ПО для реализации лицензирования "LIC-Serv-Agent"

Содержание

1. Общее описание

Сервис "LIC-Serv-Agent" (агент сервера лицензирования или LSA) осуществляет две основные функции:

1) запрос лицензионной ёмкости у сервера лицензирования (LIC-Serv);

2) раздача полученной от LIC-Serv лицензионной ёмкости лицензируемым приложениям.

2. Технические характеристики и системные требования

Системные требования:

1) операционная система CentOS7 (RHEL7-based) или AlmaLinux 8 (RHEL8-based);

2) kernel 3.10 (дефолтовое ядро для RHEL7-based) или kernel 4.18 (дефолтовое ядро для RHEL8-based);

3) наличие на целевом хосте компилятора gcc;

4) локаль en_US.UTF-8;

5) от 1Gb RAM;

6) от 1 vCPU (виртуальный ЦПУ), если запуск планируется на виртуальной машине.

3. Установка, запуск и управление

Первый запуск:

1) создаем (под root-ом) директорию /opt/ext_services (mkdir /opt/ext_services). Директорию можно выбрать на своё усмотрение;

2) меняем владельца директории /opt/ext_services (chown some_user:some_user /opt/ext_services). Опционально;

3) меняем пользователя (su - some_user) на того, которого назначили владельцем директории /opt/ext_services. Опционально;

4) переходим в директорию /opt/ext_services (cd /opt/ext_services);

5) скачиваем архив с ПО -

wget "https://github.com/vladimir-chursin000/sms_sender/raw/master/lic_serv_agent.zip" (для RHEL7-based)

или

wget "https://github.com/vladimir-chursin000/sms_sender/raw/master/lic_serv_agent_alma8.zip" (для RHEL8-based)

(в дальнейшем вводим команды с учётом выбранного архива);

6) даём команду на распаковку архива (unzip lic_serv_agent.zip). В результате распаковки появится директория /opt/ext_services/lic_serv_agent;

7) переходим в директорию /opt/ext_services/lic_serv_agent (cd /opt/ext_services/lic_serv_agent);

8) производим первый запуск ПО посредством команды "./lic_serv_agent_0.exe". В результате на экран терминала будет выведена строка

"**fail [self_lic_serv_cfg_reader:LIC_PARAMS_COMMON]. Param='mac_addr' must be like 'XX-XX-XX-XX-XX-XX'**",

а в директории с исполняемым файлом появится директория cfg_files (/opt/ext_services/lic_serv_agent/cfg_files);

9) переходим в директорию **cfg_files**, открываем на запись файл **lic_serv_agent.cfg**, находим строку "**mac_addr**=some_mac_addr" (блок конфигурации [LIC_PARAMS_COMMON]) и вместо "some_mac_addr" вписываем MAC-адрес хоста в формате "**XX-XX-XX-XX-XX-XX**". Также (в этом же блоке конфигурации) вписываем:

- ip-адрес (параметр "**ip_addr**"), соответствующий указанному ранее MAC-адресу (в рамках сетевого интерфейса);
- уникальный идентификатор пользователя LSA, например, ИНН (параметр "**uniq_client_id**");
- пароль, прописанный на стороне LIC-Serv в блоке [LIC_SHARING_PASSWORD] для LSA с ИНН, равным параметру "**uniq_client_id**" (уникальный идентификатор пользователя LSA), необходимо прописать в параметре **uid_passwd** (в конфиге **lic_serv_agent.cfg**);
- **lic_serv_addr=IP_of_LIC_Serv:UDP_port**, где **IP_of_LIC_Serv –** это **lic_serv_ip** из блока конфигурации [LIC_PARAMS_COMMON] (в файле конфигурации **LIC-Serv**), а UDP_port – udp-порт сервера лицензирования (блок конфигурации LIC-Serv [UDP_PORTS]);
- **lic_serv_node_id**. Сюда вписываем параметр node_id из файла конфигурации LIC-Serv (блок [MAIN]).

