸭扫描报告同时含有合规性、漏洞情况）

5.7.可靠性设计

可靠性目标定义：

1）可用度的目标定义：比如产品可用度达到5个9/年中断时间少于5分钟

2）可靠性的目标定义：比如故障检测&定位率达到70%，故障自动恢复率达到85%，故障人工恢复时长小于30分钟等等

可靠性系统设计：

进行系统级的FMEA分析，给出系统在冗余设计，故障管理（检测、定位、上报、恢复等）等方面的设计机制，给出系统在防人因差错方面的设计机制；给出系统在故障预测预防方面的实现机制（主要是硬件方面，比如针对硬盘的故障预测，针对内存的故障预测等等）；给出升级不中断业务方面的实现原理（包含升级回退机制）

5.8. 韧性设计

韧性目标：定义系统业务过载控制目标：比如在2倍业务情况下，成功率达到98%；在3倍业务情况下达到95%；在10倍业务情况下，成功率达到93%；产品在被拒绝服务攻击时，处理性能虽然降低，但仍然可以开展业务服务；

韧性设计：给出系统在业务过载保护方面的设计机制，包含但不限于各类业务的优先级设计，各种业务优先的控制实现机制；此外需要确保产品在遇到消耗大量性能攻击时，需要有对应的安全机制，例如：在Apache/Nginx需要设置拒绝服务的超时机制，并且对于同一时间发起大量无效请求的IP进行一段时间的拒绝响应，直至产品能够完全正常响应。

5.9. 性能设计

将来会有大量的异步流程下沉到云桥引擎中处理，所以需要评估一下云桥引擎在高负载情况下的性能和稳定性。其作用是模拟系统在实际使用中可能遇到的高负载情况，以便发现系统在负载高峰期间可能出现的问题和瓶颈，并进行优化和改进。

云桥连接器作为执行任务的进程，需要拥有并发处理的功能，需要评估一下在极限场景下，云桥连接器的处理性能

5.9.1. 测试方案

分析开发的技术方案，对方案的缺点进行深入分析，主要分析维度是需求满足度、客户场景满足度、性能、稳定性、可靠性维度，

5.10. 可监控性设计

描述模块对监控平台适配相关的设计，是否支持配置监控，如果没有，此处写不涉及即可。

云桥引擎中运行的流程实例大部分为异步进行，提供API查看所有正在运行或已经结束的流程实例，监控流程运行的情况。后续可以接入Prometheus和Grafana来监测云桥引擎集群拓扑的健康度、吞吐量指标、已处理的请求、每秒导出的事件、磁盘和内存使用情况等。

5.11. 可调试可测试设计

分析系统实际运行时，容易出现问题、定位问题比较困难、或者测试验证比较困难的逻辑。针对这些逻辑，需要在机制层面考虑怎么快速定位、解决问题，降低调试、测试的难度，同时不对系统运行施加过多负面影响。

可测试性	说明
后台CLI命令	提供zbctl，方便测试/开发人员进行测试。
接口性能测试	所有Python模块的关键接口及函数均使用以下装饰器，开启DEBUG模式下将统计接口耗时： from oslo_utils.timeutils import time_it @time_it(logger=LOG, enabled=LOG.isEnabledFor(logging.DEBUG))
性能压测	可以通过手动部署流程、并发启动流程实例的方式对云桥引擎进行压力测试。
单元测试	新增的Python代码使用单元测试覆盖
API测试	Python模块使用自动化接口覆盖完成
业务、场景测试	覆盖端到端的自动化测试
流程实例排障	通过统一的API监控和操作运行的流程实例

5.12. 系统隐私设计

隐私目标：定义系统不会出现因自身安全而造成敏感信息泄露的目标：系统在收集用户信息时，不会出现私自收集用户不清楚及用户不同意收集的数据；

隐私设计：给出系统在业务传输过程保护方面的设计机制，包含但不限于各类业务的传输敏感信息时，需要使用安全协议传输，以及对内容进行加密传输；此外需要确保系统在收集信息时，应明确为什么需要收集该信息，还应尝试预判未来是否会使用这些信息用于其他事物，并告知用户是否有这样的计划，提供一份易于阅读的条款或条件的摘要。

5.13. 可重用设计

5.13.1. 技术规范落地计划

根据产品线的技术规范基线计划，定义本版本需要遵守、实现、新增、修订的技术规范。具体参考流程说明：http://docs.sangfor.org/pages/viewpage.action?pageId=97130963

公司已发布的技术规范列表请参考流程中的相关说明。

请填写附件《技术规范基线计划》中的表格，结果以附件的形式保存在本总设文档。此计划需通过技术规范委员会评审，具体要求，请参考上述流程中的相关说明。

5.13.2.本版本可采用的公用代码

通过识别(和预研部门、RDM一起识别)，列出哪些功能点或模块可以使用公用代码。

5.13.3. 本版本可产出的公用代码

要考虑产出公用代码，以方便其它产品或版本来重用。 这样可以持续提高公司的代码重用率，进而降低研发成本，提高效率。

5.13.4. 系统依赖解耦设计

分析系统（或系统内组件）移植到其它软硬件环境，需要考虑的依赖因素。描述降低系统（或系统内组件）对软硬件环境依赖性的设计，即怎么解耦，屏蔽系统（或系统内组件）对底层软硬件环境的依赖。保证系统（或系统内组件）移植到其它软硬件环境时，已有代码逻辑无需修改，或仅需少量修改。

6. 总结

6.1. 与上一个版本的设计变化

描述本版本与上一个版本之间发生变化的总体设计(该设计涉及到多个模块交互或修改后影响多个模块)。如果这个版本总体设计没有变化，或者是新产品，可不填此表，但需在此明确说明没有变化。

本节可以按章节描述，也可以直接列成一个表格

6.1.1. 设计变化项1

6.1.1.1. 设计变化原因

为什么该设计需要变更？

6.1.1.2. 设计变化描述

描述具体发生变化的内容。

6.1.1.3. 设计变化产生的影响

该设计变化会造成什么影响？

6.1.2. 设计变化项2...

6.2. 风险问题保证

风险问题指那些对于达到预定目标没有十足把握的设计问题。

风险点	**风险预研结果/风险规避措施**
例1：处理速度达100MBPS	*根据预研文档《xxx》，可以达到要求*
例2：缓存数据耗用内存可能超过800M	*如果超过800M，使用最久未使用淘汰算法将部分数据保存在磁盘上。*

6.3. 尚未解决的问题

说明在总体设计过程中尚未解决而必须在系统完成之前必须解决的各个问题。

如果总体设计阶段没有问题需要遗留到后续阶段完成，本章节可以为空。

7. 变更控制

7.1. 变更列表

在编码过程中，可能会有一些设计不合理的地方，需要对原设计做一些调整，需要在这里说明变更的章节、内容、原因，并对变更进行评审确认（至少需要经过版本经理/开发经理的确认）。此章节在编码完对设计文档更新后进行评审确认。

需求变更引起的设计变更，在下面加以说明，并对设计变更的结果进行部门内部评审。

变更章节	变更内容	变更原因	变更对对老功能、原有设计的影响
			考虑完全这些影响，要和开发经理、架构师、设计人员、老版本的开发人员进行交流，并最后评审。