

Проектные спецификации

Библиотека лицензирования

1 страниц

Дата: 03.07.2003

Версия: 0.1

Статус: В разработке

Содержание

1. Список изменений	3
2. Введение.....	4
2.1 Цель.....	4
2.2 Контекст	4
2.3 Определение, акронимы и сокращения.....	4
2.4 Ссылки	4
2.5 Общая концепция библиотеки	4
2.6 Функции библиотеки	4
3. Реализация библиотеки	4
3.1 Общие сведения.....	4
3.2 Принципы работы библиотеки лицензирования	5
3.2.1 Общие сведения	5
3.2.2 Входная информация для библиотеки лицензирования.....	5
3.2.3 Хранилище данных библиотеки лицензирования	5
3.2.3.1 Общие сведения	5
3.2.4 Управление лицензионными ключами	6
3.2.4.1 Общие сведения	6

3.2.4.2	Хранение лицензионных ключей	6
3.2.4.3	Совместимость с ключами предыдущих версий продуктов	6
3.3	Поддерживаемые платформы	6
3.3.1	Перечень платформ	6
3.3.2	Требования к компиляторам	7
3.4	Используемые при реализации библиотеки внешние решения	7
3.5	Программные интерфейсы	7
3.5.1	Общие определения.....	8
3.5.2	Интерфейс библиотеки лицензирования на C++	9
3.5.3	Интерфейс доступа к хранилищу данных библиотеки лицензирования	14
3.5.4	Лицензионная информация, возвращаемая библиотекой лицензирования	16
3.5.5	Информация о приложении	16
3.5.6	Интерфейс библиотеки лицензирования на языке C	18

1. Список изменений

Версия	Дата	Изменения	Кто сделал
0.0	03.07.2003	Документ создан	Зубрилин С.А.
0.1	07.07.2003	Уточнены программные интерфейсы, требования к среде компиляции	Зубрилин С.А.

2. Введение

2.1 Цель

2.2 Контекст

2.3 Определение, акронимы и сокращения

2.4 Ссылки

2.5 Общая концепция библиотеки

Библиотека лицензирования предназначена для проверки соблюдения лицензионного соглашения пользователем, использующим продукты Лаборатории Касперского.

Библиотека лицензирования разрабатывается для использования в качестве единого механизма проверки лицензии для всех продуктов Лаборатории Касперского.

2.6 Функции библиотеки

Библиотека лицензирования выполняет следующие функции:

- проверяет указанный лицензионный ключ на соответствие лицензионному соглашению и по результатам проверки возвращает приложению информацию о режиме его работы. Проверка включает в себя следующие действия:
 - проверку срока действия лицензионного ключа;
 - проверку наличия серийного номера лицензионного ключа в «черном списке» (black.lst);
 - проверку факта ручной замены антивирусных баз данных после окончания срока действия ключа по файлу описания обновления kavset.xml;
 - возвращает полную информацию из любого указанного ей лицензионного ключа;
 - регистрирует устанавливаемые лицензионные ключи;
 - ведет базу данных с историей когда-либо зарегистрированных ключей для данного приложения в течение его жизненного цикла на данном компьютере.
-

3. Реализация библиотеки

3.1 Общие сведения

Библиотека лицензирования разрабатывается в виде статической библиотеки, линкуемой при сборке к каждому модулю, запрашивающему проверку выполнения лицензионного соглашения.

Язык реализации библиотеки – C++. Для приложений, написанных на языке C, предоставлен интерфейс библиотеки на этом языке.

3.2 Принципы работы библиотеки лицензирования

3.2.1 Общие сведения

Библиотека лицензирования осуществляет проверку соблюдения пользователем лицензионного соглашения путем сопоставления информации из лицензионного ключа с информацией о приложении, запросившем проверку лицензии.

Лицензионный ключ поставляется пользователю в виде отдельного файла. Лицензионный ключ должен быть явно зарегистрирован пользователем, в противном случае он не учитывается при определении режима работы приложения.

Информация о приложении внедряется на уровне исходного кода в каждый модуль приложения, который будет самостоятельно запрашивать проверку лицензии (формат и состав информации о приложении приведен в разделе [Информация о приложении](#)). Библиотека лицензирования статически линкуется с каждым таким модулем.

