# License设计

# 背景

随公司业务发展,部分产品需要独立安装远程客户机使用,对提供的软件在使用过程中如果没有限制,则产品可持续发展利益无法保证,同时为降低重复开发、节省成本，在此背景下需要设计一个通用的对各个产品线提供可约束功能、使用期限、可限制访问资源license机制,直到客户愿意付费购买正版license才可正常使用软件全部或者部分开放功能.

# 定位

为降低license重复开发成本,我们需要设计一个支持各产品的通用license管理软件,同时兼容各产品提出的license控制项.

# 产品定义

对远程部署软件进行资源、功能、使用期限等主要功能控制.

目前仅支持离线分发license；

# 使用客户

公司内部市场销售、运维、测试人员；

# License产品整体功能划分

License产品划分服务端和client端；

1. 服务端

服务端主要维护各产品License控制项模板以及之前颁发的license记录、以及颁发license文件；

服务端功能列表：

* 维护各个产品License模板；
* License模板可编辑为生成license.dat实例数据；
* License文件索引维护以及查询；
* 颁发历史记录（以前的数据维护）；
* 服务端和client端链接库公私钥管理（公钥可下载）；
* 人机交互界面；

1. Client端链接库

client端主要功能是配置项检测以及对比，主要有时间、版本(建议删除)、或附加机器ID等配置项的检测与对比；

client功能列表：

* 支持过期时间检测以及动态过期检查;
* 支持产品运行机器硬件ID绑定与检测;
* 支持产品基线或具体版本绑定控制(建议删除)；
* 支持各产品自定义KV控制项;
* 支持产品基线版本以及具体产品版本控制(建议删除);
* 支持License链接库版本与链接库主版本一一对应；

（如果模板中license链接库主版本升级则需要升级检测链接库）；

整体结构由硬件采集程序、License生成服务端、License Clent端构成；



# 模板设计与使用

License生成是主要根据各产品的License模板配置项进行编辑或者不用编辑直接模板生成；

1. License模板设计

License模板由系统默认模板配置项+产品自定义配置项组成；

模板中的所有配置项的值由各产品确定；

1. 系统默认模板配置项（系统提供必填配置项）

* License模板版本（默认开启）：产品配置项改变则新建模板并升级模板版本（或者叫配置变更独立与产品版本）；
* License连接库版本（默认开启）：

License升级的版本与客户端动态库主线版本一致（约定: 如动态库升级公钥则升级主版本号，假如新增功能则升级小版本、假如修改小版本bug则升级对应小版本之后的补丁版本号）；

* 名称（默认开启）：产品名称，例如“switch-directory-chain”；
* 产品类型（默认关闭）：例如，建议为标准版、测试版、尝鲜版；
* 过期时间（默认开启）：建议标准版默认365days、测试版180days（以天为单位）
* 开始时间（默认关闭）：License计时时间，默认申请时开始计时；
* 网卡地址（默认关闭）：防止License文件拷贝，可绑定机器ID；
* 主板地址（默认关闭）：防止License文件拷贝，可绑定机器ID；
* CPU ID （默认关闭）：防止License文件拷贝，可绑定机器ID；

默认配置项如果启用，则有license client端进行对比校验；

1. 产品自定义配置项

自定义KV控制项由各产品维护其KV配置项，其对比、校验由license client端回调函数反馈结果（回调函数具体实现由各产品负责实现）

例如：其中自定义配置项可以是可访问的路由、具体产品版本等等

（不建议license写入产品版本，因为兼容升级会有问题，具体查看第十二主题说明）

建议： 如果产品升级因新增kv控制项，原有系统加载旧的license文件，旧的kv控制项都需要对比检测，应该在回调函数返回需要检测的自定义kv对比、验证结果；新增的配置功能各产品应该默认关闭；

1. 由模本生成License

产品模板中（系统模板+自定义模板）开启的配置项 对其编辑或者直接申请license，生成license.dat文件；

* 保存生成license.dat实例的模板；
* 可根据历史实例模板再次申请license.dat文件；
* 每个license.dat拥有唯一序号（可查询、可对证）以及系统历史记录；

# 模板与产品版本关系

每个产品有自己独立于其它产品KV控制项，引入自定义KV控制和License共性控制项（启用与否由各产品决定）

对每个产品生命周期中产生各个阶段的KV控制的变更（新增、删除），

所以不强求产品版本和license版本一一对应，即license版本或简称配置集

版本和产品版本解耦；

且产品应兼容低配置集（license版本）中的配置项；

# License 服务端设计

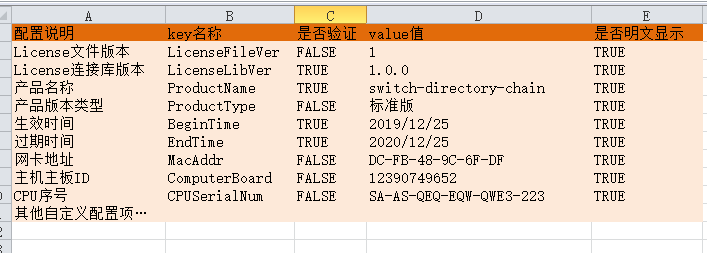
License服务端主要是颁发license证书，一般是由公司专人管理，考虑到操作的易用性，建议采用web人机交互界面，由专人在web端申请；其模板维护由各产品负责人协助License管理人完成模板录入、生成；

产品负责人对产品license 控制项的值需要关注，主要有产品主线版本、是否启用硬件ID和License绑定等等。

产品负责人提交配置模板，模板内容请查看附件：



附件内容如下：（其中开启验证的系统配置项都需要client端动态库验证）



license申请人负责具体天数以及硬件绑定等信息录入以及license生成；

流程如下图所示：



由产品负责人填写License模板，确定License版本（主要是kv配置项是否有升级、更新，且不能覆盖之前的模板license版本），并协助license管理人员手工录入模板；