10) переходим в директорию с исполнямым файлом ("cd .." или "cd /opt/ext_services/lic_serv_agent") и снова запускаем сервис (./lic_serv_agent_0.exe). В результате сервис lic_serv_agent_0.exe успешно запустится, а в директории /opt/ext_services/lic_serv_agent (корневая директория сервиса) появятся ещё несколько поддиректорий: **cfg_history** (история изменений файла конфигурации sms_sender.cfg), **proc_control** (сервисная папка), **lic_serv_dgram** (сервисная папка с dgram-сокетами), **lic_serv_log** (папка с файлами логов работы сервиса), **actual_lic_info** (директория, содержащая файлы с информацией о текущем использовании полученных от LIC-Serv лицензионных ёмкостей). Также в корневой директории сервиса можно будет увидеть файл ***lic_serv_agent_0.cfg_notice***, который содержит рекомендации по изменению файла конфигурации.

3.1. Смена режима управления. Systemd-mode

Переключение режима управления со стандартного на **systemd-mode**:

1) переходим в директорию сервиса (cd /opt/ext_services/lic_serv_agent);

2) выполняем команду "**./lic_serv_agent_0.exe use_systemd**". В директории сервиса должна появиться поддиректория **for_systemd**, внутри которой будет находиться папка **lic_serv_agent_0**;

3) переходим в директорию "**for_systemd/lic_serv_agent_0**" ("cd for_systemd/lic_serv_agent_0" или "/opt/ext_services/lic_serv_agent/for_systemd/lic_serv_agent_0"). Тут можно обнаружить 3 файла: ***lic_serv_agent_0.service*** (unit-файл для systemd), ***enable_using_systemd.sh*** (скрипт перехода к управлению сервисом через systemd), ***disable_using_systemd.sh*** (скрипт возвращения к стандартному управлению сервисом);

4) правим файл ***lic_serv_agent_0.service***, если это необходимо. Не рекомендуется изменять следующие параметры: Type, ExecStartPre, ExecStart, ExecReload, WantedBy;

5) от имени root (или от имени пользователя с правами, позволяющими конфигурировать systemd) запускаем скрипт ***enable_using_systemd.sh***. Перед запуском необходимо остановить сервис lic_serv_agent_0.exe ("./lic_serv_agent_0.exe stop"), если он был запущен ранее.

После запуска ***enable_using_systemd.sh*** стандартное управление становится недоступно, а управление сервисом будет возможно исключительно через systemd:

1) "**systemctl start lic_serv_agent_0**" – запустить сервис;

2) "**systemctl restart lic_serv_agent_0**" – перезапустить сервис;

3) "**systemctl reload lic_serv_agent_0**" – перечитать файл конфигурации;

4) "**systemctl status lic_serv_agent_0**" – запросить статус;

5) "**systemctl stop lic_serv_agent_0**" – остановить сервис.

Примечание 1. Для перехода обратно на стандартную схему управления необходимо: а) остановить сервис; б) запустить от имени рута скрипт ***disable_using_systemd.sh***.

Примечание 2. Если меняется директория сервиса (например, с "/opt/ext_services/lic_serv_agent" на "/opt/services/lic_serv_agent"), то шаги с 1-го по 5-й потребуется повторить.

4. Конфигурирование

Кконфигурирование агента сервера лицензирования осуществляется посредством правки файла конфигурации "**cfg_files/lic_serv_agent.cfg**" (находится в корневой директории сервиса).

Структура файла конфигурации основана на логических блоках, выделенных с помощью строк вида "[ИМЯ_ЛОГИЧЕСКОГО_БЛОКА]" (1-ая строка блок "открывает", 2-ая – "закрывает"). Также возможно в тексте конфигруационного файла оставлять комментарии и пояснения посредством символа "#" в начале строки.

По умолчанию файл конфигурации содержит поясняющие комментарии касательно возможностей ПО.