На приложение возлагается обязанность предоставить библиотеке лицензирования:

- некоторые настройки, необходимые для работы библиотеки (см. [Входная информация для библиотеки лицензирования](#));
 - процедуры сохранения и загрузки данных библиотеки лицензирования.
-

3.2.2 Входная информация для библиотеки лицензирования

Перед началом работы библиотека лицензирования требует инициализации. Перечень параметров инициализации приведен ниже:

- процедуры доступа к хранилищу данных библиотеки лицензирования (см. [Хранилище данных библиотеки лицензирования](#));
 - реализация интерфейса к конкретному используемому XML parser'у. Эта реализация предоставлена библиотекой, вызывающий код должен только выбрать желаемую реализацию из имеющихся и передать ее библиотеке;
 - список путей к антивирусным базам;
 - режим проверки (полный, включая проверку факта подмены антивирусных баз и проверку ключа по «черному списку», или упрощенный, заключающийся в проверке даты истечения срока действия ключа).
-

3.2.3 Хранилище данных библиотеки лицензирования

3.2.3.1 Общие сведения

Библиотека лицензирования ведет базу данных зарегистрированных ключей, которую ей нужно сохранять между сеансами своей работы. Учитывая, что на различных платформах и у

различных приложений способы хранения этой информации отличаются, библиотека лицензирования не содержит собственного механизма хранения этих данных. Вместо этого библиотека определяет интерфейс сохранения и загрузки данных, реализация которого возлагается на приложение (описание интерфейса находится в разделе [Интерфейс доступа к хранилищу данных библиотеки лицензирования](#)). При инициализации библиотеки приложение предоставляет ей эту реализацию. Приложение должно воспринимать записываемые и считываемые библиотекой данные как простой байтовый поток и не должно каким-либо образом интерпретировать или изменять их.

Данные библиотеки лицензирования содержат критическую для обеспечения выполнения лицензионного соглашения информацию, поэтому на библиотеку возлагается ответственность за сохранение своих данных в виде, обеспечивающем их сохранность и неизменность.

3.2.4 Управление лицензионными ключами

3.2.4.1 Общие сведения

Использование лицензионного ключа невозможно без его регистрации в библиотеке лицензирования.

В каждый момент времени может быть установлено не более двух ключей: один из них является активным, второй является резервным с датой установки (датой начала действия), равной дате окончания действия активного ключа. По истечении срока действия активного ключа резервный ключ автоматически становится активным.

3.2.4.2 Хранение лицензионных ключей

При регистрации лицензионного ключа библиотека лицензирования сохраняет тело ключа (образ ключевого файла) в своем хранилище данных. Библиотека лицензирования проверяет активный ключ в своем хранилище и не требует наличия соответствующего ключевого файла на диске. Тело резервного лицензионного ключа также содержится в хранилище данных библиотеки.

3.2.4.3 Совместимость с ключами предыдущих версий продуктов

Библиотека лицензирования обеспечивает совместимость с ключами предыдущих версий продуктов Лаборатории Касперского прозрачным для приложения образом. Ключи предыдущих версий устанавливаются наравне с новыми в соответствии с существующими правилами определения сроков жизни и сроков действия лицензионных ключей.

Библиотека лицензирования обеспечивает импорт информации об установленных ключах предыдущих версий продуктов по запросу использующего ее модуля.

3.3 Поддерживаемые платформы

3.3.1 Перечень платформ

Библиотека лицензирования разрабатывается для работы в составе приложений на следующих платформах:

- Microsoft Windows 9x/Millennium/NT 4.0/2000/XP
- Novell Netware

- FreeBSD
- Linux
- OpenBSD

Библиотека переносима на все перечисленные платформы на уровне исходного кода.

3.3.2 Требования к компиляторам

Для сборки библиотеки на различных платформах необходимо использовать следующие компиляторы:

Платформа	Компилятор
Microsoft Windows	Microsoft Visual C++ 6.0 Service Pack 5
Novell	PDK for Netware 5.0
FreeBSD	gcc 3.2.2 или выше
Linux	gcc 3.2.2 или выше
OpenBSD	gcc 2.95.3 или выше

3.4 Используемые при реализации библиотеки внешние решения

При реализации библиотеки лицензирования используются следующие готовые решения Лаборатории Касперского:

- библиотека работы с деревом (property.lib);
- библиотека преобразования дерева в последовательную форму (kldtser.lib) ;
- библиотека проверки цифровой подписи файлов (sign.lib).

