

# **ФИНАНСОВАЯ АВТОМАТИЗИРОВАННАЯ СИСТЕМА ТРАНСПОРТА ИНФОРМАЦИИ «FASTI 2»**

**Руководство разработчика (SDK)**

**Руководство программиста**

398-980340000290.Fasti 33.01.25031441  
Листов 99

# Содержание

<b>1 Аннотация.....</b>	<b>4</b>
<b>2 Общие сведения.....</b>	<b>5</b>
2.1 Назначение.....	5
2.2 Общий порядок работы.....	6
<b>3 Использование библиотеки libvista.....</b>	<b>8</b>
3.1 Функции библиотеки libvista.....	8
3.1.1 Функции работы с сообщениями (архивами).....	8
3.1.2 Функции работы с сервером транспортного узла.....	29
3.1.3 Функции работы с уведомлениями транспортного узла.....	44
3.1.4 Функции работы с сертификатами транспортного узла.....	47
3.1.5 Функции смены сертификата ключа пользователя.....	50
3.2 Особенности работы с памятью.....	53
3.3 Работа с отладочными библиотеками.....	53
3.4 Примеры использования библиотеки libvista.....	53
<b>4 Использование функций CryptoAPI.....</b>	<b>57</b>
4.1 Примеры использования функций CryptoAPI.....	57
<b>5 Использование библиотеки cptumar.dll.....</b>	<b>59</b>
5.1 Примеры использования библиотеки cptumar.dll.....	59
<b>6 Выполнение примеров.....</b>	<b>60</b>
6.1 Требования к программному и аппаратному обеспечению.....	60
6.2 Предварительные работы.....	60
6.3 Компиляция SDK.....	60
6.3.1 Настройка параметров.....	61
6.3.2 Windows. MS Visual Studio.....	62
6.3.3 Linux. CLion.....	65
6.3.4 Linux. CMake.....	67
6.4 Порядок использования примеров.....	68
6.4.1 Создание зашифрованного и подписанного ЭЦП сообщения.....	69
6.4.2 Создание зашифрованного и подписанного ЭЦП сообщения через оперативную память.....	70
6.4.3 Добавление ЭЦП в сообщение.....	71
6.4.4 Добавление и проверка ЭЦП в сообщении.....	72
6.4.5 Получение локального или удаленного соединения.....	72
6.4.6 Проверка ограничения соединений к серверу с одного IP-адреса.....	73
6.4.7 Проверка работоспособности основных функций API.....	74
6.4.8 Получение сообщения об ошибке.....	75
6.4.9 Отправка сообщения на сервер транспортного узла.....	76
6.4.10 Получение сообщения с сервера транспортного узла.....	77
6.4.11 Работа с сообщением (carch_work_msg).....	78
6.4.12 Работа с сообщением (work_msg_mem).....	79
6.4.13 Извлечение информации из сообщения.....	80
6.4.14 Формирование запроса на смену сертификата.....	80
6.4.15 Получение параметров PKCS#7-запроса, используя функции CryptoAPI.....	81

6.4.16	Получение параметров PKCS#7-ответа, используя функции CryptoAPI.....	82
6.4.17	Установка сертификата в ключевой контейнер.....	82
6.4.18	Удаление ключей из ключевого контейнера, используя функции CryptoAPI.....	83
6.4.19	Удаление ключей из ключевого контейнера с помощью библиотеки.....	84
6.4.20	Формирование запроса на выпуск сертификата.....	84
6.4.21	Получение параметров PKCS#7-ответа, используя библиотеку libvista.....	85
6.4.22	Установка сертификата в ключевой контейнер пользователя.....	85

<b>Приложение 1</b>	<b>Коды ошибок.....</b>	<b>86</b>
---------------------	-------------------------	-----------

# 1 Аннотация

---

В настоящем документе приводится описание программного интерфейса FASTi2 (далее – SDK).

## 2 Общие сведения

### 2.1 Назначение

FASTi2 SDK - комплект разработчика для создания пользовательских приложений, встраивания в пользовательские приложения и реализации выполнения специфических задач, для которых недостаточно стандартного программного кода поставляемого в пакете FASTi2 компании ТОО "НИЛ "Гамма Технологии".

FASTi2 предназначена для обмена информацией с гарантированной доставкой между клиентами системы. Защита передаваемой информации обеспечивается надежными средствами шифрования, выполненными согласно стандартам, принятым в Республике Казахстан.

Основные возможности системы:

- формирование сообщений пользователей в формате CMS (зашифрованных и подписанных) в соответствии с международными требованиями RFC2630;
- инвариантность к передаваемой информации;
- обеспечение конфиденциальности и целостности сообщений при передаче от отправителя до получателя;
- возможность вкладывать в сообщение неограниченное количество файлов с сохранением их атрибутов и структуры каталогов;
- сжатие вложенных в сообщение файлов;
- возможность выбора неограниченного количества получателей сообщения;
- формирование оптимальных маршрутов доставки сообщений;
- формирование подтверждений при каждом приеме сообщений пользователей узлами;
- интеграция с инфраструктурой открытых ключей (PKI);
- взаимодействие с криптографическими модулями, удовлетворяющими требованиям к интерфейсу Microsoft – CSP (Cryptographic Service Provider);
- локальное и удаленное управление и мониторинг транспортных узлов (клиентов) системы;
- выдача протоколов работы узлов системы и пользователей, биллинговых справок и информации о состоянии транспортных узлов системы.

Поддерживаемые операционные системы:

- Microsoft Windows 11;
- Windows Server 2022 Standard (21H2);
- Red Hat Enterprise Linux 9.0–9.5.

Описание:

- Набор функций:
  - работа с сообщениями (архивами);
  - работа с сервером транспортного узла;
  - работа с уведомлениями транспортного узла;
  - работа с сертификатами транспортного узла;
  - смена сертификата ключа пользователя.
- Набор примеров использования функций библиотеки libvista (для ОС Windows, Linux x32/x64).
- Набор примеров использования функций CryptoAPI (для ОС Windows).
- Набор примеров использования функций библиотеки cptumar.dll (для ОС Windows).

Состав:

- библиотеки (\*.dll / \*.so / \*\_r.so / \*.sl / \*\_r.sl) – исполняемый код;
- статические библиотеки (.lib) для неявной компоновки пользовательского кода с динамическими библиотеками;
- заголовочные файлы (\*.h) – описание констант и прототипов функций;
- примеры использования на языке Си (\*.cpp).

## 2.2 Общий порядок работы

Стандартный алгоритм обмена сообщениями включает в себя:

- Создание сообщения;
- Отправка сообщения;
- Прием сообщения;
- Просмотр сообщения;
- Сохранение вложений.

### Создание сообщения

- Сообщение создается как обычный файл в формате .cms. Здесь создается временный файл, содержащий заголовок сообщения и вложенный комментарий к сообщению, если он существует, в нешифрованном виде. **CArchCreate (..)**.



**Примечание:** Перед вызовом функции **CArchCreate (..)** рекомендуется обнулить дескриптор архива.

- Устанавливается адрес хранилища сертификатов. Здесь проверяются параметры работы с хранилищем сертификатов. **CArchSetStore (..)**.
- Инициализируется крипто-модуль. Здесь проверяется соответствие ключей и имени отправителя. Если имя отправителя не соответствует имени владельца ключевого контейнера, то сообщение не будет сформировано. **CArchInitCrypt (..)**. Если условия проверки выполнены, в дальнейшем вся информация, добавляемая в сообщение (файл .cms) будет зашифрована автоматически в соответствии со стандартами FASTi2.
- Проверяются по формату DN и регистрируется имя получателя/получателей. **CArchSetRecipients (..)**.
- Подготавливается список вложений – файлов и/или каталогов. **CArchListDir (..)**.
- Вложения заносятся (добавляются) в сообщение. **CArchAddListFile (..)**.
- Рассчитывается и добавляется в сообщение референс. **CArchGetReferens (..)**.
- Сообщение сформировано. Дальнейшие действия определяются разработчиком. Сообщение может быть немедленно отправлено или сохранено для последующей отправки.

### Отправка сообщения

- Установить соединение с локальным каталогом пользователя, в котором расположены сообщения в формате .cms, подготовленные к отправке. **vcOpenLocal (..)**.
- Установить соединение с вышестоящим узлом. Способ соединения и протокол обмена определяется клиентским приложением в соответствии с существующими условиями и настройками. Имя вышестоящего узла определяется функцией от имени пользователя. **vcOpenRemote (..)**.
- Получить список сообщений в локальном каталоге пользователя. Список содержит референс-названия сообщений. **vcList (..)**.
- Отправка сообщения функционально состоит в физическом перемещении файла-сообщения из локального каталога в домашний каталог пользователя на вышестоящем узле, формировании подтверждения и его отправки. По завершении процесса передачи файл-сообщение удаляется из домашнего каталога. **vcCopyMessage (..)**.
- По завершении отправки всех или выбранных сообщений необходимо разорвать соединение с локальным каталогом и домашним каталогом на вышестоящем узле. **vcClose (..)**.

### Прием сообщения

- Установить соединение с вышестоящим узлом. Способ соединения и протокол обмена определяется клиентским приложением в соответствии с существующими условиями и настройками. Имя вышестоящего узла определяется функцией от имени пользователя. Если соединение нормально установлено клиентское приложение получает номер обработчика (Дескриптор) соединения через который будут выполняться все последующие действия. **vcOpenRemote (..)**.
- Установить соединение с локальным каталогом пользователя, куда будут помещены сообщения с вышестоящего узла. **vcOpenLocal (..)**.
- Получить список сообщений в домашнем каталоге пользователя. Домашний каталог, в отличие от локального, расположен на вышестоящем узле и доступен только конкретному пользователю. Список содержит референс-названия сообщений, размер каждого сообщения и время отправки этого

сообщения. **vcList (..)**. Дополнительную информацию о сообщении можно получить при помощи функции **vcGetInfoMessage ()**, при этом сообщение не скачивается полностью, а передаются только его заголовок и криптографические блоки. Информацию в криптографических блоках можно получить функциями **vcGetEnvpInfo ()** и **vcGetSignInfo ()**.

- Прием сообщения функционально состоит в физическом перемещении файла-сообщения из домашнего каталога пользователя на вышестоящем узле в локальный каталог, проверки его подлинности, формировании подтверждения и его отправки. По завершении процесса передачи файл-сообщение удаляется из домашнего каталога. **vcCopyMessage (..)**.
- По завершении приема всех или выбранных сообщений необходимо разорвать соединение с вышестоящим узлом. **vcClose (..)**.

### Просмотр полученного сообщения

- Открыть сообщение находящееся в локальном каталоге. В качестве параметра в функцию открытия сообщения передается полный маршрут к файлу с сообщением. **CArchOpen (..)**.
- Инициализируется крипто-модуль. Здесь проверяется соответствие ключей и имени отправителя. Если имя отправителя не соответствует имени владельца ключевого контейнера, то сообщение не будет прочитано. **CArchInitCrypt(..)**.
- Дальнейшие действия по анализу сообщения могут производиться в любом порядке, либо не производиться вовсе, в зависимости от потребностей пользователя:
  - получить комментарий к сообщению. **CArchGetComment (..)**.
  - получить референс сообщения. **CArchGetReferens (..)**.
  - получить список вложений в сообщение. **CArchListArchFile (..)**.
  - получить список подписавших сообщение. **CArchGetSignInfo (..)**.
  - получить DN имена отправителя и получателя. **CArchGetEnvelopInfo (..)**.
- По завершении просмотра необходимо закрыть сообщение. **CArchClose (..)**.

### Сохранение вложений

- Открыть сообщение, находящееся в локальном каталоге. В качестве параметра в функцию открытия сообщения передается полный маршрут к файлу с сообщением. **CArchOpen (..)**.
- Инициализируется крипто-модуль. Здесь проверяется соответствие ключей и имени отправителя. Если имя отправителя не соответствует имени владельца ключевого контейнера, то сообщение не будет прочитано. **CArchInitCrypt (..)**.
- Получить список вложений в сообщение. **CArchListArchFile (..)**.
- Сохранить каждое вложение, отдельно указав в параметрах путь к каталогу сохранения. **CArchExtrListFile (..)**.
- По завершении процесса сохранения вложений необходимо закрыть сообщение. **CArchClose (..)**.

## 3 Использование библиотеки libvista

Таблица 2.

Описание	Имя файла <sup>1</sup>
Заголовочный файл	carch.h
Заголовочный файл	vcclient.h
Библиотека для ОС Windows	libvista.dll
Библиотека для ОС Linux	libvista-core.so.x.x.x.x
Потоковая библиотека для ОС Linux	libvista-core_r.so.x.x.x.x

### 3.1 Функции библиотеки libvista

#### 3.1.1 Функции работы с сообщениями (архивами)

Функции работы с сообщениями (архивами) предназначены для формирования защищенного сообщения в формате CMS, его проверки и извлечения вложенных данных.

Таблица 3.

Функции	Описание
int WINAPI CArchCreate (HANDLE *hArch, int type, char *fname, unsigned char *comment, int csize)	Функция <a href="#">CArchCreate</a> создает архив с указанным названием и атрибутами. На основе этого архива будет создано CMS - сообщение.
int WINAPI CArchSetRecipients (HANDLE hArch, char **rcDN, int count)	Функция <a href="#">CArchSetRecipients</a> устанавливает список получателей создаваемого сообщения (только при создании зашифрованного и/или подписанного архива).
int WINAPI CArchListDir (HANDLE hArch, char *path, bool sub, ArchListItem **list, int *count)	Функция <a href="#">CArchListDir</a> читает список файлов/каталогов в указанном каталоге. Функция используется при добавлении файлов в создаваемый архив.
int WINAPI CArchAddListFile (HANDLE hArch, ArchListItem *item)	Функция <a href="#">CArchAddListFile</a> добавляет файл/каталог в создаваемый архив. Список файлов к добавлению в архив содержится в массиве структур возвращаемой функцией CArchListDir.
int WINAPI CArchOpen (HANDLE *hArch, int *type, char *fname)	Функция <a href="#">CArchOpen</a> открывает существующий архив.
int WINAPI CArchGetMainInfo (HANDLE hArch, ArchMainInfo *info)	Функция <a href="#">CArchGetMainInfo</a> получает общую информацию об открытом архиве.
int WINAPI CArchGetComment (HANDLE hArch, unsigned char **comment, int *size)	Функция <a href="#">CArchGetComment</a> получает комментарий к архиву из открытого архива.
int WINAPI CArchGetEnvelopInfo (HANDLE hArch, char **snDN, char ***rcDN, int *count)	Функция <a href="#">CArchGetEnvelopInfo</a> получает информацию из открытого архива об отправителе и получателях.

<sup>1</sup> где x.x.x.x - версия сборки



Функции	Описание
int WINAPI CArchListArchFile (HANDLE hArch, ArchListItem **list, int *count)	Функция <a href="#">CArchListArchFile</a> получает список файлов/каталогов в открытом архиве.
int WINAPI CArchExtrListFile (HANDLE hArch, ArchListItem *item, char *dest_path, bool flag_path)	Функция <a href="#">CArchExtrListFile</a> извлекает файл или каталог, описание которого находится в структуре item, из открытого архива в указанный каталог.
int WINAPI CArchGetSignInfo (HANDLE hArch, SignData **slist, int *count)	Функция <a href="#">CArchGetSignInfo</a> получает информацию об электронных цифровых подписях из открытого архива.
int WINAPI CArchInitCrypt (HANDLE hArch, char *myContainer, char *myDN)	Функция <a href="#">CArchInitCrypt</a> позволяет установить параметры работы с крипто-модулем CSP.
int WINAPI CArchGetReferens (HANDLE hArch, char *ref)	Функция <a href="#">CArchGetReferens</a> позволяет сформировать уникальный референс для открытого/созданного архива.
int WINAPI CArchAddSign (HANDLE hArch)	Функция <a href="#">CArchAddSign</a> добавляет свою подпись к открытому архиву.
int WINAPI CArchClose (HANDLE hArch)	Функция <a href="#">CArchClose</a> закрывает открытый или созданный архив.
int WINAPI CArchRegCallbacker (HANDLE hArch, HANDLE handle, PROGRESSCALL func)	Функция <a href="#">CArchRegCallbacker</a> регистрирует пользовательскую CallBack-функцию для визуализации прогресса при работе с архивом.
ulong32 WINAPI CArchLastError (HANDLE hArch)	Функция <a href="#">CArchLastError</a> уточняет информацию об ошибке.
int WINAPI CArchSetStore (HANDLE hArch, char *host, int port, int validTime)	Функция <a href="#">CArchSetStore</a> позволяет установить параметры работы с локальным/сетевым хранилищем сертификатов.
int WINAPI CArchAddMemFile (HANDLE hArch, ArchListItem *item, unsigned char *inbuf)	Функция <a href="#">CArchAddMemFile</a> позволяет добавлять файл в архив через оперативную память.
int WINAPI CArchExtrMemFile (HANDLE hArch, ArchListItem *item, unsigned char *obuf)	Функция <a href="#">CArchExtrMemFile</a> позволяет извлекать файл из архива в оперативную память.
size_t CALLSPEC CArchGetErrorString (int err_code, int ext_err_code, char* err_string, size_t err_string_size, int lang)	Функция <a href="#">CArchGetErrorString</a> возвращает содержательное описание ошибки в буфере, адрес которого передал пользователь.
int WINAPI CArchSetPriority (HANDLE hArch, int priority)	Функция <a href="#">CArchSetPriority</a> позволяет установить приоритет сообщения в момент создания защищенного сообщения.
int WINAPI CArchSetParam (HANDLE hArch, ulong32 dwParam, unsigned char* pbData, ulong32 cbData, ulong32 dwFlags)	Функция <a href="#">CArchSetParam</a> предназначена для установки дополнительных параметров.

### 3.1.1.1 CArchAddListFile

#### Назначение

Функция **CArchAddListFile** позволяет прочитать список файлов/каталогов в указанном каталоге. Функция используется при добавлении файлов в создаваемый архив.

```
int WINAPI CArchAddListFile (HANDLE hArch, ArchListItem * item);
```

## Параметры

### **hArch** [in]

Дескриптор создаваемого архива.

### **item** [in]

Структура описывающая файл добавляемый в архив. См. описание структуры [ArchListItem](#).

## Возвращает

При успешном завершении функция возвращает ARCH\_SUCCESS, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
ERR_ARCH_HANDLE	Указан ошибочный дескриптор создаваемого архива
ERR_ARCH_PARAM	Ошибка в параметрах функции
ERR_ARCH_ADDFILE	Ошибка добавления файла/каталога

## Файл описания

Прототип описан в файле [carch.h](#).

### 3.1.1.2 CArchAddSign

#### Назначение

Функция **CArchAddSign** позволяет добавить свою подпись к открытому архиву.

```
int WINAPI CArchAddSign (HANDLE hArch);
```

## Параметры

### **hArch**

Дескриптор открытого архива.

## Возвращает

При успешном завершении функция возвращает ARCH\_SUCCESS, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
ERR_ARCH_HANDLE	Указан ошибочный дескриптор открытого архива
ERR_ARCH_TYPE	Тип данного архива не поддерживает операцию
ERR_ARCH_HASH	Ошибка вычисления/сравнения значений хэш
ERR_ARCH_CMS	Ошибка работы с форматом CMS

## Файл описания

Прототип описан в файле [carch.h](#).

### 3.1.1.3 CArchClose

#### Назначение

Функция **CArchClose** позволяет закрыть открытый или созданный архив.

```
int WINAPI CArchClose (HANDLE hArch);
```

### Параметры

**hArch** [in]

Дескриптор открытого/созданного архива.