4.1. [MAIN]

Блок **содержит ряд настроект, определяющих директории и методы взаимодействия с ПО. Сами настройки представляют собой пары вида "ключ=значение". Каждый параметр имеет описательную часть в виде комментария.**

log_file_write_buf = 1 (или 0). Параметр управления буферизацией вывода для лог-файлов.

node_id=1 (или любое другое число). Идентификатор ПО, присутствующий в имени лог-файла. Также используется для взаимодействия с сервером лицензирования (LIC-Serv).

log_dir=**_STD_DIR_**/lic_serv_log/. Параметр, определяющий директорию для записи лог-файлов. Тег "**_STD_DIR_**" – директория, где расположен исполняемый файл сервиса. Если директория log_dir расположена в _STD_DIR_, то она создаётся автоматически при запуске (если не была создана до этого).

check_logs_fh_timeout=3. Период проверки (в секундах) лог-файлов сервиса. Если лог-файл не существует, то происходит его автоматическое пересоздание.

dgram_dir=**_STD_DIR_**/lic_serv_dgram/. Служебная директория для размещения dgram-сокетов сервиса. Если располагается в _STD_DIR_, то создаётся автоматически.

try2create_serv_dirs_at_start_anyway=0 (или 1). По умолчанию, если служебные директории расположены (через файл конфигурации) в корневой директории сервиса (например, dgram_dir=_STD_DIR_/lic_serv_dgram/), то их создание при старте сервиса происходит автоматически. Если же (через файл конфигурации) указать директорию, отличную от стандартных путей, то создание служебных подпапок – задача администратора сервиса. Поведение сервиса в этом случае можно изменить, если задать try2create_serv_dirs_at_start_anyway=1, не забыв при этом выдать необходимые права (chmod / chown / selinux) на целевую директорию пользователю, из-под которого происходит запуск ПО.

lic_inf_trx_master_password=some_password. Пароль, используемый для осуществления базового шифрования (помимо специальных алгоритмов) при передаче лицензионной информации от агента сервера лицензирования (LIC-Serv-Agent) до сервера лицензирования (LIC-Serv). Данный параметр должен быть идентичен одноимённому параметру в файле конфигурации агента сервера лицензий (LIC-Serv), а длина – не менее 10 символов.

4.2. [UDP_PORTS]

Блок конфигурации, позволяющий задать UDP-порт для агента сервера лицензирования, через который впоследствии происходит взаимодействие с лицензируемыми приложениями.

Формат строки-конфигурации – "**proc_id udp_port**".

Proc_id всегда должен быть равен "0".

```
[UDP_PORTS]
#proc_id        #self_udp_port
0               7777
[UDP_PORTS]
```

4.3. [UDP_ALLOW_IP]

Блок конфигурации позволяет задать ip-адреса (где развёрнуты лицензируемые приложения), с которых разрешено запрашивать лицензионную ёмкость.

Формат - “**proc_id allow_ip**”.

Proc_id всегда должен быть равен “0”.

Варианты для **allow_ip**:

1) **all** (без ограничений);

2) единичный ip-адрес;

3) список ip-адресов, разделённых запятой.

```
[UDP_ALLOW_IP]
#V etom bloke vpisyvaem ip-adresa (ipv4) hostov s licenziruemymi prilozheniyami (naprimer, sms_sender_1.exe),
#kotorym razresheno zaprashivat licenz. emkost.
#
#Esli vmesto ip-adresa (ili spiska adresov cherez zapyatuyu) napisat "all" (bez kavychek), to pervychnyj filtr
#deistvovat ne budet.
###
#proc_id dolzhen byt raven "0".
###
#allow_ip: mozhno zherez zapyatuyu.
###
#proc_id        #allow_ip
0               all
[UDP_ALLOW_IP]
```

4.4. [SOCKET_TIMEOUTS]

Блок конфигурации отвечает за управление поведением сокетов (udp, dgram) в плане скорости обработки входящего трафика. Конфигурация (в рамках блока) представляет собой строку "**proc_id udp_dgram_value**",

где **proc_id** – идентификатор сервиса, который должен быть равен “0”, **udp_dgram_value** – значение таймаута для udp и dgram.

```
[SOCKET_TIMEOUTS]
#Umenshenie taimauta uvelichivaet skorost obrabotki vhodyashego trafika, pri etom uvelichivaetsya
#zagruzka CPU.
###
#proc_id        #udp_dgram
0               0.001
[SOCKET_TIMEOUTS]
```