Используются следующие готовые решения сторонних производителей:

- Microsoft XML Parser 2.5 или выше (только на Windows-платформах);
 - Expat XML Parser (на unix-платформах и под Novell);
 - Arabica XML Parsers wrapper;
 - STLPort 5.0.
-

3.5 Программные интерфейсы

Ниже приведены программные интерфейсы и определения структур данных библиотеки лицензирования.

3.5.1 Общие определения

```
#if !defined (__LIC_DEFS_H__)
#define __LIC_DEFS_H__

#if defined (_MSC_VER)
    // disable warning 4786
    // 'identifier' : identifier was truncated to 'number' characters in the
    // debug information
    #pragma warning (disable : 4786)
    #pragma once
#endif

//-----

#if defined (WIN32)
    #include <tchar.h>
    #include <windows.h>
#endif

#include <string>
#include <vector>

#include <err_defs.h>

//-----

#if defined (_MSC_VER)
    #pragma warning (push, 4)
#endif

namespace LicensingPolicy {

//-----

typedef unsigned long      dword_t;
typedef unsigned short     word_t;
typedef std::basic_string<_TCHAR> string_t;
typedef std::vector<string_t> string_list_t;
typedef std::string        file_image_t;

struct date_t
{
    int operator < (const date_t& d) const;
    int operator == (const date_t& d) const;

    unsigned int year;
    unsigned int month;
    unsigned int day;
};
```



```

enum functionality_level_t
{
    flUnknown = 0,
    flNoFeatures,
    flNoUpdates,
    flFullFunctionality
};

//-----

}; // namespace LicensingPolicy

#ifdef (_MSC_VER)
    #pragma warning (pop)
#endif

#endif

```

3.5.2 Интерфейс библиотеки лицензирования на C++

```

#ifdef (__LICENSING_INTF_H__)
#define __LICENSING_INTF_H__

#ifdef (_MSC_VER)
    // disable warning 4786
    // 'identifier' : identifier was truncated to 'number' characters in the
    // debug information
    #pragma warning (disable : 4786)
    #pragma once
#endif

//-----

#include <string>
#include <vector>

#include <datatree.h>
#include <keyfile.h>
#include <blacklist.h>
#include <lic_storage_intf.h>
#include <xmlhelper.h>

#include <appinfo_struct.h>

//-----

#ifdef (_MSC_VER)
    #pragma warning (push, 4)
#endif

```

```

//-----

namespace LicensingPolicy {

using helpers::IXMLHelper;

//-----

class ILicensingPolicy
{
public:

    //! Possible key invalidity reason
    enum key_invalid_reason_t
    {
        kirUnknown    = 0,
        kirValid,
        kirExpired,
        kirCorrupted,
        kirNotSigned,
        kirWrongProduct,
        kirNotFound,
        kirBlackListed,
        kirTrialAlreadyUsed
    };

    //! Checking modes
    enum checking_modes_t
    {
        cmUnknown      = 0,
        cmFullCheck     = 1,          //!< Do bases update checking using kavset.xml
                                   //!< and key serial number checking against black.lst
        cmKeyOnly       = 2          //!< Check key against system date only
    };

    //! Licensing key checking results
    struct check_info_t
    {
        string_t        keyFileName;    //!< Key file name (without path)
        key_info_t       keyInfo;       //!< Info from key file
        key_invalid_reason_t invalidReason;  //!< Key invalidity reason (or kirValid)
        date_t          keyExpirationDate;  //!< Key expiration date
        date_t          appLicenseExpDate;  //!< App license expiration date
        dword_t         daysTillExpiration;  //!< Days left till app expiration day
        functionality_level_t funcLevel;    //!< Application functionality level provided by this
key
    };

    typedef std::vector<key_info_t>    key_info_list_t;
    typedef string_list_t             file_list_t;