### Возвращает

При успешном завершении функция возвращает ARCH\_SUCCESS, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
ERR_ARCH_HANDLE	Указан ошибочный дескриптор открытого архива
ERR_ARCH_CREATE	Ошибка записи в файл архива
TCMSSIMPLE_ERROR_CMS_DN_SENDER_NOT_SIGNER	Имя отправителя не соответствует имени подписывающего

### Файл описания

Прототип описан в файле [carch.h](#).

### 3.1.1.4 CArchCreate

#### Назначение

Функция **CArchCreate** создает архив с указанным названием и атрибутами. На основе этого архива будет создано CMS-сообщение.

```
int WINAPI CArchCreate (HANDLE * phArch, int type, char * fname, unsigned char * comment, int csize);
```

### Параметры

**phArch** [out]

Дескриптор создаваемого архива.

**type** [in]

Тип создаваемого архива. Определяется комбинацией флагов.

### Возвращает

При успешном завершении функция возвращает ARCH\_SUCCESS, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
ERR_ARCH_HANDLE	Указан ошибочный дескриптор открытого архива
ERR_ARCH_CREATE	Ошибка записи в файл архива
TCMSSIMPLE_ERROR_CMS_DN_SENDER_NOT_SIGNER	Имя отправителя не соответствует имени подписывающего

### Файл описания

Прототип описан в файле [carch.h](#).

### 3.1.1.5 CArchExtrListFile

#### Назначение

Функция **CArchExtrListFile** извлекает файл или каталог, описание которого находится в структуре `item`, из открытого архива в указанный каталог.

```
int WINAPI CArchExtrListFile (HANDLE hArch, ArchListItem * item, char * dest_path, bool flag_path);
```

#### Параметры

**phArch** [in]

Дескриптор открытого архива.

**item** [in]

Структура описывающая файл/каталог, извлекаемый из архива. См. описание структуры [ArchListItem](#).

**dest\_path** [in]

Полный маршрут к каталогу куда необходимо поместить файл или каталог.

**flag\_path** [in]

Указывает использовать или нет относительный путь к файлу/каталогу.

#### Возвращает

При успешном завершении функция возвращает `ARCH_SUCCESS`, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
<code>ERR_ARCH_HANDLE</code>	Указан ошибочный дескриптор открытого архива
<code>ERR_ARCH_PARAM</code>	Ошибка в параметрах функции
<code>ERR_ARCH_LSTORE</code>	Ошибка работы с хранилищем сертификатов
<code>ERR_ARCH_CMS</code>	Ошибка работы с форматом CMS
<code>ERR_ARCH_EXTRFILE</code>	Ошибка извлечения файла/каталога из архива
<code>ERR_ARCH_TYPE</code>	Тип данного архива не поддерживает операцию

#### Файл описания

Прототип описан в файле [carch.h](#).

### 3.1.1.6 CArchGetComment

#### Назначение

Функция **CArchGetComment** позволяет получить комментарий к архиву из открытого архива.

```
int WINAPI CArchGetComment (HANDLE hArch, unsigned char ** comment, int * size);
```

#### Параметры

**hArch** [in]

Дескриптор открытого архива.

**comment** [out]

Адрес буфера с комментарием к архиву.

**size [out]**

Размер буфера с комментарием.

**Возвращает**

При успешном завершении функция возвращает ARCH\_SUCCESS, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
ERR_ARCH_HANDLE	Указан ошибочный дескриптор открытого архива
ERR_ARCH_PARAM	Ошибка в параметрах функции
ERR_ARCH_OPENFILE	Ошибка открытия архива

**Файл описания**

Прототип описан в файле [carch.h](#).

**3.1.1.7 CArchGetEnvelopInfo****Назначение**

Функция **CArchGetEnvelopInfo** позволяет получить информацию из открытого архива об отправителе и получателях.

```
int WINAPI CArchGetEnvelopInfo (HANDLE hArch, char ** snDN, char *** rcDN, int * size);
```

**Параметры****hArch [in]**

Дескриптор открытого архива.

**snDN [out]**

Адрес массива с именем отправителя в формате X.500 (Distinguished Name).

**rcDN [out]**

Адрес массива строк с именами получателей в формате X.500 (Distinguished Name).

**size [out]**

Количество получателей.

**Возвращает**

При успешном завершении функция возвращает ARCH\_SUCCESS, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
ERR_ARCH_HANDLE	Указан ошибочный дескриптор открытого архива
ERR_ARCH_PARAM	Ошибка в параметрах функции
ERR_ARCH_TYPE	Тип данного архива не поддерживает операцию
ERR_ARCH_CMS	Ошибка работы с форматом CMS

**Файл описания**

Прототип описан в файле [carch.h](#).

### 3.1.1.8 CArchGetMainInfo

#### Назначение

Функция **CArchGetMainInfo** позволяет получить общую информацию об открытом архиве.

```
int WINAPI CArchGetMainInfo (HANDLE hArch, ArchMainInfo * info);
```

#### Параметры

**hArch** [in]

Дескриптор открытого архива.

**info** [out]

Адрес структуры с общей информацией. См. описание структуры [ArchMainInfo](#).

#### Возвращает

При успешном завершении функция возвращает ARCH\_SUCCESS, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
ERR_ARCH_HANDLE	Указан ошибочный дескриптор открытого архива
ERR_ARCH_PARAM	Ошибка в параметрах функции

#### Файл описания

Прототип описан в файле [carch.h](#).

### 3.1.1.9 CArchGetReferens

#### Назначение

Функция **CArchGetReferens** позволяет сформировать уникальный референс для открытого/созданного архива.

```
int WINAPI CArchGetReferens (HANDLE hArch, char * ref);
```

#### Параметры

**hArch** [in]

Дескриптор создаваемого/открытого архива.

**ref** [out]

Массив для приема строки (уникальный 32-символьный референс).

#### Возвращает

При успешном завершении функция возвращает ARCH\_SUCCESS, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
ERR_ARCH_HANDLE	Указан ошибочный дескриптор открытого архива
ERR_ARCH_PARAM	Ошибка в параметрах функции
ERR_ARCH_FORMAT	Ошибка формата/структуры архива

## Файл описания

Прототип описан в файле [carch.h](#).

### 3.1.1.10 CArchGetSignInfo

#### Назначение

Функция **CArchGetSignInfo** позволяет получить информацию об электронных цифровых подписях из открытого архива.

```
int WINAPI CArchGetSignInfo (HANDLE hArch, SignData ** slist, int * count);
```

#### Параметры

**hArch** [in]

Дескриптор открытого архива.

**slist** [out]

Указатель на массив структур содержащих информацию о подписях. См. описание структуры [SignData](#).

**count** [out]

Количество подписей.

#### Возвращает

При успешном завершении функция возвращает ARCH\_SUCCESS, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
ERR_ARCH_HANDLE	Указан ошибочный дескриптор открытого архива
ERR_ARCH_PARAM	Ошибка в параметрах функции
ERR_ARCH_TYPE	Тип данного архива не поддерживает операцию
ERR_ARCH_HASH	Ошибка вычисления/сравнения значений хэш
ERR_ARCH_CMS	Ошибка работы с форматом CMS

## Файл описания

Прототип описан в файле [carch.h](#).

### 3.1.1.11 CArchInitCrypt

#### Назначение

Функция **CArchInitCrypt** позволяет установить параметры работы с крипто-модулем CSP.

```
int WINAPI CArchInitCrypt (HANDLE hArch, char * myContainer, char * myDN);
```

#### Параметры

**hArch** [in]

Дескриптор создаваемого/открытого архива.

**myContainer** [in]

Имя профайла или имя контейнера крипто-модуля CSP.

**myDN** [in]

Собственное имя в формате X.500 (Distinguished Name).

**Возвращает**

При успешном завершении функция возвращает ARCH\_SUCCESS, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
ERR_ARCH_HANDLE	Указан ошибочный дескриптор открытого архива
ERR_ARCH_PARAM	Ошибка в параметрах функции
ERR_ARCH_TYPE	Тип данного архива не поддерживает операцию
ERR_ARCH_CSP	Ошибка работы с криптопровайдером

**Файл описания**

Прототип описан в файле [carch.h](#).

**3.1.1.12 CArchLastError****Назначение**

Функция **CArchLastError** позволяет получить уточненные (расширенные) коды ошибок функций.

ULONG32 WINAPI **CArchLastError** (HANDLE [hArch](#));

**Параметры**

**[hArch](#)** [in]

Дескриптор открытого/создаваемого архива.

**Возвращает**

Уточненные (расширенные) коды ошибок функций

**Файл описания**

Прототип описан в файле [carch.h](#).

**3.1.1.13 CArchListArchFile****Назначение**

Функция **CArchListArchFile** позволяет получить список файлов/каталогов в открытом архиве.

int WINAPI **CArchListArchFile** (HANDLE [hArch](#), ArchListItem [list](#), int \* [count](#));

**Параметры**

**[hArch](#)** [in]

Дескриптор открытого архива.

**[list](#)** [out]

Указатель на массив структур. См. описание структуры [ArchListItem](#).

**[count](#)** [out]

Количество элементов в массиве структур.

**Возвращает**

При успешном завершении функция возвращает ARCH\_SUCCESS, в противном случае возвращается код ошибки. Возможные коды ошибок функции:



Возвращаемые значения	Описание
ERR_ARCH_HANDLE	Указан ошибочный дескриптор открытого архива
ERR_ARCH_PARAM	Ошибка в параметрах функции
ERR_ARCH_LISTFILE	Ошибка чтения содержания архива

#### Файл описания

Прототип описан в файле [carch.h](#).

#### 3.1.1.14 CArchListDir

##### Назначение

Функция **CArchListDir** позволяет прочитывать список файлов/каталогов в указанном каталоге. Функция используется при добавлении файлов в создаваемый архив.

```
int WINAPI CArchListDir (HANDLE hArch, char * path, bool sub, ArchListItem ** list, int * count);
```

##### Параметры

**hArch [in]**

Дескриптор создаваемого архива.

**path [in]**

Полный путь к каталогу.

**sub [in]**

Если true, то выполнить рекурсию по всем подкаталогам.

**list [out]**

Указатель на массив структур. См. описание структуры [ArchListItem](#).

**count [out]**

Количество элементов в массиве структур.

##### Возвращает

При успешном завершении функция возвращает ARCH\_SUCCESS, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
ERR_ARCH_HANDLE	Указан ошибочный дескриптор создаваемого архива
ERR_ARCH_PARAM	Ошибка в параметрах функции
ERR_ARCH_LIST	Ошибка чтения каталога

#### Файл описания

Прототип описан в файле [carch.h](#).

#### 3.1.1.15 CArchOpen

##### Назначение

Функция **CArchOpen** открывает существующий архив.

```
int WINAPI CArchOpen (HANDLE * phArch, int * type, char * path);
```

**Параметры****phArch [out]**

Дескриптор открытого архива.

**type [out]**

Тип открытого архива. Определяется комбинацией флагов.

Значение type	Описание
TYPE_ARCH_ENCRYPT	Зашифрованный архив
TYPE_ARCH_SIGNED	Подписанный архив
TYPE_ARCH_REQUEST	Запрос в службу запросов узла
TYPE_ARCH_RESPONSE	Ответ от службы запросов узла
TYPE_ARCH_IS_CONFIRMED	Подтверждение от сервера, с вложенным подтверждением от клиента

**sub [in]**

Если true, то выполнить рекурсию по всем подкаталогам.

**list [out]**

Указатель на массив структур. См. описание структуры [ArchListItem](#).

**count [out]**

Количество элементов в массиве структур.

**Возвращает**

При успешном завершении функция возвращает ARCH\_SUCCESS, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
ERR_ARCH_HANDLE	Указан ошибочный дескриптор создаваемого архива
ERR_ARCH_PARAM	Ошибка в параметрах функции
ERR_ARCH_LIST	Ошибка чтения каталога

**Файл описания**

Прототип описан в файле [carch.h](#).

**3.1.1.16 CArchRegCallBacker****Назначение**

Функция **CArchRegCallBacker** позволяет зарегистрировать пользовательскую CallBack-функцию для визуализации прогресса при работе с архивом.

```
int WINAPI CArchRegCallBacker (HANDLE hArch, HANDLE handle, PROGRESSCALL func);
```

**Параметры****hArch [in]**

Дескриптор создаваемого/открытого архива.

**handle [in]**

Дескриптор пользовательской CallBack-функции.

**func [in]**

Пользовательская функция.

**Возвращает**

При успешном завершении функция возвращает ARCH\_SUCCESS, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
ERR_ARCH_HANDLE	Указан ошибочный дескриптор открытого архива
ERR_ARCH_PARAM	Ошибка в параметрах функции

**Файл описания**

Прототип описан в файле [carch.h](#).

**3.1.1.17 CArchSetRecipients****Назначение**

Функция **CArchSetRecipients** устанавливает список получателей создаваемого сообщения (только при создании зашифрованного и/или подписанного архива).

```
int WINAPI CArchSetRecipients (HANDLE hArch, char ** rcDN, int count);
```

**Параметры****hArch [in]**

Дескриптор создаваемого архива.

**rcDN [in]**

Массив строк содержащий имена получателей в формате X.500 (Distinguished Name).

**count [in]**

Количество получателей.

**Возвращает**

При успешном завершении функция возвращает ARCH\_SUCCESS, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
ERR_ARCH_HANDLE	Указан ошибочный дескриптор создаваемого архива
ERR_ARCH_PARAM	Ошибка в параметрах функции
ERR_ARCH_TYPE	Тип данного архива не поддерживает операцию
ERR_ARCH_LSTORE	Ошибка работы с хранилищем сертификатов
ERR_ARCH_EXPORTKEY	Ошибка экспорта сессионного ключа
ERR_ARCH_CMS	Ошибка работы с форматом CMS

**Файл описания**

Прототип описан в файле [carch.h](#).

### 3.1.1.18 CArchSetStore

#### Назначение

Функция **CArchSetStore** устанавливает параметры работы с локальным/сетевым хранилищем сертификатов.

```
int WINAPI CArchSetStore (HANDLE hArch, char * host, int port, int validTime);
```

#### Параметры

**hArch [in]**

Дескриптор создаваемого/открытого архива.

**host [in]**

Имя хоста локального/ сетевого хранилища сертификатов.

**port [in]**

Номер порта локального/ сетевого хранилища сертификатов.

**validTime [in]**

Время доверия и хранения сертификатов в локальном хэше узла.

#### Возвращает

При успешном завершении функция возвращает ARCH\_SUCCESS, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
ERR_ARCH_HANDLE	Указан ошибочный дескриптор открытого архива
ERR_ARCH_PARAM	Ошибка в параметрах функции

#### Файл описания

Прототип описан в файле [carch.h](#).

### 3.1.1.19 CArchAddMemFile

#### Назначение

Функция **CArchAddMemFile** добавляет файл в архив через оперативную память.

```
int WINAPI CArchAddMemFile (HANDLE hArch, ArchListItem * item, unsigned char * inbuf);
```

#### Параметры

**hArch [in]**

Дескриптор открытого архива.

**item [in]**

Структура описывающая файл/каталог, добавляемый в архив. См. описание структуры [ArchListItem](#).

**inbuf [in]**

Массив, содержащий файл для добавления.

#### Возвращает

При успешном завершении функция возвращает ARCH\_SUCCESS, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
ERR_ARCH_HANDLE	Указан ошибочный дескриптор открытого архива
ERR_ARCH_PARAM	Ошибка в параметрах функции

#### Файл описания

Прототип описан в файле [carch.h](#).

#### 3.1.1.20 CArchExtrMemFile

##### Назначение

Функция **CArchExtrMemFile** извлекает файл из архива в оперативную память.

```
int WINAPI CArchExtrMemFile (HANDLE hArch, ArchListItem * item, unsigned char * outhuf);
```

##### Параметры

**hArch** [in]

Дескриптор открытого архива.

**item** [in]

Структура описывающая файл/каталог, извлекаемый из архива. См. описание структуры [ArchListItem](#).

**outhuf** [out]

Массив для извлекаемого файла.

##### Возвращает

При успешном завершении функция возвращает ARCH\_SUCCESS, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
ERR_ARCH_HANDLE	Указан ошибочный дескриптор открытого архива
ERR_ARCH_PARAM	Ошибка в параметрах функции
ERR_ARCH_LSTORE	Ошибка работы с хранилищем сертификатов
ERR_ARCH_CMS	Ошибка работы с форматом CMS
ERR_ARCH_EXTRFILE	Ошибка извлечения файла/каталога из архива
ERR_ARCH_TYPE	Тип данного архива не поддерживает операцию

#### Файл описания

Прототип описан в файле [carch.h](#).

#### 3.1.1.21 CArchGetErrorString

##### Назначение

Функция **CArchGetErrorString** возвращает содержательное описание ошибки в буфере, адрес которого передал пользователь.

```
size_t CALLSPEC CArchGetErrorString (int err_code, int ext_err_code, char * err_string, size_t err_string_size, int lang);
```

## Параметры

**err\_code [in]**

Код ошибки.

**ext\_err\_code [in]**

Расширенный код ошибки.

**err\_string [in]**

Указатель на буфер под сообщение об ошибке.

**err\_string\_size [in]**

Размер буфера под сообщение об ошибке.

**lang [in]**

Код выбора языка сообщения об ошибке: 0 - русский, 1 - английский.

## Возвращает

При успешном завершении функция возвращает размер сообщения об ошибке. Если коду ошибки не установлено строковое значение, то будет возвращена строка: "Unknown error"

## Дополнительная информация

Буфер под сообщение об ошибке выделяет пользователь. Если в качестве указателя на буфер под сообщение об ошибке передать NULL, то функция вернет длину сообщения об ошибке.

## Файл описания

Прототип описан в файле [carch.h](#).

### 3.1.1.22 CArchSetPriority

## Назначение

Функция **CArchSetPriority** позволяет установить приоритет сообщения в момент создания защищенного сообщения.

```
int WINAPI CArchSetPriority (HANDLE hArch, int ptiority);
```

## Параметры

**hArch [in]**

Дескриптор открытого/закрытого архива.

**ptiority [in]**

Приоритет сообщения.

## Возвращает

Приоритет может учитываться при обработке/приеме/передаче сообщений при указании соответствующих настроек на узлах FASTi2. Приоритет может быть установлен в одно из следующих значений (по аналогии с приоритетом в e-mail):

- #define PRIORITY\_HIGHEST 1
- #define PRIORITY\_HIGH 2
- #define PRIORITY\_NORMAL 3
- #define PRIORITY\_LOW 4
- #define PRIORITY\_LOWEST 5

По умолчанию (если данный метод не вызывался) устанавливается приоритет PRIORITY\_NORMAL.

**Файл описания**

Прототип описан в файле [carch.h](#).

**3.1.1.23 CArchSetParam****Назначение**

Функция **CArchSetParam** предназначена для установки дополнительных параметров.

```
int WINAPI CArchSetParam (HANDLE hArch, ulong32 dwParam, unsigned char* pbData, ulong32 cbData,
ulong32 dwFlags);
```

**Установка параметров при работе с определенным дескриптором**

- #define CP\_STORE\_URL 1
- #define CP\_STORE\_PROXY 2
- #define CP\_STORE\_VALID 3
- #define CP\_CA\_BODY 4
- #define CP\_CA\_PATH 5
- #define CP\_CA\_URL 6
- #define CP\_CRL\_BODY 7
- #define CP\_CRL\_PATH 8
- #define CP\_CRL\_URL 9
- #define CP\_CMS\_USAGE 10
- #define CP\_CRT\_VERIFY 11

**dwParam =**

**CP\_STORE\_URL**, pbData – строка с URL LDAP хранилища сертификатов.

**возможные значения:**

**dwParam =**

**CP\_STORE\_PROXY**, pbData – строка с URL Proxy сервера.

**возможные значения:**

**dwParam =**

**CP\_STORE\_VALID**, pbData – строка с указателем на число, обозначающее кол-во секунд хранения

**возможные значения:** сертификатов пользователей в кэше.

**dwParam =**

**CP\_CMS\_USAGE**, CARCH\_INC\_CERT\_SIGN 1 – вкладывать сертификат в CMS Signed;

**возможные значения:**

CARCH\_INC\_CERT\_EXCH 2 – вкладывать сертификат в CMS Envelopment.