4.5. [DGRAM_PARAMS]

Блок связан с редактированием параметра dgram-соединения – размера буфера вх. трафика. Конфигурация (в рамках блока) выглядит так – "**proc_id recv_buffer**", где **proc_id** – идентификатор сервиса (должен быть равен “0”), **recv_buffer** – буфер вх. трафика.

```
[DGRAM_PARAMS]
#Esli dlya kakogo-libo proc_id parametry ne opredeleny, to vystavlyautsya def-parametry:
#recv_buffer=16384.
###
#recv_buffer: buffer dlya operacii sokcet->recv.
#Ne rekomenduetsya menya etot parametr bez neobhodimosti.
###
#proc_id        #recv_buffer
0               16384
[DGRAM_PARAMS]
```

4.6. [UDP_PARAMS]

Конфигурация, аналогичная [DGRAM_PARAMS], но для UDP.

4.7. [LIC_PARAMS_COMMON]

Блок **содержит ряд настроект, определяющих параметры идентификации сервера лицензирования и его принадлежность. Сами настройки представляют собой пары вида "ключ=значение". Каждый параметр имеет описательную часть в виде комментария.**

uniq_client_id – уникальный идентификатор пользователя. Рекомендуется использовать индивидуальный номер налогоплательщика (ИНН).

mac_addr – MAC-адрес хоста в формате “XX-XX-XX-XX-XX-XX”, на котором планируется запуск futynf сервера лицензирования (LIC-Serv-Agent).

ip_addr – ip-адрес (ipv4) хоста, на котором планируется запуск агента сервера лицензирования.

uid_passwd – пароль, прописанный на стороне LIC-Serv в блоке [LIC_SHARING_PASSWORD] для LSA с ИНН, равным параметру "**uniq_client_id**" (уникальный идентификатор пользователя LSA).

lic_serv_addr – **IP_of_LIC_Serv:UDP_port**, где **IP_of_LIC_Serv –** это **lic_serv_ip** из блока конфигурации [LIC_PARAMS_COMMON] (в файле конфигурации **LIC-Serv**), а UDP_port – udp-порт сервера лицензирования (блок конфигурации LIC-Serv [UDP_PORTS]);

lic_serv_node_id – параметр node_id из файла конфигурации LIC-Serv (блок [MAIN]).

4.8. [LIC_SHARING_FOR_APP]

Блок конфигурации, позволяющий распределять полученную от LIC-Serv лицензионную ёмкость между лицензируемыми приложениями (на примере SMS-Sender).

Формат строки конфигурации - “**app_name app_ip lic_type lic_count**”.

app_name – имя лицензируемого приложения.

app_ip – ip-адрес (v4) хоста, где планируется запуск лицензируемого приложения.

lic_type – тип лицензионноё ёмкости, необходимой для работы приложения (например, sms для сервисов SMS-Sender).

lic_count – количество лицензионных единиц, выделяемых на приложение.

Пример конфигурации.

```
[LIC_SHARING_FOR_APP]
#app_name - imya licenziruemogo prilozheniya. Esli imya prilozheniya soderzhit probely, to neobhodimo
#ispolzovat dvojnye kavychki, naprimer, "some app".
###
#app_ip - ipv4.
###
#lic_type - tip licenziruemoj emkosti (ili imya licenzii).
###
#lic_count - licenzionnaya emkost, vydelyaemaya dlya prilozheniya.
###
#app_name       #app_ip         #lic_type       #lic_count
sms_sender_1    192.168.122.2   sms     70
sms_sender_2    192.168.122.2   sms     40
sms_sender_1rx  192.168.122.2   sms     1
sms_sender_1dsm 192.168.122.2   sms     1
[LIC_SHARING_FOR_APP]
```

5. Логирование. Общие сведения

По умолчанию лог-файлы пишутся в директорию, определяемую параметром **log_dir** (блок конфигурации **[MAIN]**), который возможно переопределить вручную (с условием, что будут выданы права на запись пользователю, под которым планируется запускать сервис).