```

```

typedef unsigned int                product_id_t;

typedef std::pair<string_t, check_info_t> check_info_pair_t;
typedef std::map<string_t, check_info_t> check_info_list_t;

//! Library initialization method. All methods will return E_FAIL if init hasn't been called yet
virtual HRESULT    init (ILicenseStorage    *licenseStorage,
                        const _app_info_t&  appInfo,
                        void                *context,
                        const string_list_t& basesPaths,
                        checking_mode_t     checkMode = cmFullCheck
                        ) = 0;

//! Sets news app info struct
virtual HRESULT    setAppInfo (const _app_info_t& appInfo) = 0;
virtual HRESULT    setBasesPaths    (const string_list_t& basesPaths) = 0;

// ----- Licensing checking routines -----

//! Checks application licensing mode against active key
virtual HRESULT    checkActiveKey    (check_info_t *pCheckInfo) = 0;

//! Returns licensing checking info for pointed key file
virtual HRESULT    checkKeyFile      (const string_t&    keyFileName,
                                     check_info_t       *checkInfo
                                     ) = 0;

//! Returns licensing checking info for key file preloaded from disk
virtual HRESULT    checkKeyInMemory  (const string_t&    keyFileName,
                                     const char         *fileBody,
                                     size_t              fileSize,
                                     check_info_t       *checkInfo
                                     ) = 0;

//! Checks if active key serial number found in provided black list
virtual HRESULT    checkBlacklistFile (const string_t& blacklistFileName) = 0;

    //! Returns licensing checking info for key files found in pointed location
virtual HRESULT    checkKeyFiles     (const string_t&    dir,
                                     check_info_list_t *checkList
                                     ) = 0;

//! Checks if active key serial number found in provided black list
virtual HRESULT    checkBlacklist    (const char *blackList,
                                     size_t      size
                                     ) = 0;

//! Check arbitrary license key against arbitrary black list
virtual HRESULT    checkKeyBlacklist (const string_t&    keyFileName,
                                     const string_t&    blacklistFileName
                                     ) = 0;

virtual HRESULT    checkKeyBlacklist (const char        *keyFileBody,

```

```

        size_t      keyFileSize,
        const char  *blacklistBody,
        size_t      blacklistSize
    ) = 0;

// ----- Licensing managment routines -----

//! New key registration
virtual HRESULT    addKey      (const string_t& fileName) throw () = 0;
virtual HRESULT    addKey      (const string_t& fileName,
                                const char  *fileBody,
                                size_t      fileSize
                                ) = 0;

//! New key registration with explicit black list checking
virtual HRESULT    addKeyBlacklist (const string_t& keyFileName,
                                    const char  *keyFileBody,
                                    size_t      keyFileSize,
                                    const char  *blacklistBody,
                                    size_t      blacklistSize
                                    ) = 0;
virtual HRESULT    addKeyBlacklist (const string_t& fileName,
                                    const string_t& blacklistFileName
                                    ) = 0;

//! Active key deletion
virtual HRESULT    revokeActiveKey () = 0;
//! Registered key deletion using key serial number
virtual HRESULT    revokeKey      (const key_serial_number_t& serNum) = 0;

//! Reads information about old format keys
virtual HRESULT    importLegacyKeysInfo(installed_keys_list_t *keysList) = 0;

// ----- Licensing managment helper routines -----

//! Returns active licensing key info
virtual HRESULT    getActiveKeyInfo (key_info_t *keyInfo) = 0;

//! Returns info about installed keys alone with licensing chcking info
virtual HRESULT    getInstalledKeysInfo(check_info_list_t *checkInfo) = 0;

//! Returns licensing key info
virtual HRESULT    getKeyFileInfo (const string_t& fileName,
                                   key_info_t  *keyInfo
                                   ) = 0;

//! Returns info from license keys found in pointed dir
virtual HRESULT    getKeyFilesInfo (const string_t& dir,
                                   key_info_list_t *keysList
                                   ) = 0;

virtual HRESULT    checkFileSign    (const string_t& fileName) = 0;

```

```

virtual HRESULT validateSign (const string_t& fileName,
                             const std::string& fileSign
                             ) = 0;

//! Returns stored system error. Meaningfull only when LIC_E_SYS_ERROR returned
virtual sys_err_code_t getStoredError () = 0;

//! Returns string description for key invalidity code
virtual HRESULT getKeyInvalidReasonStr (key_invalid_reason_t invalidReason,
                                       string_t *message
                                       ) = 0;

}; // class CLicensingIntf

//-----

}; // namespace Licensing

#ifdef (_MSC_VER)
#pragma warning (pop)
#endif

#endif