**dwParam =**

**CP\_CRT\_VERIFY**, CARCH\_CERT\_VERIFY\_SIGN 1 – проверять сертификат при проверке ЭЦП;

**возможные значения:**

CARCH\_CERT\_VERIFY\_EXCH 2 – проверять сертификат при расшифровании.

**Файл описания**

Прототип описан в файле [carch.h](#).

**3.1.1.24 CARCH.H**

Описание констант и прототипов функций работы с сообщениями (архивами)

```
#ifndef __CARCH_H
#define __CARCH_H

#include <time.h>

#ifndef VISTA_DEFS
```

```

#define VISTA_DEFS
typedef int          long32;
typedef unsigned int ulong32;
typedef long32 VTIME_T;
#ifndef __cplusplus
    typedef unsigned char bool;
#endif
#if defined(_WIN32) || defined(WIN32)
    #include <windows.h>
    #define WAPI __stdcall
    #define CALLSPEC __declspec(dllimport) WAPI
#else
    typedef void *HANDLE;
    #define WAPI
    #define CALLSPEC
    #define _export
#endif
#endif

#define TYPE_ENCODING_DER 0
#define TYPE_ENCODING_BASE64 1

/* Тип создаваемого архива */
#define TYPE_ARCH_ENCRYPT 1
#define TYPE_ARCH_SIGNED 2
#define TYPE_ARCH_REQUEST 4
#define TYPE_ARCH_RESPONSE 8
#define TYPE_ARCH_IS_CONFIRMED 32

/* Коды возврата */
#define ARCH_SUCCESS 0
#define ERR_ARCH_HANDLE 1
#define ERR_ARCH_PARAM 2
#define ERR_ARCH_TYPE 3
#define ERR_ARCH_CREATE 4
#define ERR_ARCH_EXPORTKEY 5
#define ERR_ARCH_LIST 6
#define ERR_ARCH_ADDFILE 7
#define ERR_ARCH_OPENFILE 8
#define ERR_ARCH_LISTFILE 9
#define ERR_ARCH_EXTRFILE 10
#define ERR_ARCH_HASH 11
#define ERR_ARCH_CMS 12
#define ERR_ARCH_CSP 13
#define ERR_ARCH_LSTORE 14
#define ERR_ARCH_FORMAT 15
#define ERR_ARCH_GET_CRL 16
#define ERR_ARCH_SET_CA 17
#define ERR_ARCH_SET_CRL 18

#define PRIORITY_HIGHEST 1
#define PRIORITY_HIGH 2
#define PRIORITY_NORMAL 3
#define PRIORITY_LOW 4
#define PRIORITY_LOWEST 5

#define CP_STORE_URL 1
#define CP_STORE_PROXY 2
#define CP_STORE_VALID 3
#define CP_CA_BODY 4
#define CP_CA_PATH 5
#define CP_CA_URL 6
#define CP_CRL_BODY 7
#define CP_CRL_PATH 8
#define CP_CRL_URL 9
#define CP_CMS_USAGE 10
#define CP_CRT_VERIFY 11

// CP_CMS_USAGE:
#define CARCH_INC_CERT_SIGN 1
#define CARCH_INC_CERT_EXCH 2
#define CARCH_INC_CRL 4

// CP_CRT_VERIFY:
#define CARCH_CERT_VERIFY_SIGN 1
#define CARCH_CERT_VERIFY_EXCH 2

/* Общая информация об архиве */
typedef struct {
    unsigned char HostOS;

```



```

unsigned char ArchType;
ulong32 DCount;
ulong32 FCount;
ulong32 Time;
ulong32 MemoSize;
ulong32 CMSEnvpsSize;
ulong32 CMSSignSize;
ulong32 Priority;
} ArchMainInfo;

/* Элемент списка вложений (файлов/каталогов) */
typedef struct {
    int type;
    char *name;
    ulong32 size;
    ulong32 usize;
    ulong32 mode;
    ulong32 c_time;
    ulong32 a_time;
    ulong32 reserv;
} ArchListItem;

/* Элемент списка электронных цифровых подписей */
typedef struct {
    char *DN;
    VTIME_T GTime;
    char *GTimeStr;
    char *HashOID;
    char *SignOID;
    bool Result;
} SignData;

/*Структура USER_SIGN_INFO предоставляет возможность работы с подтверждениями.*/
typedef struct {
    int Version; // Версия
    char *userDN; // Имя подписавшего в формате DN
    char *Issuer; // Имя ЦС выпустившего сертификат в формате DN
    unsigned char *SerialData; // Серийный номер сертификата
    long SerialSize; // Размер серийного номера
    char *HashAlgOID; // OID хэш-алгоритма
    unsigned char *HashData; // Значение хэш-вектора
    long HashSize; // Размер хэш-вектора
    char *SignAlgOID; // OID алгоритма электронной цифровой подписи (ЭЦП)
    unsigned char *SignData; // электронная цифровая подпись
    long SignSize; // размер электронной цифровой подписи
    char *SignTimeStr; // время постановки ЭЦП в формате Generalized Time
    (SYNTAX 1.3.6.1.4.1.1466.115.121.1.24)
    unsigned char *AttrData; // Данные атрибутов CMS
    long AttrSize; // Размер данных атрибутов CMS
    VTIME_T SignTime; // время постановки ЭЦП по Гривичу
    unsigned char *OKeyData; // Открытый ключ вложенного сертификата подписавшего
    long OKeySize; // Размер открытого ключа
} USER_SIGN_INFO, USER_SIGN;

/*Структура CMS_SIGN_INFO предоставляет возможность работы с подтверждениями.*/
typedef struct {
    int Version; // Версия CMS (3)
    char *ContentType; // OID содержимого CMS (1.2.840.113549.1.7.2)
    unsigned char *ContentData; // Вложенные в CMS данные
    long ContentSize; // Размер вложенных данных
    USER_SIGN_INFO **Signers; // Подписи пользователей
    int SignersCount; // Количество подписей
} CMS_SIGN_INFO, CMS_SIGN;

typedef struct _CARCH_SIGN_MESSAGE_PARAM {
    int cbSize;
    unsigned int dwProvType;
    const char *myContainer;
    const char *myDN;
    const char *store_url;
    int validTime;
    int EncodingType;
    unsigned int usage;
} CARCH_SIGN_MESSAGE_PARAM, *PCARCH_SIGN_MESSAGE_PARAM;

typedef void (WAPI *PROGRESSCALL)(HANDLE handle, char *fname, ulong32 FullSize, ulong32 ProcSize);

#ifdef __cplusplus
extern "C" {
#endif

```

```

//-----
// Create CMS archive functions family
//-----
int CALLSPEC CArchCreate(HANDLE *hArch, int type, char *fname, unsigned char *comment,
int csize);
int CALLSPEC CArchSetRecipients(HANDLE hArch, char **rcDN, int count);
int CALLSPEC CArchAddRecipient(HANDLE hArch, char *recipient);
int CALLSPEC CArchEndRecipient(HANDLE hArch);
int CALLSPEC CArchListDir(HANDLE hArch, char *path, bool sub, ArchListItem **list, int
*count);
int CALLSPEC CArchAddListFile(HANDLE hArch, ArchListItem *item);
int CALLSPEC CArchAddMemFile (HANDLE hArch, ArchListItem *item, unsigned char *inbuf);
int CALLSPEC CArchSetPriority(HANDLE hArch, int priority);
//-----
// Extract of archive
//-----
int CALLSPEC CArchOpen(HANDLE *hArch, int *type, char *fname);
int CALLSPEC CArchGetMainInfo(HANDLE hArch, ArchMainInfo *info);
int CALLSPEC CArchGetComment(HANDLE hArch, unsigned char **comment, int *size);
int CALLSPEC CArchGetEnvelopInfo(HANDLE hArch, char **snDN, char ***rcDN, int *count);
int CALLSPEC CArchGetEnvelopSN(HANDLE hArch, int index, unsigned char *sn, int *size);
int CALLSPEC CArchListArchFile(HANDLE hArch, ArchListItem **list, int *count);
int CALLSPEC CArchExtrListFile(HANDLE hArch, ArchListItem *item, char *dest_path, bool
flag_path);
int CALLSPEC CArchExtrMemFile(HANDLE hArch, ArchListItem *item, unsigned char *obuf);
int CALLSPEC CArchGetSignInfo(HANDLE hArch, SignData **slist, int *count);
int CALLSPEC CArchGetCert(HANDLE hArch, int type, char *dn, unsigned char *cert_data,
long *cert_size, char *subjDN, char *issuDN, time_t *notBefore, time_t *notAfter,
unsigned char *snumData, long *snumSize, char *okeyOID);
//-----
// Init CSP and CMS objects
//-----
int CALLSPEC CArchInitCrypt(HANDLE hArch, char *myContainer, char *myDN);
//-----
// Referens of archive
//-----
int CALLSPEC CArchGetReferens(HANDLE hArch, char *ref);
//-----
// Add Signer
//-----
int CALLSPEC CArchAddSign(HANDLE hArch);
//-----
// Close of archive
//-----
int CALLSPEC CArchClose(HANDLE hArch);
//-----
// Registration of service function for visualization
//-----
int CALLSPEC CArchRegCallbacker(HANDLE hArch, HANDLE handle, PROGRESSCALL func);
//-----
// Detailed info of error
//-----
ulong32 CALLSPEC CArchLastError(HANDLE hArch);
size_t CALLSPEC CArchGetErrorString( int err_code, int ext_err_code, char* err_string,
size_t err_string_size, int lang );
//-----
// Set Param
//-----
int CALLSPEC CArchSetStore(HANDLE hArch, char *host, int port, int validTime);
int CALLSPEC CArchSetParam(HANDLE hArch, ulong32 dwParam, unsigned char* pbData, ulong32
cbData, ulong32 dwFlags);
//-----

//-----
// Info of CMS
//-----
int CALLSPEC FreeCMSSignedInfo (HANDLE hCMS);
int CALLSPEC CMSSignedInfo (HANDLE* phCMS, unsigned char* cms_data, long cms_size,
CMS_SIGN_INFO** cms);
//-----

//-----
// CMS Sign
//-----
int CALLSPEC CMSMessageSign(PCARCH_SIGN_MESSAGE_PARAM pSignPara, int cToBeSigned,
unsigned char *rgpbToBeSigned[], int rgcbToBeSigned[], unsigned char **pbSignedBlob,
int *pcbSignedBlob);
int CALLSPEC CMSMessageAddSign(PCARCH_SIGN_MESSAGE_PARAM pSignPara, unsigned
char *pbSignedBlobIn, int cbSignedBlobIn, unsigned char **pbSignedBlobOut, int
*pcbSignedBlobOut);

```

```

int CALLSPEC CMSMessageVerifySign(PCARCH_SIGN_MESSAGE_PARAM pVerifyPara, int
dwSignerIndex, unsigned char *pbSignBlob, int cbSignBlob, int cToBeSigned, unsigned
char *rgpbToBeSigned[], int rgcbToBeSigned[]);
int CALLSPEC CMSMessageFreeRes(unsigned char *pbSignedBlob);
//-----

//-----
// Set CA/CRL for verify of certificates
//-----
int CALLSPEC SetVerify (char *profile, unsigned char *crl_data, long crl_size, int
check_sign, int *ext_code);
int CALLSPEC ChkVerify ( void );
int CALLSPEC GetCRL (HANDLE *phCRL, int type, char *destination, unsigned char
**data, long *size, int *ext_code);
int CALLSPEC GetCRLEx (HANDLE *phCRL, int type, char *destination, char *proxy_url,
unsigned char **data, long *size, int *ext_code);
void CALLSPEC FreeCRL (HANDLE hCRL);
void CALLSPEC GetCrtVrf (void **crt_vrf);
//-----

//-----
// Requests to RA
//-----
int CALLSPEC vGetCrtValidity (
char *pszContainer, // имя контейнера
ulong32 dwKeySpec, // тип ключа AT_KEYEXCHANGE(1)/AT_SIGNATURE(2)
time_t *notBefore, // начало действия сертификата
time_t *notAfter, // окончание действия сертификата
void *info, // зарезервировано, должно быть NULL
ulong32 *lastError); // код ошибки CSP

int CALLSPEC vGetCrtRequest (
char *pszContainer, // имя контейнера
ulong32 dwKeySpec, // тип ключа AT_KEYEXCHANGE(1)/AT_SIGNATURE(2)
char *DN, // зарезервировано, должно быть NULL
unsigned char *req, // результат
ulong32 *size, // размер
ulong32 *lastError); // код ошибки CSP

int CALLSPEC vSetCrtToCont (
char *pszContainer, // имя контейнера
unsigned char *crt, // тело сертификата
ulong32 size, // размер сертификата
ulong32 *lastError); // код ошибки CSP

int CALLSPEC vGetRequestTransID (
unsigned char *req, // запрос
ulong32 size, // размер запроса
char *id); // TransID

int CALLSPEC vGetCmsAttribute(
unsigned char *buf, // PKCS7 / CMS
ulong32 size, // размер
const char *oid, // OID атрибута
unsigned char *res, // результат
long *rsize); // размер результата

//-----

#ifdef __cplusplus
}
#endif

```

### 3.1.1.25 Структура ArchListItem

#### Назначение

Структура **ArchListItem** определяет элемент списка файлов/каталогов.

```
typedef struct {int type; char * name; ulong32 size; ulong32 usize; ulong32 mode; time_t c_time; time_t a_time;
ulong32 reserve;} ArchListItem;
```

**Элементы структуры**

<b>type</b>	Определяет тип элемента 1 –файл, 0 –каталог.
<b>name</b>	Имя файла/каталога.
<b>size</b>	Размер упакованного файла.
<b>usize</b>	Неупакованный (фактический) размер файла.
<b>mode</b>	Стандартные атрибуты файла/каталога.
<b>c_time</b>	Время создания файла/каталога.
<b>a_time</b>	Время последнего доступа к файлу/каталогу.
<b>reserv</b>	Зарезервировано.

**Файл описания**

Прототип описан в файле [carch.h](#).

**3.1.1.26 Структура ArchMainInfo****Назначение**

Структура **ArchMainInfo** определяет общую информацию об открытом/созданном архиве.

```
typedef struct {unsigned char HostOS; unsigned char ArchType; ulong32 DCount; ulong32 FCount; ulong32 Time; ulong32 MemoSize; ulong32 CMSEnvSize; ulong32 CMSSignSize; ulong32 Priority} ArchMainInfo;
```

**Элементы структуры**

<b>HostOS</b>	Определяет операционную систему где был создан данный архив.
<b>ArchType</b>	Определяет тип архива (простой, подписанный, зашифрованный, подписанный и зашифрованный).
<b>DCount</b>	Определяет количество вложенных в архив каталогов.
<b>FCount</b>	Определяет количество вложенных в архив файлов.
<b>Time</b>	Определяет время создания архива.
<b>MemoSize</b>	Определяет размер вложенного в архив комментария.
<b>CMSEnvSize</b>	Определяет размер блока отвечающего за шифрование.

**CMSSignSize**

Определяет размер блока отвечающего за ЭЦП.

**Priority**

Определяет приоритет сообщения.

**Файл описания**

Прототип описан в файле [carch.h](#).

**3.1.1.27 Структура SignData****Назначение**

Структура **SignData** определяет элемент списка электронных цифровых подписей (ЭЦП).

```
typedef struct {char * DN; time_t Gtime; char * GTimeStr; char * HashOID;char * SignOID; bool Result;} SignData;
```

**Элементы структуры****DN**

Имя подписавшего в формате X.500 (Distinguished Name).

**Gtime**

Структура времени постановки подписи по Гринвичу.

**GTimeStr**

Время постановки подписи в формате ГГГГММДДММССZ по Гринвичу.

**HashOID**

OID используемого алгоритма хэш.

**SignOID**

OID используемого алгоритма подписи.

**Result**

Результат: true – подпись верна, false - неверна.

**Файл описания**

Прототип описан в файле [carch.h](#).

**3.1.2 Функции работы с сервером транспортного узла**

Функции работы с сервером транспортного узла предназначены для гарантированного обмена сообщениями по защищенному протоколу с сервером транспортного узла FASTi2.

**Таблица 4.**

Функции	Описание
int WINAPI vcOpenLocal (HANDLE *hDst, char *location)	Функция <a href="#">vcOpenLocal</a> создает соединение с файловой системой.
int WINAPI vcOpenRemote (HANDLE *hDst, char *host, char *port, char *prof, char *mydn, int type, int ltout, int stout, char *srdn, char *store_host, int store_port, int valid_time, int proxy_type, char *proxy_host, int proxy_port, char *proxy_user, char *proxy_pass, char *proxy_exclude_list)	Функция <a href="#">vcOpenRemote</a> создает защищенное сетевое соединение с сервером транспортного узла FASTi2.

Функции	Описание
int WINAPI vcList (HANDLE hDst, VLISTITEM **list, int *count)	Функция <a href="#">vcList</a> получает список файлов и каталогов для установленного соединения.
int WINAPI vcCopyMessage (HANDLE hDst, const char *file, HANDLE hSrc, unsigned char **conf_data, long *conf_size, char *conf_mess)	Функция <a href="#">vcCopyMessage</a> копирует сообщение между любыми двумя типами соединений.
int WINAPI vcDelMessage (HANDLE hDst, const char *file)	Функция <a href="#">vcDelMessage</a> удаляет сообщение.
int WINAPI vcGetInfoMessage (HANDLE hDst, const char *file, unsigned char **envp_data, unsigned long *envp_size, unsigned char **sign_data, unsigned long *sign_size)	Функция <a href="#">vcGetInfoMessage</a> возвращает криптографическую информацию о сообщении.
int WINAPI vcClose (HANDLE hDst)	Функция <a href="#">vcClose</a> закрывает соединение.
int WINAPI vcAbort (HANDLE hDst)	Функция <a href="#">vcAbort</a> экстренно прерывает выполняемую операцию.
int WINAPI vcRegCallback (HANDLE hDst, HANDLE cont, callback function)	Функция <a href="#">vcRegCallback</a> регистрирует для соединения функцию пользователя, в которую будут передаваться данные о прогрессе приема/передачи сообщения.
int WINAPI vcGetEnvpInfo (HANDLE hDst, unsigned char *envp_data, unsigned long envp_size, time_t *time_create, char *sender, char **recipients, int *count)	Функция <a href="#">vcGetEnvpInfo</a> возвращает криптографическую информацию о блоке шифрования CMS Enveloped.
int WINAPI vcGetSignInfo (HANDLE hDst, unsigned char *sign_data, unsigned long sign_size, char **signers, time_t **signtime, int *count)	Функция <a href="#">vcGetSignInfo</a> возвращает криптографическую информацию о блоке электронной цифровой подписи CMS Signed.
int WINAPI vcGetLastError (HANDLE hDst)	Функция <a href="#">vcGetLastError</a> выдает дополнительную информацию об ошибке.
size_t CALLSPEC vcGetErrorString (int err_code, int ext_err_code, char* err_string, size_t err_string_size, int lang)	Функция <a href="#">vcGetErrorString</a> возвращает содержательное описание ошибки в буфере, адрес которого передал пользователь.
bool CALLSPEC vcIsRemote (HANDLE hDst)	Функция <a href="#">vcIsRemote</a> получает информацию о соединении: файловая система/удаленная система.