如果对应产品需要绑定硬件信息，须告知客户（市场、运维、测试）采集机器数据，机器数据主要是主板序号、mac地址、cpu序号等，考虑到非技术人员的可能采集困难，可提供二进制程序协助采集数据；应保密单位禁止安装二进制则提供命令手册协助采集；

License管理人员申请license是根据license模板生成，系统会保留其采用模板生成的license.dat实例和时间（申请人有可能会修改过期时间以及其他默认配置项）；

# 签名私钥与公钥管理（模板KV控制项）

链接库替换公钥才升级主版本，所以链接库如果升级，则系统之前的最后一次使用的模板也应该进行更新，应使用新的私钥签名KV配置项；

假如系统升级链接库版本，即产生新的签名私钥和公钥，则公钥下载下来，需要硬编码到链接库中（替换之前的公钥）；

系统之前的签名私钥以及公钥需要保留并和license链接库主版本对应；

私钥采用加密方式保存系统中，可考虑对等密钥加密签名私钥，公钥不加密；

公私密钥在系统动态生成；

对等加密密钥硬编码系统中，即使copy走加密的私钥，不知道代码中的加密密钥，仍然难于破解，知道代码中的加密密钥，无法获取签名私钥文件也无法签发license；

# License Client端设计

License文件解析由动态库统一解析并验证签名、其模板中开启的验证项由client端的动态库直接检测；

目前大多数系统都是Linux平台，其client端链接库是so结尾的文件，编译时剔除符号表，进行内部符号隐藏；

动态库提供给产品的接口，考虑到目录链、switch都是go实现、数易通是java实现，所以动态库尽量提供C接口（内部实现可以是go，用C包裹go接口，在实现时需确认可行性，如果无法实现，只能C实现供其它异构语言调用）

license中各产品自定义的配置项，需要各产品自行对比验证，链接库提供回调函数，统一返回license检测结果，其回调函数内部由各产品自行实现验证、对比；



业务系统调用链接库，链接库检测License配置文件，获取配置项，并检测过期时间、机器信息等配置，统一返回是否可用；

同时动态库提供获取license配置项功能，供其产品其他权限逻辑功能实现；

例如：从license里面直接读取接口路由配置，初始化restful接口，对没有初始化的接口则禁止访问。

其接口有：

* 提供给各产品的license过期回调函数（license链接库检测过期时间并调用回调函数，回调里面的内容由产品确定，是关闭程序还是限制功能，需提供子函数，让回调函数触发）
* 各产品自定义KV配置项对比、验证回调函数，如有验证失败需要有明细出错信息带回，具体参数在实现时考虑；
* 读取license文件的配置项接口
* 待定、其他接口

# 升级造成的兼容性问题

兼容性问题有一下几种情况：

* license系统默认配置项更新，需要升级动态链接库（目前仅支持的系统默认配置项不在更新，可忽略）
* 已经部署产品进行升级对原有license文件是否有效：

如果是产品主版本没有更新，则可以完全兼容；

如果是主版本更新则需要重新生成license文件（是否在默认配置项启用主版本验证情况待定）；

* 产品迭代开发中应兼容之前的KV控制项，最好的方式是产品版本和配置版本独立（或名称上叫License版本）

例如，产品1的license版本：配置版本1 --配置项有{k1:v1,k2:v2,…}

且在每个产品实现中，小心的处理好当前配置版本有哪些自定义的配置项需要验证；

如下形式：

If LicenseVersion == “1”{

对比k1

对比k2

} else if LicenseVersion == “1”{

对比k1

对比k2

对比K3

}

* 动态链接库升级（一般是修改bug或者替换公钥）如果是小版本修改bug之类的则不影响，如果是链接库升级公钥则之前以前发布的license文件不能使用，需要重新生成license文件（一般情况很少替换链接库公钥）；

# License文件生成格式



其中文件内容如下：



其中“配置说明”是模板中的配置说明文字，是否显示在license文件中由模板中的“是否明文显示”开关控制；

文中含有的base64编码是系统模板配置项和产品自定义配置项以及原文的签名数据；

License只有已经开启的配置项对其进行base64编码，没有开启的不编码也不显示；

其中显示在license明文的配置项是为了解决市场人员与客户沟通问题，

方便双方知道产品控制项的限制；

# 其他问题与解答

* License Server端考虑到部署问题，可以其采用嵌入式kv数据库，web页面也由go实现，最终程序是一个二进制（web资源打包到二进制中）
* 各产品在没有license情况下，或者license文件删除的情况下，应该是一个功能有限或者不可用的状态；
* Web开发比较麻烦，也可以考虑简单的实现方式，例如根据的xls表直接生成license，其弊端是没有历史数据保留，需要各自产品维护其license版本与二进制对应关系，且不能丢弃以前license版本；
* 安全性，安全性由client端的so动态库接口保证，go调用一般二进制，所以一般能很好的隐藏接口，java需要class进行代码混淆处理，防止反编译出java源代码；
* 考虑更强安全性，签名密钥丢失、泄露都对之前的版本造成影响，补救措施只有升级替换签名私钥，更新包含公钥的连接库（设计中应该把公钥硬编码到连接库，否则公钥独立与链接库，则有可能编码合适的kv控制项，使用非法私钥签名，替换独立公钥文件，也可以验证通过，造成license控制失效）；
* 目录链需要在license中设置switch可使用的连接数的限制，方式

超时限制的对端节点连接地址提交；

* 其他细节问题，在实践中求证以及文档补充；