```

Название	Комментарий
init	Процедура инициализации библиотеки. До вызова этой процедуры работа с библиотекой невозможна.
checkActiveKey	Проверяет соблюдение лицензионного соглашения, сопоставляя информацию о приложении с данными из активного ключа. Возвращает приложению режим работы, соответствующий этому ключу.
checkKeyFile	Проверяет соблюдение лицензионного соглашения, сопоставляя информацию о приложении с указанным ключевым файлом на диске. Возвращает приложению режим работы этого приложения, обеспечиваемый при установке этого ключа в качестве активного. Имя ключевого файла должно быть задано полностью.
checkKeyFiles	Выполняет аналогичную checkKeyFile проверку, но для каждого найденного в указанном каталоге лицензионного файла. Возвращает приложению информацию по результатам проверки каждого файла.
checkBlacklist	Проверяет наличие активного лицензионного ключа в заданном файле «черного списка». Процедура получает образ дискового файла.
addKey (const string_t& fileName)	Процедура регистрации нового лицензионного ключа. Ключ считывается библиотекой лицензирования из указанного дискового файла.
addKey (const string_t& filename, const char *fileBody, size_t fileSize)	Процедура регистрации нового лицензионного ключа. Ключ передается в виде образа дискового файла. Имя файла передается для сохранения его в хранилище библиотеки лицензирования и отображения его пользователю в справочных целях. Библиотека не пытается читать файл с этим именем с диска.

revokeActiveKey	Отменяет использование текущего активного ключа. Если при этом установлен резервный ключ, его установка автоматически отменяется, при этом в базе зарегистрированных ключей стирается информация об этом ключе. С этого момента библиотека лицензирования считает, что этот ключ никогда не регистрировался.
revokeKey	Отменяет использование лицензионного ключа с указанным серийным номером. Если указан серийный номер активного ключа, выполняется процедура revokeActiveKey.
getActiveKeyInfo	Возвращает полную информацию из активного ключа. Формат и состав информации приведен в разделе Лицензионная информация, возвращаемая библиотекой лицензирования .
getInstalledKeysInfo	Возвращает полную информацию из всех установленных ключей. Формат и состав информации приведен в разделе Лицензионная информация, возвращаемая библиотекой лицензирования .
getKeyFileInfo	Возвращает полную лицензионную информацию из указанного ключевого файла.
getKeyFilesInfo	Возвращает полную лицензионную информацию из всех ключевых файлов, найденных в указанном каталоге.
checkFileSign	Выполняет проверку цифровой подписи.
setBasesPaths	Устанавливает новое значение пути к антивирусным базам
setAppInfo	Задаёт новое описание продукта
checkKeyInMemory	Проверяет лицензионный ключ, переданный в виде образа файла
checkBlacklistFile	Проверяет серийный номер активного лицензионного ключа на присутствие в заданном файле «черного списка»
checkBlacklist	Проверяет серийный номер активного лицензионного ключа на присутствие в заданном файле «черного списка». «Черный список» задается образом файла
checkKeyBlacklist	Проверяет серийный номер заданного лицензионного ключа на присутствие в заданном файле «черного списка».
addKeyBlacklist	Регистрирует новый лицензионный ключ, проверяя его по заданному файлу «черного списка».
checkFileSign	Проверяет цифровую подпись заданного файла
validateSign	Проверяет соответствие образцу цифровой подписи заданного файла
getStoredError	Возвращает код последней запомненной системной ошибки.
getKeyInvalidReasonStr	Возвращает строку описания ошибки по ее коду.

3.5.3 Интерфейс доступа к хранилищу данных библиотеки лицензирования

```
#if !defined (__LIC_STORAGE_H__)
#define __LIC_STORAGE_H__
```

```
#if defined (_MSC_VER)
#pragma warning (disable : 4786)
#pragma once
#endif
```

```
//-----
```

```
#if defined (_MSC_VER)
#pragma warning (push, 4)
#endif
```

```
//-----
```

```
typedef int (*READ_SECURE_DATA_PROC) (char **data, unsigned int *size, void
*context);
typedef int (*WRITE_SECURE_DATA_PROC) (const char *data, unsigned int size, void
*context);
```

```
typedef struct
{

    READ_SECURE_DATA_PROC  readSecureData;
    WRITE_SECURE_DATA_PROC  writeSecureData;

} ILicenseStorage;
```

```
//-----
```

```
#if defined (_MSC_VER)
#pragma warning (pop)
#endif
```

```
#endif
```

Название	Комментарий
READ_SECURE_DATA_PROC	Указатель на функцию, возвращающую данные из хранилища библиотеки лицензирования. Память под данные выделяется этой функцией с помощью функции стандартной библиотеки C malloc, освобождается библиотекой лицензирования. Возвращает количество прочитанных байт или отрицательное число в случае возникновения ошибки.
WRITE_SECURE_DATA_PROC	Указатель на функцию, записывающую данные в хранилище библиотеки лицензирования. Память освобождается самой библиотекой лицензирования. Возвращает количество записанных байт или отрицательное число в случае возникновения ошибки.