### 3.1.2.1 vcOpenLocal

#### Назначение

Функция **vcOpenLocal** создает соединение с файловой системой.

```
int WINAPI vcOpenLocal (HANDLE * hDst, char * location);
```

#### Параметры

**hDst** [out]

Указатель на дескриптор соединения.

**location** [in]

Каталог файловой системы.

## Возвращает

При успешном завершении функция возвращает `VCLIENT_SUCCESS`, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
<code>VCLIENT_ERROR_CONNECT</code>	Ошибка. Не удалось создать соединение

## Файл описания

Прототип описан в файле [vclient.h](#).

### 3.1.2.2 vcOpenRemote

#### Назначение

Функция **vcOpenRemote** создает защищенное сетевое соединение с сервером транспортного узла FASTi2.

```
int WINAPI vcOpenRemote (HANDLE * hDst, char * host, char * port, char * prof, char * myDN, int type, int ltout, int stout, char * srDN, char * store_host, int store_port, int valid_time, int proxy_type, char * proxy_host, int proxy_port, char * proxy_user, char * proxy_pass, char * proxy_exclude_list);
```

#### Параметры

##### **hDst** [out]

Указатель на дескриптор соединения.

##### **host** [in]

Имя хоста или IP-адрес сервера транспортного узла FASTi2.

##### **port** [in]

Порт сервера транспортного узла FASTi2.

##### **prof** [in]

Имя профайла криптопровайдера CSP (Windows) или имя файла ключевого контейнера криптопровайдера CSP (UNIX/LINUX).

##### **myDN** [in]

Собственное имя в формате X.500 (Distinguished Name).

##### **type** [in]

Тип подключения: 0 – прием данных, 1 – передача данных.

##### **ltout** [in]

Время ожидания для локальных сетевых операций в секундах от 1 до 360.

##### **stout** [in]

Время ожидания для серверных сетевых операций в секундах от 60 до 3600.

##### **srDN** [in]

Имя сервера в формате X.500 (Distinguished Name).

##### **store\_host** [in]

Имя хоста или IP-адрес сервера хранилища сертификатов.

##### **store\_port** [in]

Порт сервера хранилища сертификатов.

##### **valid\_time** [in]

Время хранения сертификатов в кэше.

**proxy\_type [in]**

Тип прокси сервера ( 0-(не использовать прокси), 1-(HTTP), 2-(SOCKS4), 3-(SOCKS5), 4-(TELNET) ).

**proxy\_host [in]**

Имя хоста или IP-адрес прокси сервера.

**proxy\_port [in]**

Порт прокси сервера.

**proxy\_user [in]**

Имя пользователя прокси для авторизации, если не используется [NULL] или пустая строка.

**proxy\_pass [in]**

Пароль пользователя прокси для авторизации, если не используется [NULL] или proxy\_pass[0]==0.

**proxy\_exclude\_list [in]**

Список адресов, для которых не использовать прокси, разделитель ';' (символ \* в адресе заменяет любой символ), если не используется [NULL].

**Возвращает**

При успешном завершении функция возвращает VCLIENT\_SUCCESS, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
VCLIENT_ERROR_CONNECT(*)	Ошибка. Не удалось создать соединение. (*) - Имеет десятичное представление <i>lppp</i> , где <i>ppp</i> - дополнительный код ошибки детали см. <a href="#">Коды ошибок</a>
VCLIENT_ERROR_CHANGE_DIR	Ошибка. Не удалось выполнить смену каталога

**Дополнительная информация**

Параметр [valid\\_time](#) ([Время хранения сертификатов в кэше](#)) должен принимать ненулевое значение (рекомендуемые значения 600 - 3600). Использование кэша оправдано тем, что это практически не влияет на безопасность выполнения операций, но позволяет избавиться от ошибок, вызванных невозможностью соединения с хранилищем сертификатов в какой-то момент времени.

**Файл описания**

Прототип описан в файле [vclient.h](#).

**3.1.2.3 vcList****Назначение**

Функция **vcList** получает список файлов и каталогов для установленного соединения.

```
int WINAPI vcList (HANDLE hDst, char * sort_rule, VLISTITEM ** list, int * count);
```

**Параметры****hDst [in]**

Дескриптор соединения. В качестве параметра может выступать дескриптор полученный из вызовов [vOpenLocal](#) и [vOpenRemote](#).



**sort\_rule [in]**

Цепочка символов, определяющих правила сортировки списка сообщений.

**list [out]**

Указатель на список структур типа [VLISTITEM](#).

**count [out]**

Количество элементов в списке структур.

**Возвращает**

При успешном завершении функция возвращает `VCLIENT_SUCCESS`, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
<code>VCLIENT_ERROR_HANDLE</code>	Ошибка. Неверный дескриптор соединения.
<code>VCLIENT_ERROR_USER_ABORT</code>	Ошибка. Операция прервана пользователем.
<code>VCLIENT_ERROR_LIST</code>	Ошибка. Не удалось получить список файлов.

**Особенности выполнения**

В цепочке возможна комбинация следующих символов:

- N – сортировка по имени файлов сообщений (референсу);
- T – сортировка по времени создания файлов сообщений;
- S – сортировка по размеру файлов сообщений;
- P – сортировка по приоритету файлов сообщений.

Сортировка выполняется последовательно от первого символа до последнего. Элементы в отсортированном списке располагаются от меньшего к большему, а именно:

- отсортированные по имени – от меньшего референса к большему (например, имя файла "AAA..." меньше "BBB...");
- отсортированные по времени создания – от старых к новым;
- отсортированные по размеру – от меньшего размера к большему;
- отсортированные по приоритету от большего приоритета к меньшему.

Параметр `sort_rule` может быть `NULL`. В этом случае будет выполняться правило сортировки по умолчанию ("T").



**Примечание:** Если правило содержит сортировку по времени, то она выполняется с точностью (секунды и доли секунды), определяемой операционной и файловой системами. `vcList()` работает с локальной файловой системой и удаленным сервером CSFTP, при этом точность сортировки определяется соответственно локальной или удаленной системой.



**Примечание:** Если требуется сортировка в обратном порядке, то цепочка символов может включать:

- n – сортировка по имени файлов сообщений (референсу) в обратном порядке;
- t – сортировка по времени создания файлов сообщений в обратном порядке;
- s – сортировка по размеру файлов сообщений в обратном порядке;
- p – сортировка по приоритету файлов сообщений в обратном порядке.

Если сортировка не требуется, то первый символ в цепочке должен быть 'U'.

**Файл описания**

Прототип описан в файле [vclient.h](#).

### 3.1.2.4 vcCopyMessage

#### Назначение

Функция **vcCopyMessage** копирует сообщение между любыми двумя типами соединений.

```
int WINAPI vcCopyMessage (HANDLE hDst, const char * file, HANDLE hSrc, unsigned char ** conf_data,
long * conf_size, char * conf_message);
```

#### Параметры

##### **hDst [in]**

Дескриптор соединения. В качестве параметра может выступать дескриптор полученный из вызовов vOpenLocal и vOpenRemote.

##### **file [in]**

Имя файла (без маршрута).

##### **hSrc [in]**

Дескриптор соединения, откуда будет копироваться сообщение.

##### **conf\_data [out]**

Указатель, на буфер содержащий подтверждение об отправке файла.

##### **conf\_size [out]**

Размер буфера с подтверждением.

##### **conf\_message [out]**

Строка с сообщением в полученном подтверждении или строка с сообщением о проверке сообщения (в отправленном подтверждении).

#### Возвращает

При успешном завершении функция возвращает VCLIENT\_SUCCESS, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
VCLIENT_ERROR_HANDLE	Ошибка. Неверный дескриптор соединения.
VCLIENT_ERROR_USER_ABORT	Ошибка. Операция терминирована пользователем.
VCLIENT_ERROR_CHANGE_DIR	Ошибка. Не удалось выполнить смену каталога.
VCLIENT_ERROR_MESS_COPY	Ошибка. Не удалось передать/получить сообщение.
VCLIENT_ERROR_MESS_CONFIRM_CREATE	Ошибка. Не удалось создать подтверждение на сообщение.
VCLIENT_ERROR_MESS_CONFIRM_COPY	Ошибка. Не удалось передать/получить подтверждение на сообщение.
VCLIENT_ERROR_MESS_CONFIRM_OPEN	Ошибка. Не удалось открыть подтверждение на сообщение.
VCLIENT_ERROR_MESS_CONFIRM_SIZE	Ошибка. Превышен размер подтверждения на сообщение.
VCLIENT_ERROR_MESS_CONFIRM_READ	Ошибка при получении подтверждения на сообщение.
VCLIENT_ERROR_MESS_CONFIRM_WRITE	Ошибка при отправке подтверждения на сообщение.
VCLIENT_ERROR_MESS_CONFIRM_CHECK	Ошибка при проверке подтверждения на сообщение.

В случае если при проверке ЭЦП подтверждения на сообщение возникнет ситуация: нет кэша / нет связи с хранилищем / нет связи с сервером, то будет возвращен код возврата 2014:2 (см. [Коды ошибок](#)).

### Дополнительная информация

1. Сообщение пересылается с локальной файловой системы на сервер транспортного узла FASTi2 (hDst-удаленное соединение, hSrc-локальное соединение): только в этом случае предусмотрено подтверждение, которое помещается в буфер conf\_data и имеет размер conf\_size, а строка с сообщением в полученном подтверждении записывается в conf\_message.

Возможен вариант, когда функция вернет VCLIENT\_SUCCESS, что означает, что не было ошибок в протоколе обмена, сообщение отправлено и на него получено подтверждение с подлинной ЭЦП, но при этом может оказаться, что при проверке сообщения на сервере возникла ошибка. В этом случае, сервер должен указать в ответном подтверждении код ошибки.

Проверка сообщения включает в себя проверку формата, проверку целостности сообщения, проверку всех электронных цифровых подписей. При возникновении ошибки при проверке сообщения на сервере, это не отражается на основном коде возврата функции, это сделано, намерено, чтобы отделить непосредственно ошибки передачи и ошибки в содержимом отправляемого сообщения. При наличии ошибок при проверке сообщения в дополнительном коде возврата устанавливается предупреждение со значением равным WARNING\_CONFIRM\_CONTENT\_ERROR. В таких ситуациях при использовании API, эксперт по создаваемой информационной системе, должен сам решить, что нужно сделать, если например одна подпись из нескольких оказалась не подлинной.



**Важное замечание:** После вызова функции в этом режиме, необходимо проанализировать дополнительный код возврата или содержимое строки подтверждения conf\_message, в которой указывается результат проверки сообщения на сервере транспортного узла FASTi2.

- Сообщение отправлено и проверено без ошибок, тогда код возврата VCLIENT\_SUCCESS, в строке conf\_message: REFERENCE OF MESSAGE OK
- Сообщение отправлено и при проверке сервером возникла ошибка, тогда код возврата VCLIENT\_SUCCESS, в строке conf\_message: REFERENCE OF MESSAGE ERROR=<код ошибки>:[дополнительный код ошибки]

Дополнительный код возврата будет установлен в WARNING\_CONFIRM\_CONTENT\_ERROR.

2. Сообщение пересылается с удаленного сервера транспортного узла FASTi2 на локальную файловую систему (hDst-удалённое соединение, hSrc-локальное соединение): в параметр conf\_message записывается строка с информацией о проверке полученного сообщения, в параметр conf\_size -размер отправленного серверу подтверждения:
  - Сообщение получено от сервера и проверено сервером без ошибок: REFERENCE OF MESSAGE OK Серверу отправлено подтверждение с эти сообщением.
  - Сообщение получено от сервера и при проверке сервером возникла ошибка: REFERENCE OF MESSAGE ERROR=<код ошибки>:[дополнительный код ошибки]. Серверу отправлено подтверждение с этим сообщением об ошибке.
  - Сообщение получено от сервера и при проверке одна из подписей оказалась неверной: REFERENCE OF MESSAGE ERROR=<код ошибки>, Signature №<номер подписи в сообщении> (<всего подписей>) <имя подписавшего> [<время подписи>] is failed. Серверу отправлено подтверждение с этим сообщением об ошибке.

### Файл описания

Прототип описан в файле [vclient.h](#).

### 3.1.2.5 vcDelMessage

#### Назначение

Функция **vcDelMessage** копирует сообщение между любыми двумя типами соединений.

```
int WINAPI vcDelMessage (HANDLE hDst, const char * file);
```

## Параметры

### **hDst [in]**

Дескриптор соединения. В качестве параметра может выступать дескриптор полученный из вызовов [vOpenLocal](#) и [vOpenRemote](#).

### **file [in]**

Имя файла.

## Возвращает

При успешном завершении функция возвращает VCLIENT\_SUCCESS, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
VCLIENT_ERROR_HANDLE	Ошибка. Неверный дескриптор соединения.
VCLIENT_ERROR_USER_ABORT	Ошибка. Операция терминирована пользователем.
VCLIENT_ERROR_CHANGE_DIR	Ошибка. Не удалось выполнить смену каталога.
VCLIENT_ERROR_COMM_DELETE	Ошибка при передаче команды удаления сообщения.
VCLIENT_ERROR_MESS_CONFIRM_CREATE	Ошибка. Не удалось создать подтверждение для команды.
VCLIENT_ERROR_MESS_CONFIRM_OPEN	Ошибка. Не удалось открыть подтверждение для команды.
VCLIENT_ERROR_MESS_CONFIRM_WRITE	Ошибка при отправке подтверждения для команды.

## Файл описания

Прототип описан в файле [vclient.h](#).

### 3.1.2.6 vcGetInfoMessage

#### Назначение

Функция **vcGetInfoMessage** возвращает криптографическую информацию о сообщении.

```
int WINAPI vcGetInfoMessage (HANDLE hDst, const char * file, unsigned char ** envp_data, unsigned long * envp_size, unsigned char ** sign_data, unsigned long * sign_size);
```

## Параметры

### **hDst [in]**

Дескриптор соединения. В качестве параметра может выступать дескриптор полученный из вызовов [vOpenLocal](#) и [vOpenRemote](#).

### **file [in]**

Имя файла (без маршрута).

### **envp\_data [out]**

Указатель на буфер с криптографическим блоком шифрования CMS Enveloped.

### **envp\_size [out]**

Размер буфера с криптографическим блоком шифрования CMS Enveloped.

### **sign\_data [out]**

Указатель на буфер с криптографическим блоком ЭЦП CMS Signed.

**sign\_size [out]**

Размер буфера с криптографическим блоком ЭЦП CMS Signed.

**Возвращает**

При успешном завершении функция возвращает VCLIENT\_SUCCESS, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
VCLIENT_ERROR_HANDLE	Ошибка. Неверный дескриптор соединения.
VCLIENT_ERROR_USER_ABORT	Ошибка. Операция терминирована пользователем.
VCLIENT_ERROR_MESS_OPEN	Ошибка. Не удалось открыть сообщение.
VCLIENT_ERROR_MESS_READ	Ошибка. Не удалось передать/получить сообщение.
VCLIENT_ERROR_MESS_CLOSE	Ошибка. Не удалось закрыть сообщение.
VCLIENT_ERROR_MESS_HEADER	Ошибка. Не удалось передать/получить заголовок сообщения.
VCLIENT_ERROR_MESS_CRYPT_OPEN	Ошибка. Не удалось открыть крипто-блока сообщения.
VCLIENT_ERROR_MESS_CRYPT_SIZE	Ошибка. Не удалось определить размер крипто-блока сообщения.
VCLIENT_ERROR_MESS_CRYPT_READ	Ошибка. Не удалось прочитать крипто-блок сообщения.
VCLIENT_ERROR_MESS_CRYPT_CLOSE	Ошибка. Не удалось закрыть крипто-блок сообщения.

**Файл описания**

Прототип описан в файле [vclient.h](#).

**3.1.2.7 vcClose****Назначение**

Функция **vcClose** закрывает соединение.

```
int WINAPI vcClose (HANDLE hDst);
```

**Параметры****hDst [in]**

Дескриптор соединения. В качестве параметра может выступать дескриптор полученный из вызовов [vOpenLocal](#) и [vOpenRemote](#).

**Возвращает**

При успешном завершении функция возвращает VCLIENT\_SUCCESS, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
VCLIENT_ERROR_HANDLE	Ошибка. Неверный дескриптор соединения

**Файл описания**

Прототип описан в файле [vclient.h](#).

### 3.1.2.8 vcAbort

#### Назначение

Функция **vcAbort** экстренно прерывает выполняемую операцию.

```
int WINAPI vcAbort (HANDLE hDst);
```

#### Параметры

**hDst [in]**

Дескриптор соединения. В качестве параметра может выступать дескриптор полученный из вызовов **vOpenLocal** и **vOpenRemote**.

#### Возвращает

При успешном завершении функция возвращает **VCLIENT\_SUCCESS**, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
VCLIENT_ERROR_HANDLE	Ошибка. Неверный дескриптор соединения

#### Файл описания

Прототип описан в файле **vcclient.h**.

### 3.1.2.9 vcRegCallback

#### Назначение

Функция **vcRegCallback** позволяет зарегистрировать пользовательскую **CallBack**-функцию для визуализации прогресса при работе с архивом.

```
int WINAPI vcRegCallback (HANDLE hDst, HANDLE cont, callbacker function);
```

#### Параметры

**hDst [in]**

Дескриптор соединения. В качестве параметра может выступать дескриптор полученный из вызовов **vOpenLocal** и **vOpenRemote**.

**count [in]**

Дескриптор пользовательской **CallBack**-функции.

**function [in]**

Пользовательская функция.

#### Возвращает

При успешном завершении функция возвращает **VCLIENT\_SUCCESS**, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
VCLIENT_ERROR_HANDLE	Указан ошибочный дескриптор соединения

#### Файл описания

Прототип описан в файле **vcclient.h**.

### 3.1.2.10 vcGetEnvpInfo

#### Назначение

Функция **vcGetEnvpInfo** возвращает криптографическую информацию о блоке шифрования CMS Enveloped.

```
int WINAPI vcGetEnvpInfo (HANDLE hDst, unsigned char * envp_data, unsigned long envp_size, time_t * time_create, char * sender, char ** recipients, int * count);
```

#### Параметры

##### **hDst** [in]

Дескриптор соединения. В качестве параметра может выступать дескриптор полученный из вызовов [vOpenLocal](#) и [vOpenRemote](#).

##### **envp\_data** [in]

Указатель на буфер с криптографическим блоком шифрования (CMS Enveloped).

##### **envp\_size** [in]

Размер буфера с криптографическим блоком шифрования.

##### **time\_create** [out]

Время создания.

##### **sender** [out]

Указатель на буфер, содержащий имя отправителя в формате DN (имя пользователя возвращается без завершающего символа "точка с запятой").

##### **recipients** [out]

Указатель на массив буферов, с именами получателей в формате DN.

##### **conf** [out]

Количество получателей в сообщении.

#### Возвращает

При успешном завершении функция возвращает VCLIENT\_SUCCESS, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
VCLIENT_ERROR_HANDLE	Ошибка. Неверный дескриптор соединения
VCLIENT_ERROR_BLOCK_ENVELOPED	Ошибка. Неверная структура криптографического блока шифрования

#### Файл описания

Прототип описан в файле [vclient.h](#).

### 3.1.2.11 vcGetSignInfo

#### Назначение

Функция **vcGetSignInfo** возвращает криптографическую информацию о блоке электронной цифровой подписи CMS Signed.

```
int WINAPI vcGetSignInfo (HANDLE hDst, unsigned char * sign_data, unsigned long sign_size, char ** signers, time_t ** sign_time, int * count);
```

## Параметры

### **hDst [in]**

Дескриптор соединения. В качестве параметра может выступать дескриптор полученный из вызовов [vOpenLocal](#) и [vOpenRemote](#).