Формат имени основного лог-файла - “**DATE_node_id_serv_name**.log”, где *DATE* – текущая дата, *node_id* – одноимённый параметр из блока *[MAIN]*, *serv_name* – имя сервиса без расширения “.exe” (lic_serv_agent_0).

Строка в лог-файле (в общем случае) представляет собой конструкцию вида "**LOG_TYPE**; +**HH:MI:SS.mmmmmm**;+**LOG_INFO**" (данные разделены посредством ";+"), где *LOG_TYPE* – тип информационного сообщения (например, info), *HH* – часы, *MI* – минуты, *SS* – секунды, *mmmmm* – микросекунды, *LOG_INFO* – информация.

Возможные варианты *LOG_TYPE*: info, error, udp_info, udp_error, dgram_info, dgram_error.

info – информация общего характера.

error – ошибки общего характера.

udp_info – информация о трафике UDP.

udp_error – ошибки, связанные с UDP-трафиком.

dgram_info – информация о DGRAM-трафике.

dgram_error – ошибки, связанные с DGRAM-трафиком.

5.1. Логирование. UDP-трафик

Формат - “**LOG_TYPE**;+**HH:MI:SS.mmmmmm**;+**op**;+**host**;+**port**;+**udp_msg**;+**err_txt**” (значения разделены посредством “;+”).

op – тип операции. Возможные варианты (в рамках сервиса): *send_ans* (отправка ответа на поступишвий запрос), *get_req* (получение запроса).

host – адрес хоста, принявшего / отправившего udp-пакет. В рамках сервиса значение всегда равно адресу хоста, отправившего udp-пакет.

port – порт на стороне хоста, принявшего udp-пакет. В рамках сервиса выставляется стандартное значение "N" (т.е. No), т.к. сервис не отправляет запросы, а только принимает и отвечает.

udp_msg – содержание udp-пакета (скрыто).

err_txt – текст ошибки или "OK", если ошибки отсутствуют.

5.2. Логирование. DGRAM-трафик

Формат – “**LOG_TYPE**;+**HH:MI:SS.mmmmmm**;+**op**;+**dgram_file**;+**dgram_msg**;+**err_txt**” (значения разделены посредством “;+”).

op – тип операции. Возможные варианты: get_req (получение информации), send_str (отправка информции), new_dgram (подключение к dgram-сокету), close_dgram (окончание передачи).

dgram_file – DGRAM-файл (сокет), принявший dgram-пакет (или на который был отправлен dgram-пакет).