3.5.4 Лицензионная информация, возвращаемая библиотекой лицензирования

```
struct key_info_t
{
    dword_t      keyVer;           //!< key file structure version
    date_t       keyCreationDate;  //!< key file creation date
    key_serial_number_t keySerNum;  //!< key file serial number
    key_type_t    keyType;         //!< key type (commercial, trial, ...)
    dword_t      keyLifeSpan;      //!< key validity period since installation (days)
    date_t       keyExpDate;       //!< key limit validity period (days)
    dword_t      licenseCount;     //!< license number (key ver 1-3)
    string_t      productName;     //!< product name
    dword_t      appId;            //!< application name
    dword_t      productId;        //!< product id
    string_t      productVer;      //!< major product version
    licensed_object_list_t licensedObjLst; //!< licensed objects with license count
    string_t      licenseInfo;     //!< descriptive license info
    string_t      supportInfo;     //!< support company properties
    dword_t      marketSector;    //!< market sector
    components_level_list_t compFuncLevel; //!< components functionality levels list
    string_t      customer_info;   //!< customer info
};
```

3.5.5 Информация о приложении

```
#if !defined (__APPINFO_STRUCT_H__)
#define __APPINFO_STRUCT_H__

#define uint unsigned int

// Component description
typedef struct
{
    uint componentId;           // component id
    uint funcLevel;             // component functionality level
} _component_info_t;

// Components list
typedef struct
{
    uint      componentsNumber; // number of list elements
    _component_info_t *componentList; // pointer to components info vector
} _component_list_t;

// Product description
typedef struct
{
    uint      prodId;           // product id
    char      *prodName;        // product name
    char      *prodMajorVer;    // product major version
```



```

    uint        publicKeyId;    // public key id
} _product_info_t;

// Products list
typedef struct
{
    uint        productsNumber;  // number of list elements
    _product_info_t *productsList;  // pointer to products info vector
} _product_list_t;

// Public key GUID description
typedef struct
{
    uint        ID;
    char        *account;        // developer account
    char        *devGUID;        // developer id
    char        *pubStr1;
    char        *pubStr2;
    char        *pubStr3;
    char        *pubStr4;
} _pub_key_info_t;

// GUIDs list
typedef struct
{
    uint        GUID_number;     // number of list elements
    _pub_key_info_t *GUID_list;  // pointer to GUIDs info vector
} _pub_key_info_list_t;

// Old application compatibility information
typedef struct
{
    uint        appId;           // application id
    char        *appName;        // application name
} _app_compat_info_t;

// Old application compatibility list
typedef struct
{
    uint        appsNumber;      // number of list elements
    _app_compat_info_t *appsList;  // Compatible applications list
} _app_compat_list_t;

typedef struct
{
    unsigned int year;
    unsigned int month;
    unsigned int day;
} _date_t;

typedef enum
{

```

```

aitUnknown = 0,
aitApplicationInfo,
aitComponentInfo
} _app_info_type_t;

// Component functionality level bit mask in particular app context
typedef struct
{
    uint    appId;           // id of application, which can contain component
    uint    funcLevelMask;   // component functionality level bit mask for this app
} _app_component_info_t;

typedef struct
{
    uint          appsNumber;   // number of list elements
    _app_component_info_t *list;
} _app_component_list_t;

// Application info

typedef struct
{
    char          *description; // "Kaspersky lab application info"
    uint          appInfoVer;   // application info structure version
    _date_t       appInfoDate;  // application info creation date
    _app_info_type_t    infoType; // application info type (app or component)
    uint          id ;          // app or component id
    char          *name;        // app or componet name
    char          *ver;         // app or componet version
    int           isRelease;     // 0 - beta, not 0 - release
    _component_list_t    compList; // components list (app only)
    _app_component_list_t appList; // applications list (component only)
    _product_list_t    productList; // products list
    _pub_key_info_list_t    GUID_list; // public key GUIDs list
    _app_compat_list_t    appCompatList; // old applications compatibility info
} _app_info_t;

#endif

```

3.5.6 Интерфейс библиотеки лицензирования на языке C

```

#if !defined (__LICENSING_POLICY_C_INTF_H__)
#define __LICENSING_POLICY_C_INTF_H__

#if defined (_MSC_VER)
    // disable warning 4786
    // 'identifier' : identifier was truncated to 'number' characters in the
    // debug information