### **sign\_data [in]**

Указатель на буфер с криптографическим блоком ЭЦП (CMS Signed).

### **sign\_size [out]**

Размер буфера с криптографическим блоком ЭЦП.

### **signers [out]**

Указатель на массив буферов, содержащих имена подписавших в формате DN (можно передать NULL, чтобы узнать кол-во пользователей подписавших сообщение).

### **sign\_time [out]**

Указатель на массив буферов, для записи времен подписи (можно передать NULL, чтобы узнать кол-во пользователей подписавших сообщение).

### **count [out]**

Количество выделенных элементов в массивах signers и sign\_time.

## Возвращает

При успешном завершении функция возвращает VCLIENT\_SUCCESS, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
VCLIENT_ERROR_HANDLE	Ошибка. Неверный дескриптор соединения
VCLIENT_ERROR_BLOCK_SIGNED	Ошибка. Неверная структура криптографического блока ЭЦП

## Файл описания

Прототип описан в файле [vclient.h](#).

### 3.1.2.12 vcGetLastError

## Назначение

Функция **vcGetLastError** позволяет получить уточненные (расширенные) [коды ошибок](#) функций.

int WINAPI **vcGetLastError** (HANDLE **hDst**);

## Параметры

### **hDst [in]**

Дескриптор соединения. В качестве параметра может выступать дескриптор полученный из вызовов [vOpenLocal](#) и [vOpenRemote](#).

## Файл описания

Прототип описан в файле [vclient.h](#).



### 3.1.2.13 vcGetErrorString

#### Назначение

Функция **vcGetErrorString** возвращает содержательное описание ошибки в буфере, адрес которого передал пользователь.

```
size_t CALLSPEC vcGetErrorString (int err_code, int ext_err_code, char * err_string, size_t err_string_size, int lang);
```

#### Параметры

**err\_code [in]**

Код ошибки.

**ext\_err\_code [in]**

Расширенный код ошибки.

**err\_string [in]**

Указатель на буфер под сообщение об ошибке.

**err\_string\_size [in]**

Размер буфера под сообщение об ошибке.

**lang [in]**

Код выбора языка сообщения об ошибке: 0 - русский, 1 - английский.

#### Возвращает

При успешном завершении функция возвращает размер сообщения об ошибке.

Если коду ошибки не установлено строковое значение, то будет возвращена строка: "Unknown error".

Функция также может вызываться при неудачной попытке создания дескриптора. Например: функция **vcOpenRemote** вернула код возврата, при этом дескриптор соединения **hDst** не создан, тогда можно вызвать (NULL,code,strerr).

#### Дополнительная информация

Буфер под сообщение об ошибке выделяет пользователь. Если в качестве указателя на буфер под сообщение об ошибке передать NULL, то функция вернет длину сообщения об ошибке.

#### Файл описания

Прототип описан в файле **vcclient.h**.

### 3.1.2.14 vcIsRemote

#### Назначение

Функция **vcIsRemote** получает информацию о соединении: файловая система/удаленная система.

```
bool CALLSPEC vcIsRemote (HANDLE hDst);
```

#### Параметры

**hDst [in]**

Дескриптор соединения.

#### Возвращает

При успешном завершении функция возвращает true - удаленная, false - локальная.

## Файл описания

Прототип описан в файле [vclient.h](#).

### 3.1.2.15 VCLIENT.H

#### Назначение

Описание констант и прототипов функций работы с сервером транспортного узла

```

#ifndef __VCLIENT_H
#define __VCLIENT_H

#include <time.h>

#ifndef VISTA_DEFS
#define VISTA_DEFS
typedef int      long32;
typedef unsigned int ulong32;
typedef long32 VTIME_T;
#endif
#ifdef __cplusplus
typedef unsigned char bool;
#endif
#if defined(_WIN32) || defined(WIN32)
#include <windows.h>
#define WAPI __stdcall
#define CALLSPEC __declspec(dllimport) WAPI
#else
typedef void *HANDLE;
#define WAPI
#define CALLSPEC
#define _export
#endif
#endif

#define VCLIENT_ERROR_CONNECT          1000
#define VCLIENT_ERROR_CHANGE_DIR      2001
#define VCLIENT_ERROR_USER_ABORT      2002
#define VCLIENT_ERROR_LIST            2003
#define VCLIENT_ERROR_MESS_COPY       2004
#define VCLIENT_ERROR_MESS_OPEN       2005
#define VCLIENT_ERROR_MESS_READ       2006
#define VCLIENT_ERROR_MESS_CLOSE      2007
#define VCLIENT_ERROR_MESS_HEADER     2008
#define VCLIENT_ERROR_MESS_CONFIRM_COPY 2009
#define VCLIENT_ERROR_MESS_CONFIRM_OPEN 2010
#define VCLIENT_ERROR_MESS_CONFIRM_SIZE 2011
#define VCLIENT_ERROR_MESS_CONFIRM_READ 2012
#define VCLIENT_ERROR_MESS_CONFIRM_WRITE 2013
#define VCLIENT_ERROR_MESS_CONFIRM_CHECK 2014
#define VCLIENT_ERROR_MESS_CONFIRM_CREATE 2015
#define VCLIENT_ERROR_MESS_CRYPT_OPEN 2016
#define VCLIENT_ERROR_MESS_CRYPT_SIZE 2017
#define VCLIENT_ERROR_MESS_CRYPT_READ 2018
#define VCLIENT_ERROR_MESS_CRYPT_CLOSE 2019
#define VCLIENT_ERROR_COMM_DELETE     2020
#define VCLIENT_ERROR_COMM_CONFIRM_CREATE 2021
#define VCLIENT_ERROR_COMM_CONFIRM_OPEN 2022
#define VCLIENT_ERROR_COMM_CONFIRM_WRITE 2023
#define VCLIENT_ERROR_BLOCK_ENVELOPED 2024
#define VCLIENT_ERROR_BLOCK_SIGNED    2025
#define VCLIENT_ERROR_HANDLE          2026
#define VCLIENT_ERROR_MESS_CHECK      2027
#define VCLIENT_ERROR_PARAM           2028

#define VP_STORE_URL      1
#define VP_STORE_PROXY    2
#define VP_STORE_VALID    3
#define VP_CA_BODY        4
#define VP_CA_PATH        5
#define VP_CA_URL         6
#define VP_CRL_BODY       7
#define VP_CRL_PATH       8
#define VP_CRL_URL        9
#define VP_CMS_USAGE      10

#define VC_INC_CERT_SIGN 1

```

```

#define VC_INC_CERT_EXCH 2
#define VC_INC_CRL 4

//-----
#ifndef VLISTITEM_STRUCT_H
#define VLISTITEM_STRUCT_H

/* Структура VLISTITEM определяет элемент списка,
   возвращаемого функцией vcList() */
typedef struct
{
    int         type;
    char        *name;
    ulong32     size;
    ulong32     c_time;
    ulong32     mode;
    int         nlink;
    int         priority;
} VLISTITEM;

#endif

typedef void (*callback)( HANDLE cont, int operation, long complete, long fullsize,
    char *namemess );

//-----

#ifdef __cplusplus
extern "C" {
#endif
//-----
int     CALLSPEC vcOpenRemote( HANDLE *pipe, char *host, char *port, char *prof, char
    *mydn, int type, int ltout, int stout, char *srdn, char *store_host, int store_port,
    int valid_time, int proxy_type, char *proxy_host, int proxy_port, char *proxy_user,
    char *proxy_pass, char *proxy_exclude_list );
int     CALLSPEC vcOpenLocal( HANDLE *pipe, char *location );
int     CALLSPEC vcList( HANDLE pipe, char* sort_rule, VLISTITEM **list, int *count );
int     CALLSPEC vcCopyMessage( HANDLE pipe, const char *file, HANDLE from, unsigned char
    **conf_data, long *conf_size, char *conf_mess );
int     CALLSPEC vcDelMessage( HANDLE pipe, const char *file );
int     CALLSPEC vcGetInfoMessage( HANDLE pipe, const char *file, unsigned char
    **envp_data, unsigned long *envp_size, unsigned char **sign_data, unsigned long
    *sign_size );
int     CALLSPEC vcGetInfoMessageEx( HANDLE pipe, const char *file, unsigned char
    **envp_data, unsigned long *envp_size, unsigned char **sign_data, unsigned long
    *sign_size, char *memo );
int     CALLSPEC vcClose( HANDLE pipe );
int     CALLSPEC vcAbort( HANDLE pipe );
int     CALLSPEC vcRegCallback ( HANDLE pipe, HANDLE cont, callback function );
int     CALLSPEC vcGetEnvpInfo( HANDLE pipe, unsigned char *envp_data, unsigned long
    envp_size, time_t *time_create, char *sender, char **recipients, int *count );
int     CALLSPEC vcGetSignInfo( HANDLE pipe, unsigned char *sign_data, unsigned long
    sign_size, char **signers, time_t **signtime, int *count );
bool    CALLSPEC vcIsRemote ( HANDLE hDst );
int     CALLSPEC vcGetLastError( HANDLE pipe );
size_t  CALLSPEC vcGetErrorString( int err_code, int ext_err_code, char* err_string,
    size_t err_string_size, int lang );
int     CALLSPEC vcChangeDir( HANDLE pipe, char *path );
int     CALLSPEC vcSetParam( HANDLE pipe, ulong32 dwParam, unsigned char* pbData, ulong32
    cbData, ulong32 dwFlags );
//-----
#ifdef __cplusplus
}
#endif
#endif

```

### 3.1.2.16 Структура VLISTITEM

#### Назначение

Структура **VLISTITEM** определяет элемент списка, возвращаемого функцией **vcList()**.

```

typedef struct {int type; char * name; ulong32 size; ulong32 c_time; ulong32 mode; int nlink; int priority}
VLISTITEM;

```

## Параметры

### type

Определяет тип элемента 1 – файл, 0 – каталог.

### name

Имя файла/каталога.

### size

Размер файла.

### c\_time

Время создания файла/каталога.

### mode

Права доступа на файл/каталог (аналог элемента st\_mode стандартной структуры stat, описанной в заголовочном файле sys/stat.h).

### nlink

Количество линков на файл/каталог (второй столбец при выполнении команды `ls -l` в ОС Unix/Linux).

### priority

Приоритет сообщения.

## Файл описания

Прототип описан в файле [vclient.h](#).

### 3.1.3 Функции работы с уведомлениями транспортного узла

Таблица 5.

Функции	Описание
int WINAPI CMSSignedInfo (HANDLE *phCMS, unsigned char **cms_data, long cms_size, CMS_SIGN_INFO **cms)	Функция <a href="#">CMSSignedInfo</a> получает информацию о CMS
int WINAPI FreeCMSSignedInfo (HANDLE hCMS)	Функция <a href="#">FreeCMSSignedInfo</a> закрывает дескриптор, освобождает память

#### 3.1.3.1 CMSSignedInfo

### Назначение

Функция [CMSSignedInfo](#) получает информацию о CMS.

```
int CALLSPEC CMSSignedInfo (HANDLE * phCMS, unsigned char * cms_data, long cms_size, CMS_SIGN_INFO ** cms);
```

### Параметры

#### cms\_data [in]

Адрес буфера памяти, содержащий подтверждение.

#### cms\_size [in]

Размер буфера с подтверждением.

#### phCMS [out]

Адрес буфера памяти, в который функция копирует дескриптор объекта CMS.

**cms [out]**

Адрес, по которому функция копирует указатель на созданную структуру типа **CMS\_SIGN\_INFO**.

**Возвращает**

При успешном завершении функция возвращает **ARCH\_SUCCESS**.

В случае ошибки - расширенный код возврата серии **TCMMIMPLE\_XXX RR\_ARCH\_PARAM** - ошибка в параметрах функции.

**Файл описания**

Прототип описан в файле [carch.h](#).

**3.1.3.2 FreeCMSSignedInfo****Назначение**

Функция **FreeCMSSignedInfo** закрывает дескриптор, освобождает память.

```
int CALLSPEC FreeCMSSignedInfo (HANDLE hCMS);
```

**Параметры****hCMS [in]**

Дескриптор объекта CMS.

**Возвращает**

При успешном завершении функция возвращает **ARCH\_SUCCESS**.

В случае ошибки - расширенный код возврата серии **TCMMIMPLE\_XXX RR\_ARCH\_PARAM** - ошибка в параметрах функции.

**Файл описания**

Прототип описан в файле [carch.h](#).

**3.1.3.3 Структура CMS\_SIGN\_INFO****Назначение**

Структура **CMS\_SIGN\_INFO** предоставляет возможность работы с подтверждениями.

```
typedef struct {int Version; char * ContentType; unsigned char * ContentData; long ContentSize;  
USER_SIGN_INFO ** Signers; int SignersCount;} CMS_SIGN_INFO;
```

**Параметры****Version**

Определяет версию CMS.

**ContentType**

OID содержимого CMS.

**ContentData**

Вложенные в CMS данные.

**ContentSize**

Размер вложенных в CMS данных.

## Signers

Подписи пользователей.

## SignersCount

Количество подписей пользователей.

## Файл описания

Прототип описан в файле [carch.h](#).

### 3.1.3.4 Структура USER\_SIGN\_INFO

#### Назначение

Структура **USER\_SIGN\_INFO** предоставляет возможность работы с подтверждениями.

```
typedef struct {int Version; char * userDN; char * Issuer; unsigned char * SerialData; long SerialSize; char * HashAlgOID; unsigned char * HashData; long HashSize; char * SignAlgOID; unsigned char * SignData; long SignSize; char * SignTimeStr; unsigned char * AttrData; long AttrSize; VTIME_T SignTime; unsigned char * OKeyData; long OKeySize;} USER_SIGN_INFO;
```

#### Параметры

##### Version

Определяет версию библиотеки.

##### userDN

Имя подписавшего в формате DN.

##### Issuer

Имя центра сертификации, выпустившего сертификат в формате DN.

##### SerialData

Серийный номер сертификата.

##### SerialSize

Размер (длина) серийного номера.

##### HashAlgOID

OID хэш-алгоритма.

##### HashData

Значение хэш-вектора.

##### HashSize

Определяет размер хэш-вектора.

##### SignAlgOID

OID алгоритма электронной цифровой подписи (ЭЦП).

##### SignData

Электронная цифровая подпись (ЭЦП).

##### SignSize

Определяет размер электронной цифровой подписи (ЭЦП).

##### SignTimeStr

Время постановки электронной цифровой подписи (ЭЦП) по Гринвичу.

##### AttrData

Данные атрибутов CMS.

**AtrrSize**

Размер данных атрибутов CMS.

**SignTime**

Время постановки электронной цифровой подписи (ЭЦП) в формате Generalized Time (SYNTAX 1.3.6.1.4.1.1466.115.121.1.24). .

**OKeyData**

Открытый ключ вложенного сертификата подписавшего.

**OKeySize**

Размер открытого ключа вложенного сертификата подписавшего.

**Файл описания**

Прототип описан в файле [carch.h](#).

**3.1.4 Функции работы с сертификатами транспортного узла**

Таблица 6.

Функции	Описание
int WINAPI GetCRL (HANDLE* phCRL, int type, char* destination, unsigned char** data, long* size, int* ext_code);	Функция <a href="#">GetCRL</a> предназначена для получения списка отозванных сертификатов СОС из хранилища или из файла
void WINAPI FreeCRL (HANDLE* hCRL);	Функция <a href="#">FreeCRL</a> предназначена для освобождения ресурсов после использования функции GetCRL()
int WINAPI SetVerify (char* profile, unsigned char* crl_data, long * crl_size, int* ext_code);	Функция <a href="#">SetVerify</a> предназначена для установки параметров для проверки сертификатов пользователей, используемых при формировании и проверке сообщений CMS Signed и CMS Envelopment, при аутентификации и организации защищённого протокола обмена
int WINAPI ChkVerify (void);	Функция <a href="#">ChkVerify</a> предназначена для определения необходимости установки параметров (самоподписанного сертификата Центра сертификации и Списка отозванных сертификатов) для проверки сертификатов пользователей

**3.1.4.1 GetCRL****Назначение**

Функция **GetCRL** предназначена для получения списка отозванных сертификатов СОС из хранилища или из файла.

```
int WINAPI GetCRL (HANDLE* phCRL, int type, char* destination, unsigned char** data, long* size, int* ext\_code);
```

**Параметры****type [in]**

Параметр, определяющий источник получения СОС. Возможные значения: 0 – получение из хранилища; 1 – получение из файла.

**destination [in]**

[in] для type = 0, URL LDAP хранилища сертификатов, содержащего СОС; для type = 1, полный путь к файлу, содержащему СОС. Например: "ldap://store.gamma.kz:62222/t=R;", "/Vista/etc/current.crl"

**phCRL [out]**

Указатель на дескриптор операции, для дальнейшего освобождения ресурсов.

**data [out]**

Указатель, на буфер содержащий СОС.

**size [out]**

Размер буфера с СОС.

**ext\_code [out]**

Дополнительный код возврата, в случае ошибки при выполнении.

**Возвращает**

При успешном завершении функция возвращает ARCH\_SUCCESS, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
ERR_ARCH_PARAM	Неверно заданы входные параметры
ERR_ARCH_GET_CRL	Не удалось получить СОС

**Дополнительная информация**

При успешном получении СОС, необходимо после использования выходных параметров, вызвать функцию [FreeCRL\(\)](#) для освобождения задействованных ресурсов.

При завершении с ошибкой для получения текстового описания ошибки необходимо вызвать функцию [CArchGetErrorString](#) (code, ext\_code, ...), в которую передать первым параметром код возврата функции *code*, а вторым - дополнительный код возврата *ext\_code*.

**Файл описания**

Прототип описан в файле [carch.h](#).

**3.1.4.2 FreeCRL****Назначение**

Функция **FreeCRL** предназначена для освобождения ресурсов после использования функции [GetCRL\(\)](#).

```
void WINAPI FreeCRL (HANDLE* hCRL);
```

**Параметры****hCRL [in]**

Дескриптор операции для освобождения ресурсов.

**Файл описания**

Прототип описан в файле [carch.h](#).



### 3.1.4.3 SetVerify

#### Назначение

Функция **SetVerify** предназначена для установки параметров для проверки сертификатов пользователей, используемых при формировании и проверке сообщений CMS Signed и CMS Envelopment, при аутентификации и организации защищённого протокола обмена. Установка сертификата Центра Сертификации (ЦС) выполняется из ключевого контейнера, соответствующего заданному имени профайла profile. При установке выполняется проверка ЭЦП сертификата ЦС и сроки его действия, проверка ЭЦП списка отозванных сертификатов, сроки его действия и соответствие ЦС.

```
int WINAPI SetVerify (char* profile, unsigned char* crt_data, long crt_size, int check_sign, int* ext_code);
```

#### Параметры

**profile** [in]

Имя профайла или имя контейнера крипто-модуля CSP.

**crt\_data** [in]

Указатель на буфер со списком отозванных сертификатов.

**crt\_size** [out]

Размер буфера со списком отозванных сертификатов.

**check\_sign** [out]

Возможные значения: 0 - не проверять ЭЦП списка отозванных сертификатов; 1 - проверять ЭЦП списка отозванных сертификатов.

**ext\_code** [out]

Дополнительный код возврата, в случае ошибки при выполнении.