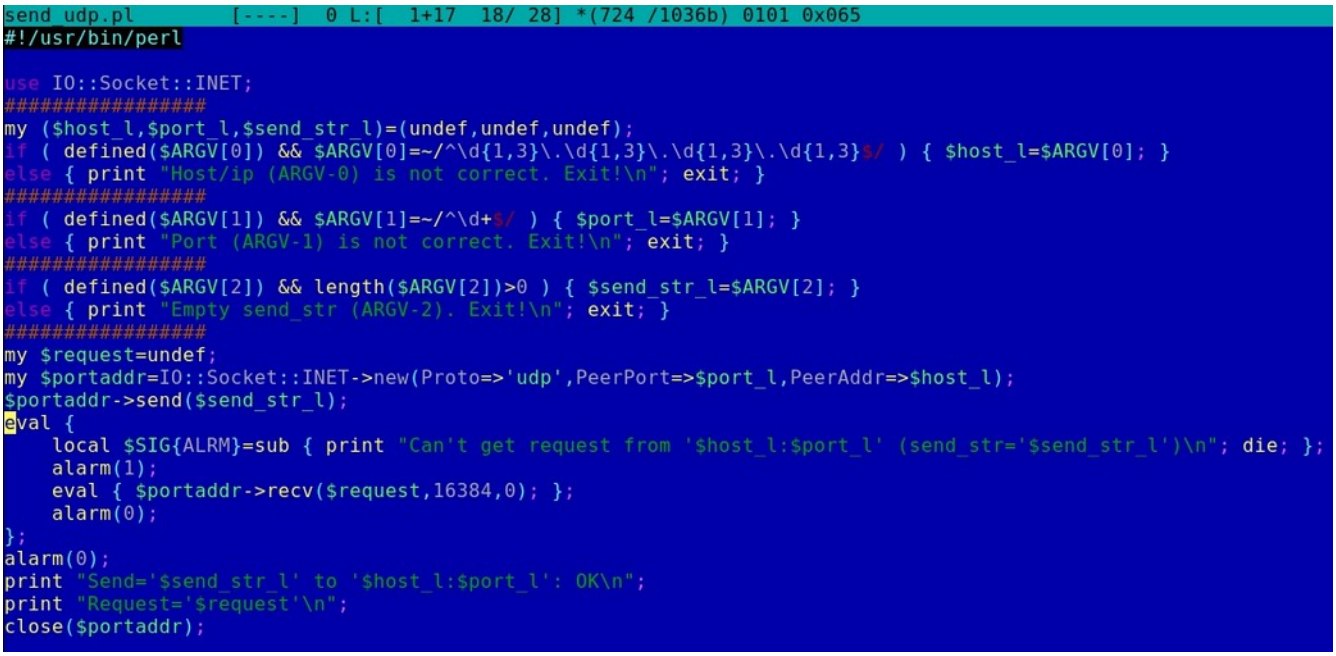
dgram_msg – содержание dgram-пакета (скрыто).

err_txt – текст ошибки или "OK", если ошибки отсутствуют.

6. UDP-запросы к сервисам

1) ping. Отправлять на порт, указанный в блоке конфигурации [UDP_PORTS]. Если сервис активен, то в ответ получим "OK".

Пример скрипта для отправки UDP.



```
send_udp.pl        [----]  0 L:[  1+17  18/ 28] *(724 /1036b) 0101 0x065
#!/usr/bin/perl

use IO::Socket::INET;
#################
my ($host_l,$port_l,$send_str_l)=(undef,undef,undef);
if ( defined($ARGV[0]) && $ARGV[0]=~/^\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}$/ ) { $host_l=$ARGV[0]; }
else { print "Host/ip (ARGV-0) is not correct. Exit!\n"; exit; }
#################
if ( defined($ARGV[1]) && $ARGV[1]=~/^\d+$/ ) { $port_l=$ARGV[1]; }
else { print "Port (ARGV-1) is not correct. Exit!\n"; exit; }
#################
if ( defined($ARGV[2]) && length($ARGV[2])>0 ) { $send_str_l=$ARGV[2]; }
else { print "Empty send_str (ARGV-2). Exit!\n"; exit; }
#################
my $request=undef;
my $portaddr=IO::Socket::INET->new(Proto=>'udp',PeerPort=>$port_l,PeerAddr=>$host_l);
$portaddr->send($send_str_l);
eval {
    local $SIG{ALRM}=sub { print "Can't get request from '$host_l:$port_l' (send_str='$send_str_l')\n"; die; };
    alarm(1);
    eval { $portaddr->recv($request,16384,0); };
    alarm(0);
};
alarm(0);
print "Send='$send_str_l' to '$host_l:$port_l': OK\n";
print "Request='$request'\n";
close($portaddr);
```

7. DGRAM-запросы к сервисам

Механика запросов аналогична UDP-запросам, но с учётом специфики взаимодействия с DGRAM-сокетами (необходимо открывать как сокет для отправки данных, так и сокет для получения ответа).