```

```
#pragma warning (disable : 4786)
```

```
#pragma once
```

```
#endif
```

```
#include <lic_storage_intf.h>
```

```
#include <err_defs.h>
```

```
#include <appinfo_struct.h>
```

```
#if defined (_MSC_VER)
```

```
    #pragma warning (push, 4)
```

```
#endif
```

```
#if defined (__cplusplus)
```

```
extern "C"
```

```
{
```

```
#endif
```

```
/*-----*/
```

```
typedef unsigned int _dword_t;
```

```
typedef struct
```

```
{
```

```
    unsigned int day;
```

```
    unsigned int month;
```

```
    unsigned int year;
```

```
} _date_t;
```

```
typedef struct
```

```
{
```

```
    _dword_t memberId;
```

```
    _dword_t appId;
```

```
    _dword_t serNum;
```

```
} _key_serial_number_t;
```

```
typedef struct
```

```
{
```

```
    _dword_t  objId;
```

```
    _dword_t  licenseNumber;
```

```
} _licensed_object_t;
```

```
typedef struct
```

```
{
```

```
    _dword_t      size;
```

```
    _licensed_object_t *vector;
```

```
} _licensed_obj_list_t;
```

```
typedef struct
```

```
{
```

```
    _dword_t  componentId;
```

```
    _dword_t  funcLevel;
```

```
} _component_level_t;
```

```
typedef struct
{
    _dword_t      size;
    _component_level_t *vector;
} _components_level_list_t;
```

///
Possible key invalidity reason

```
typedef enum
{
    kirUnknown    = 0,
    kirValid,
    kirExpired,
    kirCorrupted,
    kirNotSigned,
    kirWrongProduct,
    kirNotFound,
    kirBlackListed,
    kirTrialAlredyUsed,
    kirIllegalBaseUpdate
} _key_invalid_reason_t;
```

```
typedef enum
{
    ktUnknown = 0,
    ktBeta,
    ktTrial,
    ktTest,
    ktOEM,
    ktCommercial
} _key_type_t;
```

```
typedef enum
{
    flUnknown = 0,
    flNoFeatures,
    flNoUpdates,
    flFullFunctionality
} _functionality_level_t;
```

///
Checking modes

```
typedef enum
{
    cmUnknown      = 0,
    cmFullCheck     = 1,          ///< Do bases update checking using kavset.xml
                                ///< and key serial number checking against black.lst
    cmKeyOnly       = 2          ///< Check key against system date only
} _checking_modes_t;
```

```
typedef struct
{
    _dword_t      keyVer;        ///< key file structure version
```

```

_date_t      keyCreationDate;  //!< key file creation date
_key_serial_number_t  keySerNum;    //!< key file serial number
_key_type_t  keyType;    //!< key type (commercial, trial, ...)
_dword_t     keyLifeSpan;    //!< key validity period since installation (days)
_date_t      keyExpDate;    //!< key limit validity period (days)
_dword_t     licenseCount;    //!< license number (key ver 1-3)
_TCHAR*      productName;    //!< product name
_dword_t     appId;    //!< application name
_dword_t     productId;    //!< product id
_TCHAR*      productVer;    //!< major product version
_licensed_obj_list_t  licensedObjLst;  //!< licensed objects with license count
_TCHAR*      licenseInfo;    //!< descriptive license info
_TCHAR*      supportInfo;    //!< support company properties
_dword_t     marketSector;    //!< market sector
_components_level_list_t  compFuncLevel;  //!< components functionality levels list
_TCHAR*      customer_info;    //!< customer info
} _key_info_t;

```

//! Licensing key checking results

```
typedef struct
```

```

{
    _TCHAR*      keyFileName;    //!< Key file name (without path)
    _key_info_t*  keyInfo;    //!< Info from key file
    _key_invalid_reason_t  invalidReason;    //!< Key invalidity reason (or kirValid)
    _date_t      keyExpirationDate;  //!< Key expiration date
    _date_t      appLicenseExpDate;  //!< App license expiration date
    _dword_t     daysTillExpiration;  //!< Days left till app expiration day
    _functionality_level_t  funcLevel;    //!< Application functionality level provided by this key
    unsigned int  componentFuncBitMask;  //!< Component functionality level bit mask
} _check_info_t;

```

```
typedef struct
```

```

{
    unsigned int  size;
    _key_info_t  *vector;
} _key_info_list_t;

```

```
typedef struct
```

```

{
    unsigned int  size;
    _check_info_t  *vector;
} _check_info_list_t;