#### Возвращает

При успешном завершении функция возвращает ARCH\_SUCCESS, в противном случае возвращается код ошибки. Возможные коды ошибок функции:

Возвращаемые значения	Описание
ERR_ARCH_SET_CA	ошибка при установке самоподписанного сертификата ЦС
ERR_ARCH_SET_CRL	ошибка при установке списка отозванных сертификатов

#### Дополнительная информация

При завершении с ошибкой для получения текстового описания ошибки необходимо вызвать функцию [CArchGetErrorString](#) (code, ext\_code, ...), в которую передать первым параметром код возврата функции code, а вторым - дополнительный код возврата ext\_code.

Функция устанавливает режим проверки сертификатов пользователей, который применяется для всех вызовов библиотеки.

#### Файл описания

Прототип описан в файле [carch.h](#).

### 3.1.4.4 ChkVerify

#### Назначение

Функция **ChkVerify** предназначена для определения необходимости установки параметров (самоподписанного сертификата Центра сертификации и Списка отозванных сертификатов) для проверки сертификатов пользователей.

```
int WINAPI ChkVerify (void);
```

#### Возвращает

Возвращаемые значения	Описание
0	нет необходимости установки, параметры установлены
1	есть необходимость установки, т.е. параметры не установлены

#### Файл описания

Прототип описан в файле [carch.h](#).

## 3.1.5 Функции смены сертификата ключа пользователя

Таблица 7.

Функции	Описание
int WINAPI vGetCrtValidity (char *pszContainer, ulong32 dwKeySpec, time_t *notBefore, time_t *notAfter, void *info, ulong32 *lastError);	Функция <a href="#">vGetCrtValidity</a> позволяет узнать даты валидности сертификата ключа на подпись или ключевой обмен
void WINAPI vGetCrtRequest (char *pszContainer, ulong32 dwKeySpec, char *DN, unsigned char *req, ulong32 *size, ulong32 *lastError);	Функция <a href="#">vGetCrtRequest</a> позволяет сформировать запрос на выпуск нового сертификата ключа на подпись или ключевой обмен
int WINAPI vSetCrtToCont (char *pszContainer, unsigned char *crt, ulong32 *size, ulong32 *lastError);	Функция <a href="#">vSetCrtToCont</a> позволяет установить выпущенный сертификат в контейнер криптопровайдера
int WINAPI vGetRequestTransID (unsigned char *req, ulong32 size, char *id);	Функция <a href="#">vGetRequestTransID</a> позволяет узнать уникальный номер сформированного запроса

### 3.1.5.1 vGetCrtValidity

#### Назначение

Функция **vGetCrtValidity** позволяет узнать даты валидности сертификата ключа на подпись или ключевой обмен.

```
int WINAPI vGetCrtValidity (char * pszContainer, ulong32 dwKeySpec, time_t * notBefore, time_t * notAfter, void * info, ulong32 * lastError);
```

#### Параметры

**pszContainer** [in]

Имя контейнера.

**dwKeySpec [in]**

Тип ключа AT\_KEYEXCHANGE(1)/AT\_SIGNATURE(2).

**notBefore [out]**

Начало действия сертификата.

**notAfter [out]**

Окончание действия сертификата.

**info [out]**

Зарезервировано, должно быть NULL.

**lastError [out]**

Код ошибки CSP.

**Возвращает**

Возвращаемые значения	Описание
0	нет необходимости установки, параметры установлены
1	есть необходимость установки, т.е. параметры не установлены

**Файл описания**

Прототип описан в файле [carch.h](#).

**3.1.5.2 vGetCrtRequest****Назначение**

Функция **vGetCrtRequest** позволяет сформировать запрос на выпуск нового сертификата ключа на подпись или ключевой обмен.

```
int WINAPI vGetCrtRequest (char * pszContainer, ulong32 dwKeySpec, char * DN, unsigned char * req,
ulong32 * size, ulong32 * lastError);
```

**Параметры****pszContainer [in]**

Имя контейнера.

**dwKeySpec [in]**

Тип ключа AT\_KEYEXCHANGE(1)/AT\_SIGNATURE(2).

**DN [out]**

Зарезервировано, должно быть NULL.

**req [out]**

Результат.

**size [out]**

Размер.

**lastError [out]**

Код ошибки CSP.

**Возвращает**

Возвращаемые значения	Описание
0	нет необходимости установки, параметры установлены
1	есть необходимость установки, т.е. параметры не установлены

**Файл описания**

Прототип описан в файле [carch.h](#).

**3.1.5.3 vSetCrtToCont****Назначение**

Функция **vSetCrtToCont** позволяет установить выпущенный сертификат в контейнер криптопровайдера.

```
int WINAPI vSetCrtToCont (char * pszContainer, unsigned char * crt, ulong32 * size, ulong32 * lastError);
```

**Параметры**

**[pszContainer](#) [in]**

Имя контейнера.

**[crt](#) [in]**

Тело сертификата.

**[size](#) [in]**

Размер сертификата.

**[lastError](#) [out]**

Код ошибки CSP.

**Файл описания**

Прототип описан в файле [carch.h](#).

**3.1.5.4 vGetRequestTransID****Назначение**

Функция **vGetRequestTransID** позволяет узнать уникальный номер сформированного запроса.

```
int WINAPI vGetRequestTransID (unsigned char * req, ulong32 size, char * id);
```

**Параметры**

**[req](#) [in]**

Запрос.

**[size](#) [in]**

Размер запроса.

**[id](#) [out]**

TransID (уникальный номер запроса).

**Файл описания**

Прототип описан в файле [carch.h](#).

## 3.2 Особенности работы с памятью

Следующие функции, входящие в состав библиотеки libvista.dll, в качестве выходных параметров возвращают адреса массивов переменной длины:

- CArchGetEnvelopInfo
- CArchListDir
- CArchGetComment
- CArchListArchFile
- CArchGetSignInfo
- vcList
- vcCopyMessage
- vcGetInfoMessage
- vcGetEnvpInfo
- vcGetSignInfo

В функциях реализован следующий механизм выделения и освобождения оперативной памяти: при вызове функции память выделяется непосредственно внутри функции, указатель на выделенный блок возвращается вызвавшей программе (пользователю). Пользователь не должен сам освобождать память. Память будет освобождена либо при последующем вызове функции, либо при вызове функций CArchClose(), vcClose() или vcAbort() при использовании libvista.dll.

## 3.3 Работа с отладочными библиотеками

Для записи протокола вызовов API библиотеки создают в каталогах временного хранения подкаталог debugx, в котором библиотеки пишут протоколы вызовов в файлы: carch\_debug.txt и vclient\_debug.txt соответственно в /tmp для Linux, HP-UX или результат функции GetTempPath() для Windows (\*).

(\*) Каталоги временного хранения должны существовать и на них должны быть установлены права доступа на запись.

В протокол записывается информация о вызове API: название метода, переданные параметры и коды возврата метода.

## 3.4 Примеры использования библиотеки libvista

Таблица 8.

Наименование примера	Описание примера	Ссылка на пример
Получение значений атрибутов из запроса на сертификат и ответов УЦ	Пример представляет программный код на языке Си, демонстрирующий как с использованием функций библиотеки libvista получить значение атрибута Transaction ID из запроса на сертификат (PKCS#7) и значений атрибутов Transaction ID, PKI Status, Fail Info из ответов УЦ (PKCS#7/CMS).	для ОС Windows: \SDK\API for Win\vGetCmsAttribute\ для ОС Linux/HP-UX ia64: \SDK\API for UNIX\vGetCmsAttribute\

Наименование примера	Описание примера	Ссылка на пример
Создание сообщения. Загрузка файлов из каталога	Пример представляет программный код на языке Си, демонстрирующий как с использованием функций библиотеки libvista создать зашифрованное и подписанное сообщение. Файлы для сообщения находятся в каталоге test. Полученное сообщение может быть использовано в других примерах работы (работа с архивом, добавление подписи, импорт сообщения, отправка сообщения и др.).	для ОС Windows: \SDK\API for Win\carch_create_msg\ для ОС Linux/HP-UX ia64: \SDK\API for UNIX\vcarch_create_msg\
Создание сообщения. Загрузка файлов из оперативной памяти	Пример представляет программный код на языке Си, демонстрирующий как с использованием функций библиотеки libvista создать зашифрованное и подписанное сообщение. Файлы для сообщения передаются через оперативную память. Полученное сообщение может быть использовано в других примерах работы (работа с архивом, добавление подписи, импорт сообщения, отправка сообщения и др.).	для ОС Windows: \SDK\API for Win\carch_create_msg_mem\ для ОС Linux/HP-UX ia64: \SDK\API for UNIX\carch_create_msg_mem\
Работа с сообщением. Извлечение содержимого файлов в каталог	Пример представляет программный код на языке Си, демонстрирующий как с использованием функций библиотеки libvista работать с сообщением. Исходный файл для данного примера можно создать, используя пример создания зашифрованного и подписанного ЭЦП сообщения. При извлечении файлов из архива будет создан каталог result.	для ОС Windows: \SDK\API for Win\carch_work_msg\ для ОС Linux/HP-UX ia64: \SDK\API for UNIX\carch_work_msg\
Работа с сообщением. Извлечение содержимого файлов на экран	Пример представляет программный код на языке Си, демонстрирующий как с использованием функций библиотеки libvista работать с сообщением. Исходный файл для данного примера можно создать, используя пример создания зашифрованного и подписанного ЭЦП сообщения. При извлечении файлов из архива их содержимое будет выводиться на экран.	для ОС Windows: \SDK\API for Win\carch_work_msg_mem\ для ОС Linux/HP-UX ia64: \SDK\API for UNIX\carch_work_msg_mem\

Наименование примера	Описание примера	Ссылка на пример
Работа с подтверждением	Пример представляет программный код на языке Си, демонстрирующий как с использованием функций библиотеки libvista работать с подтверждением о принятии сообщения.	для ОС Windows: \SDK\API for Win\carch_work_conf\ для ОС Linux/HP-UX ia64: \SDK\API for UNIX \carch_work_conf\
Добавление подписи	Пример представляет программный код на языке Си, демонстрирующий как с использованием функций библиотеки libvista добавить свою ЭЦП в сообщение VISTA. Исходный файл для данного примера можно создать, используя пример создания зашифрованного и подписанного ЭЦП сообщения.	для ОС Windows: \SDK\API for Win\carch_add_sign\ для ОС Linux/HP-UX ia64: \SDK\API for UNIX \carch_add_sign\
Добавление и проверка подписи	Пример представляет программный код на языке Си, демонстрирующий как с использованием функций библиотеки libvista добавить и проверить собственную подпись в сообщении.	для ОС Windows: \SDK\API for Win\carch_sign_verify\ для ОС Linux/HP-UX ia64: \SDK\API for UNIX \carch_sign_verify\
Отправка сообщения	Пример представляет программный код на языке Си, демонстрирующий как с использованием функций библиотеки libvista отправить сообщение с сервера транспортного узла.	для ОС Windows: \SDK\API for Win\vclient_send\ для ОС Linux/HP-UX ia64: \SDK\API for UNIX \vclient_send\
Прием сообщения	Пример представляет программный код на языке Си, демонстрирующий как с использованием функций библиотеки libvista получить сообщение с сервера транспортного узла.	для ОС Windows: \SDK\API for Win\vclient_recv\ для ОС Linux/HP-UX ia64: \SDK\API for UNIX \vclient_recv\
Получение типа соединения	Пример представляет программный код на языке Си, демонстрирующий как с использованием функций библиотеки libvista получить тип соединения - локальное или удаленное.	для ОС Windows: \SDK\API for Win\vclient_is_remote\ для ОС Linux/HP-UX ia64: \SDK\API for UNIX \vclient_is_remote\

Наименование примера	Описание примера	Ссылка на пример
Проверка лимита соединений	Пример представляет программный код на языке Си, демонстрирующий как с использованием функций библиотеки libvista проверить лимит соединений с сервером с одного IP-адреса.	для ОС Windows: \SDK\API for Win\vclient_limit_conn\ для ОС Linux/HP-UX ia64: \SDK\API for UNIX \vclient_limit_conn\
Формирование запроса на смену сертификата	Пример представляет программный код на языке Си, демонстрирующий как с использованием функций библиотеки libvista сформировать запрос на смену сертификата. Запросы сохраняются в автоматически формируемых файлах key_ex и key_sig.	для ОС Windows: \SDK\API for Win\vGetCrtRequest\ для ОС Linux/HP-UX ia64: \SDK\API for UNIX \vGetCrtRequest\
Установка сертификата в контейнер	Пример представляет программный код на языке Си, демонстрирующий как с использованием функций библиотеки libvista установить сертификат в ключевой контейнер. Исходные файлы, содержащие ответ сервера, для данного примера нужно поместить в каталог ..\bin (key_ex.req и key_sig.req).	для ОС Windows: \SDK\API for Win\vSetCrtToCont\ для ОС Linux/HP-UX ia64: \SDK\API for UNIX \vSetCrtToCont\
Получение сообщения об ошибке	Пример представляет программный код на языке Си, демонстрирующий как с использованием функций библиотеки libvista получить сообщения об ошибке, используя стандартный и <a href="#">расширенный код ошибки</a> .	для ОС Windows: \SDK\API for Win\CArchGetErrorString\ для ОС Linux/HP-UX ia64: \SDK\API for UNIX \CArchGetErrorString\
Контроль дескрипторов функций	Пример представляет программный код на языке Си, демонстрирующий как с использованием функций библиотеки libvista осуществлять контроль дескрипторов функций.	для ОС Windows: \SDK\API for Win\carch_vclient_desc\ для ОС Linux/HP-UX ia64: \SDK\API for UNIX \carch_vclient_desc\



## 4 Использование функций CryptoAPI

### 4.1 Примеры использования функций CryptoAPI

Таблица 9.

Наименование примера	Описание примера	Ссылка на пример
Формирование новых ключей для ЭЦП и ключевого обмена	Пример представляет программный код на языке Си, демонстрирующий как с использованием функций CryptoAPI осуществлять формирование запроса на смену сертификата.	Пример расположен в каталоге: \SDK\API for Win \CAPI_CrtRequest\
Удаление новых ключей из контейнера криптопровайдера	Пример представляет программный код на языке Си, демонстрирующий как с использованием функций CryptoAPI осуществлять удаление ключей из ключевого контейнера, имеющих статус "Новый".	Пример расположен в каталоге: \SDK\API for Win \CAPI_DelKey\
Получение выпущенных сертификатов из ответов УЦ (PKCS#7/CMS) и установка их в ключевой контейнер	Пример представляет программный код на языке Си, демонстрирующий как с использованием функций CryptoAPI осуществлять получение выпущенных сертификатов из ответов УЦ (PKCS#7/CMS) и устанавливать их в ключевой контейнер криптопровайдера.	Пример расположен в каталоге: \SDK\API for Win \CAPI_SetCert\
Формирование запроса на сертификат для нового ключа в формате PKCS#7	Пример представляет программный код на языке Си, демонстрирующий как с использованием функций CryptoAPI осуществлять формирование запроса на сертификат для нового ключа в формате PKCS#7.	Пример расположен в каталоге: \SDK\API for Win \CAPI_CrtRequest\
Получение значения атрибута Transaction ID из запроса на сертификат (PKCS#7)	Пример представляет программный код на языке Си, демонстрирующий как с использованием функций CryptoAPI получить значение атрибута Transaction ID из запроса на сертификат (PKCS#7).	Пример расположен в каталоге: \SDK\API for Win \CAPI_GetParamReq\

Наименование примера	Описание примера	Ссылка на пример
Получение значений атрибутов Transaction ID, PKI Status, Fail Info из ответов УЦ (PKCS#7/CMS)	Пример представляет программный код на языке Си, демонстрирующий как с использованием функций CryptoAPI получить значения атрибутов Transaction ID, PKI Status, Fail Info из ответов УЦ (PKCS#7/CMS).	Пример расположен в каталоге: \SDK\API for Win \CAPI_GetParamResp\

## 5 Использование библиотеки cptumar.dll

Таблица 10.

Имя файла	Описание
cptumar.h	Заголовочный файл
cptumar.dll	Библиотека для ОС Windows

### 5.1 Примеры использования библиотеки cptumar.dll

Таблица 11.

Наименование примера	Описание примера	Ссылка на пример
Удаление новых ключей из контейнера криптопровайдера	Пример представляет программный код на языке Си, демонстрирующий как с использованием функций библиотеки cptumar.dll осуществлять удаление ключей из ключевого контейнера, имеющих статус "Новый".	Пример расположен в каталоге: \SDK\API for Win \cspDelKey\

## 6 Выполнение примеров

### 6.1 Требования к программному и аппаратному обеспечению

Программное и аппаратное обеспечение, необходимое для демонстрации функционала на основе примеров:

1. Тестовый стенд для выполнения работ.
2. Программное средство криптографической защиты информации «ТУМАР-CSP» версии 6.2 и выше.
3. MS Visual Studio 2019 (с CMake) для ОС Windows
4. CLion 2022 для ОС Linux.
5. Ключевая информация пользователей для тестовых примеров.

### 6.2 Предварительные работы

1. Скопируйте с дистрибутивного диска на рабочую станцию каталог с примерами по использованию функций API FASTi2.
2. Запустите ПО «ТУМАР-CSP» и создайте профайлы для ключевой информации, указанной в каталоге `keys`.<sup>2</sup>
3. Укажите имена созданных профайлов в файле `common_params.h` в параметрах `CSP_PROFILE_USER1` и `CSP_PROFILE_USER2`).<sup>3</sup> Например:

```
const char CSP_PROFILE_USER1[200]{"sdk_user1"};
const char CSP_PROFILE_USER2[200]{"sdk_user2"};
```

4. Укажите DN-имена, соответствующие данной ключевой информации пользователей, в файле `common_params.h` в параметрах `MY_DN_USER1` и `MY_DN_USER2`.<sup>4</sup> Например:

```
const char MY_DN_USER1[1000]{ "CN=VS11.USER1;O=VS11;C=KZ" };
const char MY_DN_USER2[1000]{ "CN=VS11.USER2;O=VS11;C=KZ" };
```

5. Укажите DNS/EMAIL-адреса для DN-имен в файле `common_params.h` в параметрах `RCPT1` и `RCPT2`. DNS/EMAIL-адрес формируется из DN-имени в виде «CN+O+C». Например:

```
#define RCPT1 "USER1.VS11.VS11.KZ"
#define RCPT2 "USER2.VS11.VS11.KZ"
```

6. Убедитесь, что выполняется пинг-запрос LDAP-сервера, адрес сервера указан в файле `common_params.h` в параметре `LOCAL_STORE_HOST`.
7. Убедитесь, что выполняется пинг-запрос сервера FASTi2, адрес сервера указан в файле `common_params.h` в параметре `SERVER_HOST_ADDRESS`.

### 6.3 Компиляция SDK

**Основные шаги:**

1. Скопировать папки с примерами на диск пользователя.
2. Удостовериться что есть инструменты компиличования (MSVS, CLion, CMake, gcc).
3. Открыть проект в IDE (MSVS, CLion).
4. Настроить параметры в файле `include\common_params.h`.
5. Запустить компиличование «Build».
6. Запустить выполнение «Run».

<sup>2</sup> Наименования профайлов могут быть любые.

<sup>3</sup> В документе, в качестве примера, наименования профайлов следующие: `sdk_user1` и `sdk_user2`

<sup>4</sup> В документе, в качестве примера, наименования DN-имен следующие: `CN=VS11.USER1;O=VS11;C=KZ` и `CN=VS11.USER2;O=VS11;C=KZ`



**Примечание:** В примерах есть проект `sample_sdk`, который последовательно запускает все примеры. Порядок можно задать в массиве в файле `main.cpp`.