Пример скрипта для отправки DGRAM.

```
send_dgram.pl      [----]  0 L:[  1+ 0   1/ 30] *(0   /1150b) 0035 0x023
#!/usr/bin/perl
use strict;
use warnings;
use IO::Socket::UNIX;
my ($send_str_l,$send_dgram_name)=(undef,undef);
if ( defined($ARGV[1]) && length($ARGV[1])>0 ) { $send_str_l=$ARGV[1]; }
else { print "Empty send_str (ARGV-1). Exit!\n"; exit; }
if ( defined($ARGV[0]) && $ARGV[0]=~/^sms_sender_/ ) { $send_dgram_name=$ARGV[0]; }
else { print "Empty send_dgram_name (ARGV-0). Exit! \n"; exit; }
my $dgram_file_s='/home/export/services/sms_sender/work/sms_sender_dgram/'.$send_dgram_name;
my $dgram_file_r='/tmp/tmp_sock.dgram';
my $request=undef;
my $portaddr_s=IO::Socket::UNIX->new(Peer=>$dgram_file_s,Type=>SOCK_DGRAM,Proto=>0);
my $portaddr_r=IO::Socket::UNIX->new(Local=>$dgram_file_r,Type=>SOCK_DGRAM,Proto=>0);
$portaddr_s->send($send_str_l.'->'.$dgram_file_r);
print "Send='$send_str_l' to '$dgram_file_s': OK\n";
close($portaddr_s);
eval {
    local $SIG{ALRM}=sub { print "Can't get request from '$dgram_file_r' (send_str='$send_str_l')\n"; die; };
    alarm(1);
    eval {
        $portaddr_r->recv($request,16384,0);
    };
    alarm(0);
};
alarm(0);
print "Get request='$request' from '$dgram_file_r'\n";
close($portaddr_r);
unlink $dgram_file_r;
```

8. Статус-файл об использовании лицензионной ёмкости приложениями

Статус-файл располагается в директории “**_STD_DIR_/actual_lic_info**”, имеет имя “**actual_info.txt**” и обновляется по факту изменения информации об использовании лицензионной ёмкости (полученной от LIC-Serv), например, было запущено лицензируемое приложение, которое задействовало предоставленную ему ёмкость.

Файл представляет собой блоки с привязкой к типу лицензии, разделённые строкой из ряда символов “#”. Каждый блок содержит информацию трёх типов (на примере типа лицензии “sms”):

1) Общая информация о типе лицензии.

all_cnt – суммарная лицензионная ёмкость, полученная от LIC_Serv.

now_in_use_cnt – используемая в данный момент лицензионная ёмкость.

free_cnt – свободная на данный момент лицензионная ёмкость.

2) Информация о таймерах изолированного использования лицензионной ёмкости (если связь с сервером лицензирования была потеряна).

standalone_time – время сохранения лицензионной ёмкости на стороне LIC-Serv-Agent, если связь с сервером лицензирования была потеряна.

last_success_get_lic_dt – дата и время последнего обращения к LIC-Serv за актуальной информацией о выделенной лицензионной ёмкости.

expire_standalone_lic_dt – дата и время устаревания изолированного использования выделенной лицензионной ёмкости (если в данный момент пропадёт сетевая связанность с сервером лицензирования).

3) Информация о лицензируемых приложениях, получивших лицензионную ёмкость.

9. Используемые в ПО модули Perl5

1. IO::Socket::INET.
2. IO::Socket::UNIX.
3. IO::Select.
4. Encode.
5. POSIX.
6. Time::HiRes.
7. Data::Dumper.
8. Cwd.
9. FileHandle.
10. Time::Local.
11. List::MoreUtils.
12. Crypt::Blowfish.
13. Crypt::CBC.
14. Crypt::RSA.
15. Crypt::RSA::Key::Public.
16. Crypt::RSA::Errorhandler.
17. Crypt::RSA::Key.
18. Crypt::RSA::DataFormat.
19. Crypt::RSA::ES::OAEP.
20. Crypt::Primes.
21. Crypt::Random.
22. Digest::MD2.
23. Digest::MD5.
24. Digest::SHA1.
25. Class::Loader.
26. Convert::ASCII::Armour.

27. Data::Buffer.
28. Math::Pari.
29. Sort::Versions.
30. Tie::EncryptedHash.
31. Carp.
32. FindBin.
33. Storable.
34. Compress::Bzip2.
35. PAR::Packer.