```

```
typedef struct
```

```

{
    _key_serial_number_t  keySerialNumber;
    _key_invalid_reason_t  invalidReason;
    _date_t      keyExpDate;
    int          isActiveKey;
} _light_check_info_t;

```

```
typedef struct
```

```
{
    unsigned int    size;
    _light_check_info_t *vector;
} _light_check_info_list_t;
```

```
typedef struct
{
    unsigned int    size;
    char            *image;
} _key_file_image_t;
```

```
typedef struct
{
    unsigned int    size;
    _key_file_image_t *vector;
} _key_file_img_list_t;
```

```
//-----
```

```
HRESULT    init        (ILicenseStorage *licenseStorage,
                        _app_info_t      *appInfo,
                        const _TCHAR     *basePath,
                        _checking_modes_t  checkMode
                        );
```

```
// ----- Licensing checking routines -----
```

```
///! Checks application licensing mode against active key
HRESULT    checkActiveKey  (_check_info_t *pCheckInfo);
///! Checks key file and installes it if the key is new
HRESULT    checkKeyAndInstall (const _TCHAR *fileName,
                                char        *fileBody,
                                unsigned int  fileSize,
                                _check_info_t *checkInfo
                                );
///! Checks key presented as file image
HRESULT    checkKeyInMemory (const _TCHAR *fileName,
                                char        *fileBody,
                                unsigned int  fileSize,
                                _check_info_t *checkInfo
                                );
///! Returns licensing checking info for pointed key file
HRESULT    checkKeyFile    (const _TCHAR *keyFileName,
                            _app_info_t  *appInfo,
                            _check_info_t *checkInfo
                            );
///! Returns licensing checking info for key files found in pointed location
HRESULT    checkKeyFiles   (const _TCHAR *dir,
                            _check_info_list_t *checkList
                            );
```

```

    //! Checks if active key serial number found in provided black list
    HRESULT    checkBlacklist    (const _TCHAR *blackList,
                                size_t    size
                                );

    //#if defined (__MWERKS__)
    HRESULT    syncKeys    (const _key_file_img_list_t *keyFilesList,
                            _light_check_info_list_t *keysCheckInfo,
                            _date_t    *appLicExpDate,
                            _functionality_level_t    *funcLevel
                            );
    //#endif

    // ----- Licensing managment routines -----

    //! New key registration
    HRESULT    addKey    (const _TCHAR    *fileName,
                        _TCHAR    *fileBody,
                        size_t    fileSize
                        );
    //! Active key deletion
    HRESULT    revokeActiveKey    ();
    //! Registered key deletion using key serial number
    HRESULT    revokeKey    (const _key_serial_number_t *serNum);

    // ----- Licensing managment helper routines -----

    //! Returns active licensing key info
    HRESULT    getActiveKeyInfo    (_key_info_t *keyInfo);

    //! Returns installed keys info alone with licensing checking info
    HRESULT    getInstalledKeysInfo (_check_info_list_t *checkInfo);

    //! Returns licensing key info
    HRESULT    getKeyFileInfo    (const _TCHAR    *fileName,
                                _key_info_t    *keyInfo
                                );

    //! Returns info from license keys found in pointed dir
    HRESULT    getKeyFilesInfo    (const _TCHAR    *dir,
                                _key_info_list_t    *keysList
                                );

    HRESULT    getKeyInfo    (const char    *fileBody,
                            size_t    fileSize,
                            _key_info_t    *keyInfo
                            );

```

```

    //! Frees memory allocated for key_info_t structure members
    HRESULT    freeKeyInfo    (_key_info_t *keyInfo);

    //! Frees memory allocated for key_info_list_t structure members
    HRESULT    freeKeysInfo    (_key_info_list_t *keysInfo) ;

    //! Frees memory allocated for check_info_t structure members
    HRESULT    freeCheckInfo    (_check_info_t *checkInfo);

    //! Frees memory allocated for check_info_list_t structure members
    HRESULT    freeCheckInfoList (_check_info_list_t *checkInfo);

    //! Frees memory allocated for _light_check_info_list_t structure members
    HRESULT    freeLightCheckInfoList
                (_light_check_info_list_t *checkInfoList);

    HRESULT    checkFileSign    (const _TCHAR *fileName);

//-----

#ifdef __cplusplus
}; // extern "C"
#endif

#ifdef _MSC_VER
#pragma warning (pop)
#endif

#endif

```