**Важное замечание:** В MSVC так же как и в CLion можно выбрать отдельный проект для запуска в среде.

### 6.3.1 Настройка параметров

Откройте файл `\include\common_params.h` и укажите следующие параметры:

1. Профайлы ПО «ТУМАР-CSP»:

```
CSP_PROFILE_USER1
CSP_PROFILE_USER2
```



**Важное замечание:** Для ОС Windows – наименования профайлов ПО «ТУМАР-CSP». Для ОС Linux – пути к ключевой информации.

2. DN-имена пользователей (имена отправителя и получателя):

```
MY_DN_USER1
MY_DN_USER2
```

3. Адрес сервера и порт FASTi2:

```
SERVER_HOST_ADDRESS
SERVER_PORT
```

4. Адрес хранилища сертификатов и порт:

```
LOCAL_STORE_HOST
LOCAL_STORE_PORT
```

5. Адреса пользователей (CN):

```
RCPT1
RCPT2
```

6. Остальные данные – каталоги для получения, для отправки и пр.

Пример файла `common_params.h`:

```
// Keys' data
const char CSP_PROFILE_USER1[500]{ "sdk_user1" };
const char CSP_PROFILE_USER2[500]{ "sdk_user2" };

// Users' data ( это DN имена профайлов, указанных выше )
const char MY_DN_USER1[1000]{ "CN=VS11.USER1;O=VS11;C=KZ" };
const char MY_DN_USER2[1000]{ "CN=VS11.USER2;O=VS11;C=KZ" };

// the LDAP' data
const char LOCAL_STORE_HOST[500]{ "172.16.172.65" };
const int LOCAL_STORE_PORT = 62222;

const char SERVER_HOST_ADDRESS[400]{ "VS11.VS11.KZ" };
const int SERVER_PORT = 62200;

#define RCPT1 "USER1.VS11.VS11.KZ"
#define RCPT2 "USER2.VS11.VS11.KZ"
```

Список возможных параметров:

- MY\_DN\_USER1 – имя пользователя 1 в формате DN;
- MY\_DN\_USER2 – имя пользователя 2 в формате DN;
- CSP\_PROFILE\_USER1 – имя профайла ключевого контейнера пользователя1;
- CSP\_PROFILE\_USER2 – имя профайла ключевого контейнера пользователя2;
- RCPT1 – имя пользователя 1 в формате DNS/EMAIL;

- RCPT2 – имя пользователя 2 в формате DNS/EMAIL;
- TEST\_FILE – имя файла используемого для демонстрации примеров (создаваемый/открываемый файл);
- MAX\_CONNECTION – максимальное число соединений с одного IP-адреса;
- LOCAL\_STORE\_HOST – имя хоста/IP-адрес хранилища сертификатов;
- LOCAL\_STORE\_PORT – порт хранилища сертификатов;
- SERVER\_HOST – имя хоста/IP-адрес сервера;
- SERVER\_PORT – порт сервера;
- MY\_STORE – URL хранилища сертификатов;
- SENT\_FOLDER – каталог сформированных CMS-сообщений для дальнейшей отправки;
- RECEIVED\_FOLDER – каталог полученных от сервера CMS-сообщений;
- BE\_CONFIRMED – флаг необходимости подтверждения.

### 6.3.2 Windows. MS Visual Studio

Компилирование и сборка примеров в MS Visual Studio 2019 (Community) (с CMake):



**Важное замечание:** Предварительно заполните файл `\include\common_params.h` актуальными данными.

1. Скопируйте с дистрибутивного диска на рабочую станцию каталог с примерами по использованию функций API FASTi2.
2. Запустите MS Visual Studio 2019 (с CMake).
3. Выберите **Открыть проект** → **Open a local folder**.

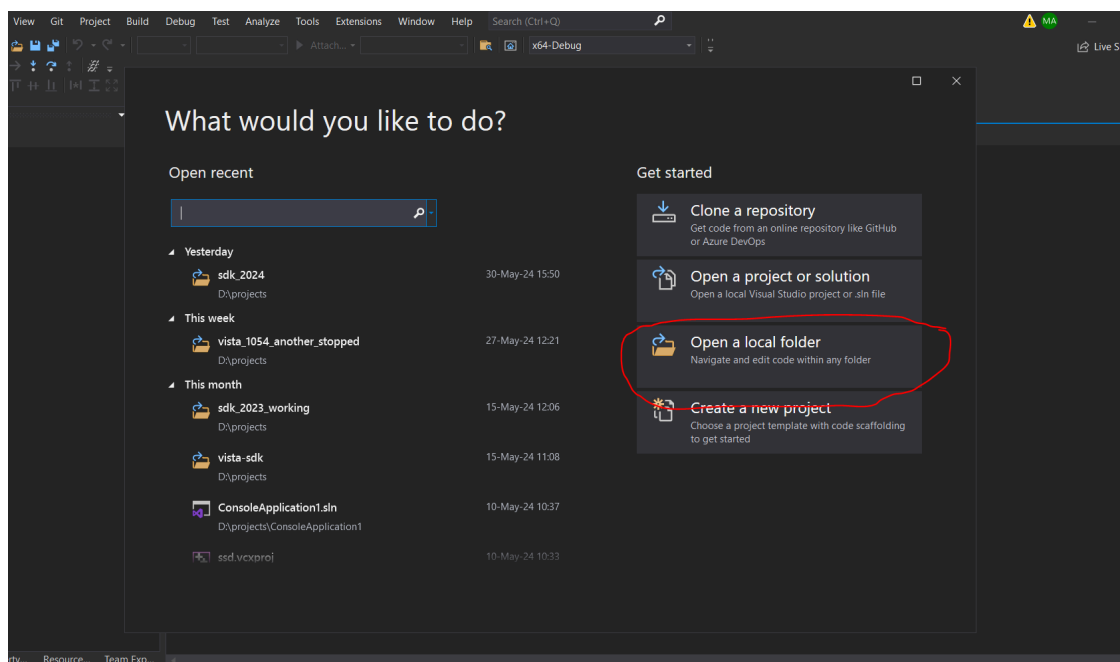


Рисунок 1. Запуск среды компиляции и открытие проекта в MSVS

4. Выберите каталог, в который были скопированы примеры и нажмите на кнопку **Select folder**.

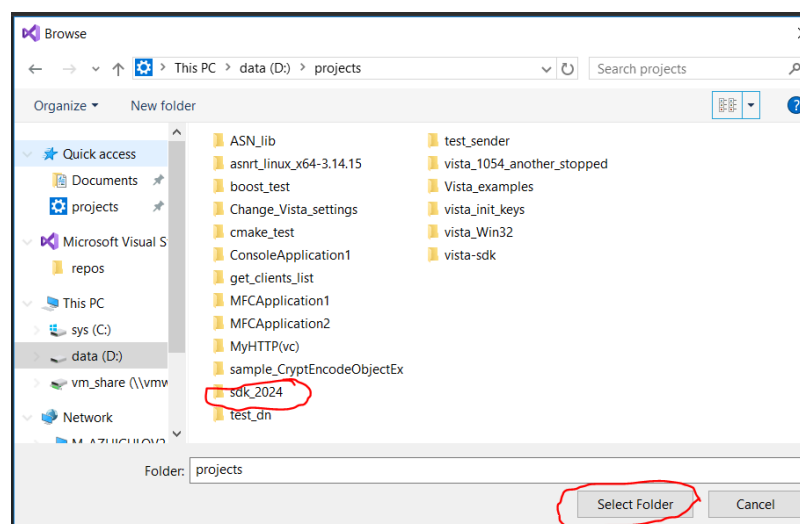


Рисунок 2. Диалоговое окно открытия файла проекта

5. В результате будет выполнена загрузка проекта.

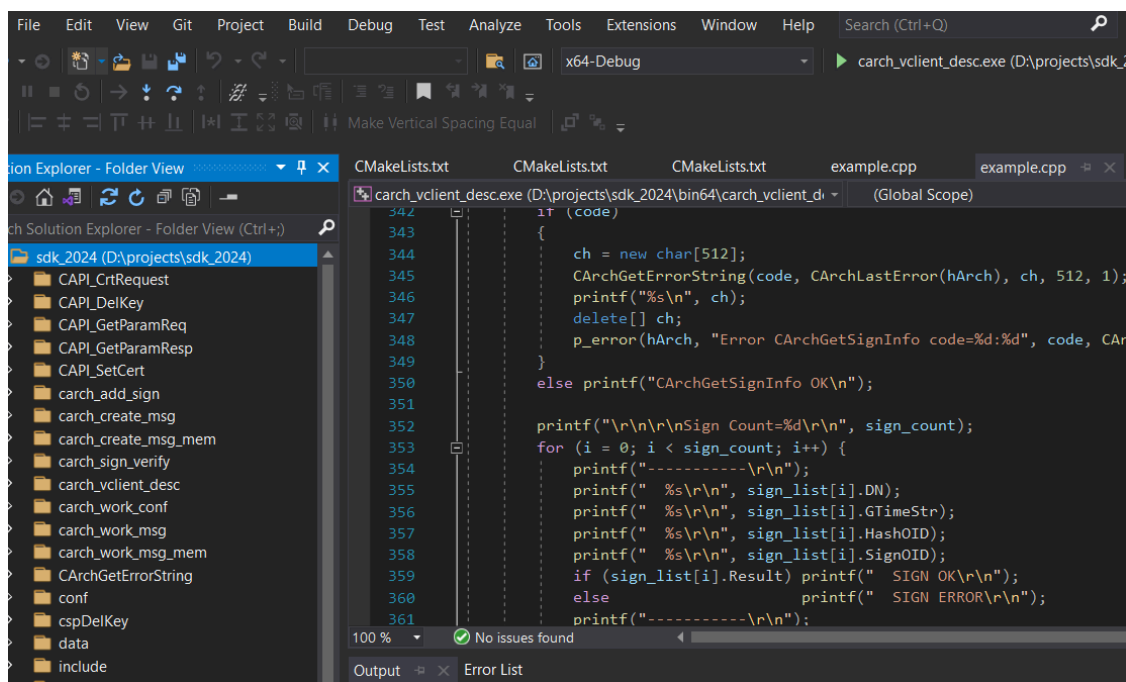


Рисунок 3. Результат открытия проекта в среде компиляции

6. Откройте **Менеджер проектов (Solution Explorer)**, выберите файл `CMakeLists.txt`. С помощью правой кнопки мыши вызовите контекстное меню и выберите пункт **Set as Star Up item**.

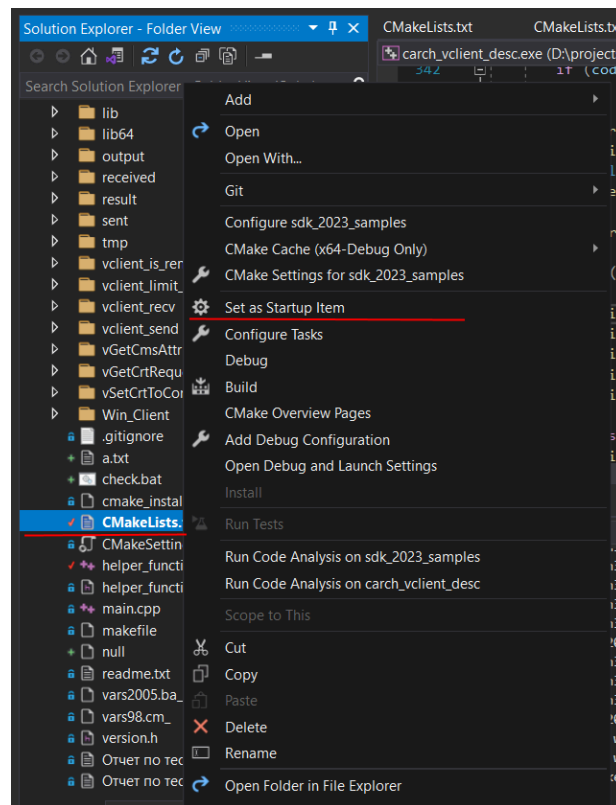


Рисунок 4. Выбор файла CMakeLists.txt

7. Выберите **Build** → **Build all**.

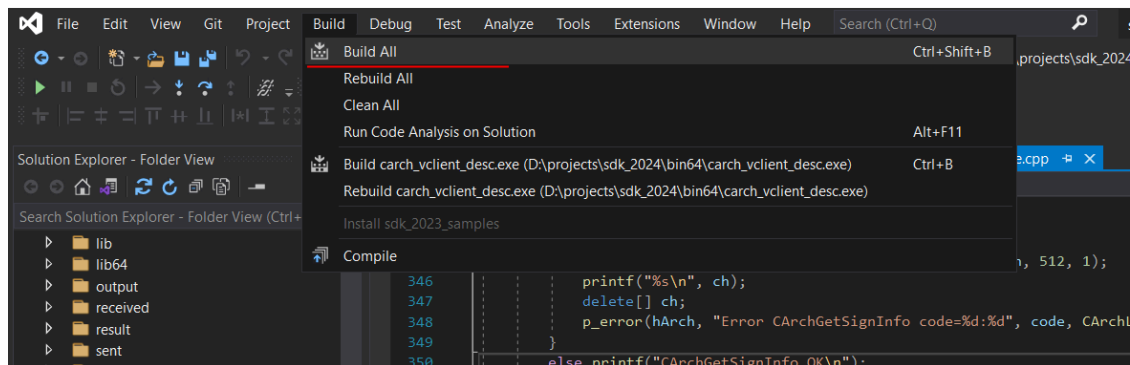


Рисунок 5. Выбор пункта меню Build all



**Примечание:** Проект **sdk\_samples** запускает все примеры последовательно и формирует общий лог.



**Примечание:** Для запуска отдельного примера нажмите в панели инструментов на выпадающий список и выберите проект для компиляции.



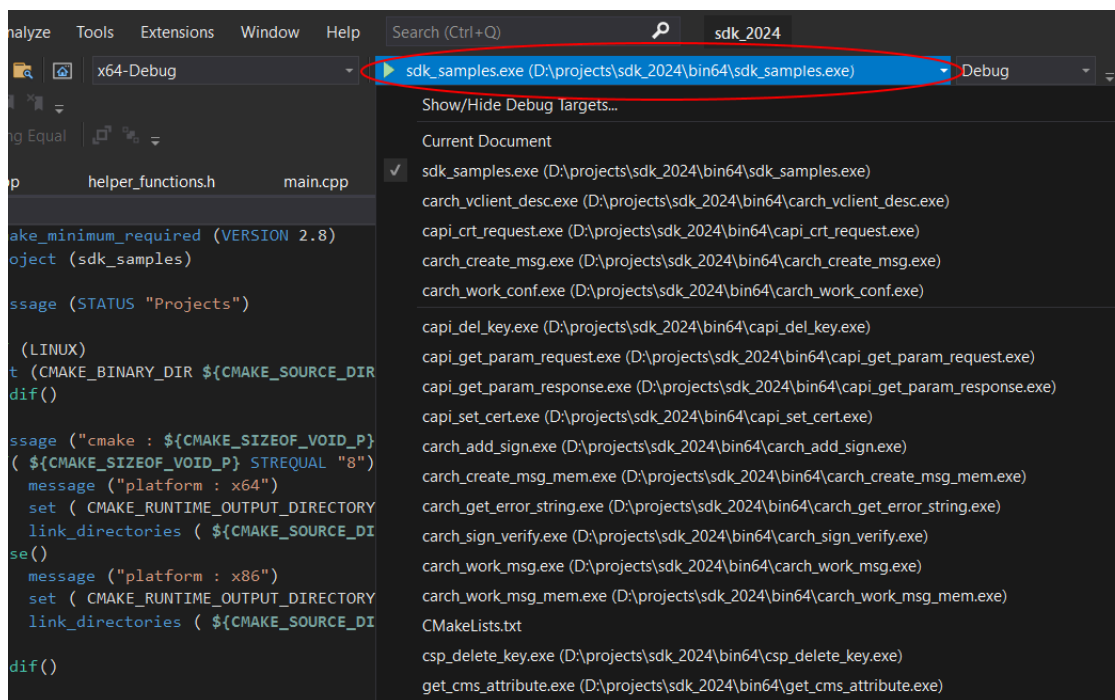


Рисунок 6. Запуска отдельного примера

### 6.3.3 Linux. CLion

Компилирование и сборка примеров в CLion 2022:



**Важное замечание:** Предварительно заполните файл `include\common_params.h` актуальными данными.

1. Скопируйте с дистрибутивного диска на рабочую станцию каталог с примерами по использованию функций API FASTi2 (например, в каталог `/home/user/`).
2. Запустите CLion 2022.
3. Выберите **Открыть проект** → **Open a local folder**.

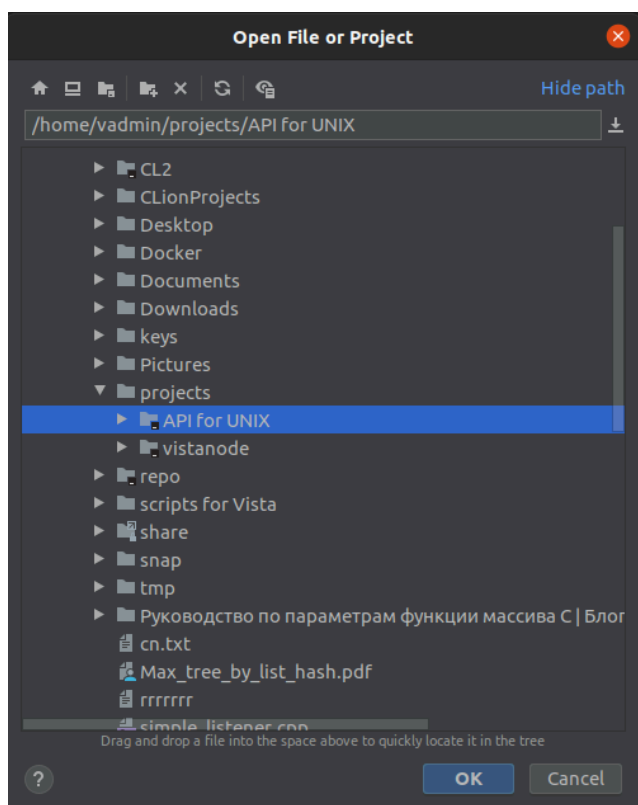


Рисунок 7. Открытие файла проекта

4. Выберите **Build** → **Rebuild all**.

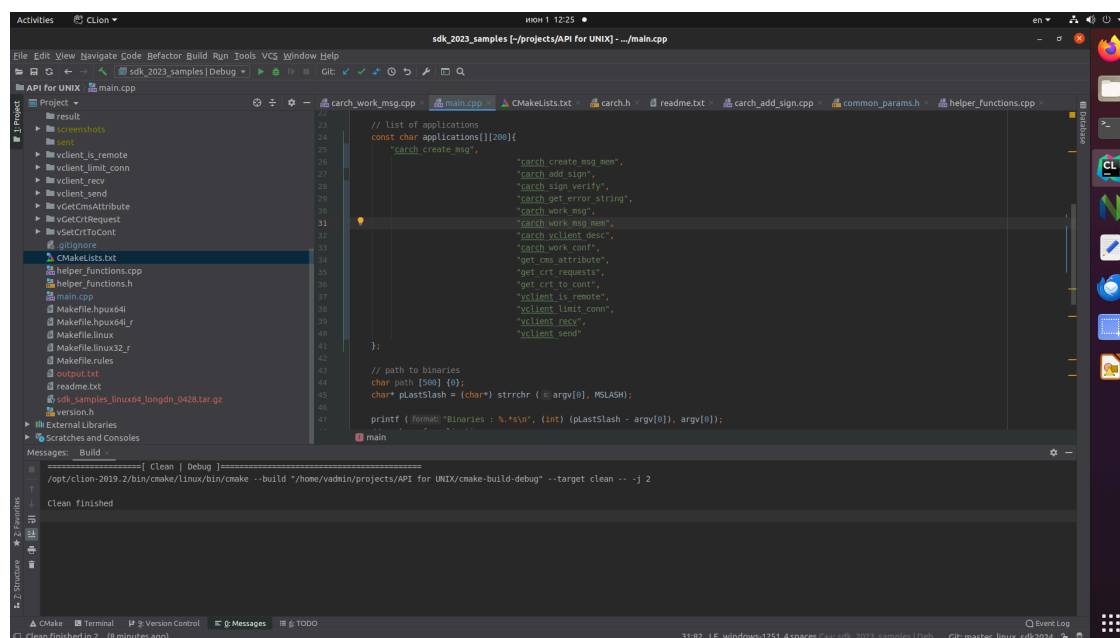


Рисунок 8. Выполнение команды Build

5. В результате успешно собранная программа.

```

Messages: Build x
Scanning dependencies of target get_crt_requests
[ 88%] Building CXX object vGetCrtRequest/CMakeFiles/get_crt_requests.dir/vGetCrtRequest.cpp.o
[ 91%] Linking CXX executable ../../bin64/get_cms_attribute
[ 94%] Linking CXX executable ../../bin64/get_crt_requests
[ 94%] Built target get_cms_attribute
Scanning dependencies of target get_crt_to_cont
[ 97%] Building CXX object vSetCrtToCont/CMakeFiles/get_crt_to_cont.dir/get_crt_to_cont.cpp.o
[ 97%] Built target get_crt_requests
[100%] Linking CXX executable ../../bin64/get_crt_to_cont
[100%] Built target get_crt_to_cont

Build finished

```

Рисунок 9. Результат сборки программы

- Пункт меню **Run** запускает на выполнение.

### 6.3.4 Linux. CMake

Компилирование и сборка примеров с помощью CMake (без IDE):

**! Важное замечание:** Предварительно заполните файл `include\common_params.h` актуальными данными.

- Скопируйте с дистрибутивного диска на рабочую станцию каталог с примерами по использованию функций API FASTi2 (например, в каталог `/home/user/`).
- В консоли перейдите в этот каталог ( в данном случае `~/sdk_samples_2024`).
- Выполните последовательно команды:

```
mkdir cmake-build-debug
```

```
cd cmake-build-debug
```

```
cmake ..
```

```

vadmin@ubuntu:~/projects/API for UNIX/cmake-build-debug$ cmake ..
-- The C compiler identification is GNU 9.4.0
-- The CXX compiler identification is GNU 9.4.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Projects
-- Configuring done
-- Generating done
-- Build files have been written to: /home/vadmin/projects/API for UNIX/cmake-build-debug

```

Рисунок 10. Запуск компиляции

- Выполните команду:

```
cmake -build .
```

```
Scanning dependencies of target carch_work_msg
[ 47%] Building CXX object carch_work_msg/CMakeFiles/carch_work_msg.dir/carch_w
[ 50%] Linking CXX executable ../../bin64/carch_work_msg
[ 50%] Built target carch_work_msg
Scanning dependencies of target carch_work_msg_mem
[ 52%] Building CXX object carch_work_msg_mem/CMakeFiles/carch_work_msg_mem.dir
[ 55%] Linking CXX executable ../../bin64/carch_work_msg_mem
[ 55%] Built target carch_work_msg_mem
Scanning dependencies of target carch_get_error_string
[ 58%] Building CXX object CArchGetErrorString/CMakeFiles/carch_get_error_strin
[ 61%] Linking CXX executable ../../bin64/carch_get_error_string
[ 61%] Built target carch_get_error_string
Scanning dependencies of target vclient_is_remote
[ 63%] Building CXX object vclient_is_remote/CMakeFiles/vclient_is_remote.dir/v
[ 66%] Linking CXX executable ../../bin64/vclient_is_remote
[ 66%] Built target vclient_is_remote
Scanning dependencies of target vclient_limit_conn
[ 69%] Building CXX object vclient_limit_conn/CMakeFiles/vclient_limit_conn.dir
[ 72%] Linking CXX executable ../../bin64/vclient_limit_conn
[ 72%] Built target vclient_limit_conn
Scanning dependencies of target vclient_recv
[ 75%] Building CXX object vclient_recv/CMakeFiles/vclient_recv.dir/vclient_rec
[ 77%] Linking CXX executable ../../bin64/vclient_recv
[ 77%] Built target vclient_recv
Scanning dependencies of target vclient_send
[ 80%] Building CXX object vclient_send/CMakeFiles/vclient_send.dir/vclient_sen
[ 83%] Linking CXX executable ../../bin64/vclient_send
[ 83%] Built target vclient_send
Scanning dependencies of target get_cms_attribute
[ 86%] Building CXX object vGetCmsAttribute/CMakeFiles/get_cms_attribute.dir/ge
[ 88%] Linking CXX executable ../../bin64/get_cms_attribute
[ 88%] Built target get_cms_attribute
Scanning dependencies of target get_crt_requests
[ 91%] Building CXX object vGetCrtRequest/CMakeFiles/get_crt_requests.dir/vGetC
```

Рисунок 11. Выполнение команды Build

## 6.4 Порядок использования примеров

Общий порядок использования примеров полностью повторяет алгоритм работы по созданию, сжатию, зашифровыванию, отправке, приему, расшифровыванию сообщений и других функций.

Примеры, расположенные в папках, название которых начинаются на **carch**, демонстрируют возможности использования работы с сообщениями (архивами).

Примеры, расположенные в папках, название которых начинаются на **vclient**, демонстрируют возможности использования работы с сервером транспортного узла.

Пример [vGetCrtRequest](#) демонстрирует возможность формирования запроса на выпуск сертификатов и генерации пары ключей пользователя с использованием функций библиотеки libvista.

Пример [vGetCmsAttribute](#) демонстрирует возможность получения атрибутов PKCS#7 сообщения, полученного с сервера с использованием функций библиотеки libvista.

Пример [vSetCrtToCont](#) демонстрирует возможность импорта сертификатов в ключевой контейнер пользователя с использованием функций библиотеки libvista.

Примеры, расположенные в папках, название которых начинаются на CAPI, демонстрируют осуществление взаимодействия клиентского программного обеспечения с Удостоверяющим центром с использованием функций [CryptoAPI](#).

Пример [cspDelKey](#) демонстрирует метод удаления ключей из ключевого контейнера, имеющих статус **Новый** с использованием функций библиотеки cptumar.dll.

### Цели компиляции примеров:

- для примера `carch_create_msg` цель компиляции `create_msg`;
- для примера `carch_create_msg_mem` цель компиляции `create_msg_mem`;
- для примера `carch_add_sign` цель компиляции `add_sign`;
- для примера `carch_sign_verify` цель компиляции `sign_verify`;

- для примера `vclient_is_remote` цель компиляции `is_remote`;
- для примера `vclient_limit_conn` цель компиляции `limit_conn`;
- для примера `carch_vclient_desc` цель компиляции `vclient_desc`;
- для примера `CArchGetErrorString` цель компиляции `get_error_str`;
- для примера `vclient_send` цель компиляции `send`;
- для примера `vclient_recv` цель компиляции `recv`;
- для примера `carch_work_msg` цель компиляции `work_msg`;
- для примера `carch_work_msg_mem` цель компиляции `work_msg_mem`;
- для примера `carch_work_conf` цель компиляции `work_conf`;
- для примера `CAPICrtRequest` цель компиляции `cap_i_certrequest`;
- для примера `CAPIGetParamReq` цель компиляции `cap_i_getparamreq`;
- для примера `CAPIGetParamResp` цель компиляции `cap_i_getparamresp`;
- для примера `CAPISetCert` цель компиляции `cap_i_setcert`;
- для примера `CAPIDelKey` цель компиляции `cap_i_delkey`;
- для примера `cspDelKey` цель компиляции `csp_del_key`;
- для примера `vGetCrtRequest` цель компиляции `get_crt_request`;
- для примера `vGetCmsAttribute` цель компиляции `get_cms_attribute`;
- для примера `vSetCrtToCont` цель компиляции `set_to_cont`.



**Примечание:** Для проверки работы системы под управлением ОС семейства Windows рекомендуется использовать примеры, реализованные с помощью [функций CryptoAPI](#), для других ОС – примеры, реализованные с помощью [функций библиотеки libvista](#).



**Примечание:** Проект `sdk_samples` запускает все примеры последовательно и формирует общий лог.

### 6.4.1 Создание зашифрованного и подписанного ЭЦП сообщения

Пример создания зашифрованного и подписанного ЭЦП сообщения для FASTi2 (

`carch_create_msg`).

Файлы-вложения для упаковки в сообщение находятся в каталоге `\data`.

Полученное сообщение может быть использовано в других примерах работы с FASTi2 (работа с архивом, добавление подписи, импорт сообщения, отправка сообщения и др.).

#### Назначение примера

- Создание тестового архива (с атрибутом «Требуется подтверждение» или без него), комментария;
- Установка параметров работы с хранилищем сертификатов и модулем криптографии;
- Чтение содержимого каталога;
- Регистрация функции визуализации;
- Вычисление референса архива;
- Установка информации о получателе;
- Добавление файлов в архив;
- Закрытие дескриптора архива.

#### Компиляция и выполнение примера

1. Перед компиляцией примера убедитесь, что файл `\include\common_params.h` корректно настроен. В данном случае важно указать:<sup>5</sup>

```
CSP_PROFILE_USER1
MY_DN_USER1
```

<sup>5</sup> В примере формируется зашифрованное сообщение от USER1 к USER2

RCPT2

- В результате в каталоге \API for Win\sent\_folder должен быть создан \*.cms файл. Например: 39EFWJP4P9DCC6FA4PHYANN90CHW95DT.cms.

```

Select Microsoft Visual Studio Debug Console
CArchCreate - Ok
CArchSetStore - Ok
lic check license C:\Program Files (x86)\GammaTech\TumarCSP\lib64\bcctoken.plg
lic check license C:\Program Files (x86)\GammaTech\TumarCSP\lib64\jtoken.plg
lic check license C:\Program Files (x86)\GammaTech\TumarCSP\lib64\kztoken.plg
lic check license C:\Program Files (x86)\GammaTech\TumarCSP\lib64\newoids.plg
CArchInitCrypt - Ok
CArchListDir - Ok
CArchSetRecipients - Ok

[      7/      7/ 0%] file1.txt
[      7/      7/ 0%] file2.txt
Referens: 39EFWJP4P9DCC6FA4PHYANN90CHW95DT
CArchClose - Ok

FINAL OK

D:\projects\sdk_2024\bin64\carch_create_msg.exe (process 16004) exited with code 0.
Press any key to close this window . . .

```

Рисунок 12. Пример создания зашифрованного и подписанного ЭЦП сообщения

## 6.4.2 Создание зашифрованного и подписанного ЭЦП сообщения через оперативную память

Пример создания зашифрованного и подписанного ЭЦП сообщения через оперативную память для FASTi2 (carch\_create\_msg\_mem).

Файлы-вложения для сообщения передаются через оперативную память.

Полученное сообщение может быть использовано в других примерах работы с FASTi2 (работа с архивом, добавление подписи, импорт сообщения, отправка сообщения и др.).

### Назначение примера

Пример демонстрирует методы:

- Создания тестового архива (с атрибутом «Требуется подтверждение» или без него), комментария;
- Установки параметров работы с хранилищем сертификатов и модулем криптографии;
- Чтения содержимого каталога;
- Регистрации функции визуализации;
- Вычисления референса архива;
- Установки информации о получателях;
- Добавления файлов в архив;
- Закрытия дескриптора архива.

### Компиляция и выполнение примера

- Для запуска из MSVC (с CMake) необходимо выбрать проект carch\_create\_mem\_msg.
- Перед компиляцией примера убедитесь, что файл \include\common\_params.h корректно настроен. В данном случае важно указать:<sup>6</sup>

```

CSP_PROFILE_USER1="USER1"
MY_DN_USER1="c=KZ;o=BUS7200;cn=TEST.USER1"

```

<sup>6</sup> Профили должны быть предварительно созданы в ПО «ТУМАР-CSP»

```
RCPT2 ="RCPT2.VS11.VS11.KZ"
```

- В результате в каталоге \API for Win\sent\_folder должен быть создан \*.cms файл. Например: XCFCJKFH60013UD4CPYCXGD1AXHU0444.cms

```
C:\SDK\API for Win\bin>create_msg_mem.exe
[ 63/ 63/ 0%] test_file.txt
Referens: XCFCJKFH60013UD4CPYCXGD1AXHU0444
FINAL OK
```

Рисунок 13. Пример создания зашифрованного и подписанного ЭЦП сообщения через оперативную память

### 6.4.3 Добавление ЭЦП в сообщение

Пример добавления ЭЦП в сообщение FASTi2 (carch\_add\_sign).

#### Назначение примера

Пример демонстрирует методы:

- Открытия тестового архива;
- Установки параметров работы с хранилищем сертификатов и модулем криптографии;
- Добавления ЭЦП в сообщение;
- Закрытия дескриптора архива.

#### Компиляция и выполнение примера

- Перед компиляцией примера убедитесь, что файл \include\common\_params.h корректно настроен. В данном случае важно указать:

```
CSP_PROFILE_USER2="USER2"
MY_DN_USER2="c=KZ;o=BUS7200;cn=TEST.USER2"
TEST_FILE="..\data\VJKLSKD392032.CMS"
```



**Примечание:** Так как происходит редактирование уже созданного CMS-файла, необходимо указать путь+имя этого CMS-файла, созданного в примерах [carch\\_create\\_msg](#) или [carch\\_create\\_msg\\_mem](#) сообщения. Имя можно задать либо как параметр командной строки (carch\_add\_sign ../data/XXXXXX.cms), либо прописав в файле common\_params.h переменную `CMS_FILE`

Например:

```
const char CMS_FILE[500] {"..\data\XXXXXX.cms"};
```

- Результат: признаком добавления подписи является увеличение размера файла cms сообщения.

```
td:\projects\sdk_2024\bin64>carch_add_sign.exe ..\sent\39EFWJP4P9DCC6FA4PHYAHN90CHW95DT.cms
..\sent\39EFWJP4P9DCC6FA4PHYAHN90CHW95DT.cms
lic check license C:\Program Files (x86)\GammaTech\TumarCSP\lib64\bcctoken.plg
lic check license C:\Program Files (x86)\GammaTech\TumarCSP\lib64\jtoken.plg
lic check license C:\Program Files (x86)\GammaTech\TumarCSP\lib64\kztoken.plg
lic check license C:\Program Files (x86)\GammaTech\TumarCSP\lib64\newoids.plg
CArchAddSign - OK
FINAL OK
```

Рисунок 14. Пример добавления ЭЦП в сообщение



#### 6.4.4 Добавление и проверка ЭЦП в сообщении

Пример добавления и проверки ЭЦП в сообщении FASTi2 (**carch sign verify**).

## Назначение примера

Пример демонстрирует методы:

- Установки параметров работы с хранилищем сертификатов и модулем криптографии;
- Добавления ЭЦП в сообщение с использованием функций **CMSMessageSign** и **CMSMessageAddSign**;
- Проверки ЭЦП в сообщении с использованием функции **CMSMessageVerifySign**.

## Компиляция и выполнение примера

1. Перед компиляцией примера убедитесь, что файл `\include\common_params.h` корректно настроен. В данном случае важно указать:

```
Const char CSP_PROFILE_USER1[500] {"USER2"};
Const char MY_DN_USER1[500] {"c=KZ;o=BUS7200;cn=TEST.USER2"};
Const char MY_STORE[500] {"ldap://192.168.7.203:62222"};
```

”

**Примечание:** Так как происходит редактирование уже созданного CMS-файла, необходимо указать путь+имя этого CMS-файла, созданного в примерах `carch_create_msg` или `carch_create_msg_mem` сообщения. Имя можно задать либо как параметр командной строки (`carch_add_sign ../data/XXXXX.cms`), либо прописав в файле `common_params.h` переменную `CMS_FILE`

Например:

```
const char CMS_FILE[500] {"..\data\XXXXXX.cms"};
```

- ## 2. Результат:

[illegible]

Рисунок 15. Пример добавления и проверки ЭЦП в сообщении

### 6.4.5 Получение локального или удаленного соединения

Пример получения типа соединения - локального или удаленного (**vclient is remote**).

В примере создается подключение к удаленному узлу, корневому серверу FASTi2, и подключение к локальному каталогу с CMS для отправки/получения.

## Компиляция и выполнение примера

1. Перед компиляцией примера убедитесь, что файл `\include\common_params.h` корректно настроен. В данном случае важно указать:

```
CSP_PROFILE_USER1
MY_DN_USER1
SERVER_HOST_ADDRESS
SERVER_PORT
```

- ## 2. Результат:



```

connect to the remote node : VISTA-TESTMARKLEN.NPCK.KZ
vcOpenRemote, Ok
Connected type is remote, Ok

connect to the local path : ../received/
Connected type is local, Ok
vcOpenLocal, Ok
Final

```

Рисунок 16. Пример получения типа соединения - локального или удаленного

### 6.4.6 Проверка ограничения соединений к серверу с одного IP-адреса

Пример проверки ограничения на количество соединений к серверу с одного IP-адреса (`vcclient_limit_conn`).

Для правильной работы примера необходимо, чтобы были заполнены параметры раздела примера «Тестовые параметры» для данного узла.

#### Компиляция и выполнение примера

1. Перед компиляцией примера убедитесь, что файл `\include\common_params.h` корректно настроен. В данном случае важно указать: DN-имя отправителя, профайл с ключами отправителя, адрес удаленного узла, порт и количество соединений:

```

CSP_PROFILE_USER1
MY_DN USER1
SERVER_HOST_ADDRESS
SERVER_PORT
MAX_CONNECTION=10

```

2. Результат:

```

C:\SDK\API for Win\bin>limit_conn.exe
Iteration 0 - vcOpenRemote, Ok
Iteration 1 - vcOpenRemote, Ok
Iteration 2 - vcOpenRemote, Ok
Iteration 3 - vcOpenRemote, Ok
Iteration 4 - vcOpenRemote, Ok
Iteration 5 - vcOpenRemote, Error = 1133
Iteration 6 - vcOpenRemote, Error = 1133
Iteration 7 - vcOpenRemote, Error = 1133
Iteration 8 - vcOpenRemote, Error = 1133
Iteration 9 - vcOpenRemote, Error = 1133

5 connections have been opened from max=10
Connection 0 is closed
Connection 1 is closed
Connection 2 is closed
Connection 3 is closed
Connection 4 is closed
Connection 5 is closed
Connection 6 is closed
Connection 7 is closed
Connection 8 is closed
Connection 9 is closed
Final

C:\SDK\API for Win\bin>_

```

Рисунок 17. Пример проверки ограничения количества соединений к серверу с одного IP-адреса

## 6.4.7 Проверка работоспособности основных функций API

Пример проверки работоспособности основных функций API FASTi2 (`carch_vclient_desc`).

### Назначение примера

Пример демонстрирует возможность проверки контроля дескрипторов функций.

Функцию можно разделить на 3 части:

- Отправка сообщений:
  - соединение к локальному каталогу;
  - соединение к удаленному узлу;
  - отправка сообщений.
- Получение сообщений:
  - соединение к локальному каталогу;
  - подключение к удаленному узлу;
  - получение количества сообщений на удаленном узле;
  - запросы на последовательное получение сообщений с удаленного узла.
- Разбор сообщений – последовательное открытие каждого сообщения и печать информации.

### Компиляция и выполнение примера

1. Перед компиляцией примера убедитесь, что файл `\include\common_params.h` корректно настроен. В данном случае важно указать:

- параметры отправителя:

```
CSP_PROFILE_USER1  
MY_DN_USER1
```

- параметры сервера:

```
SERVER_HOST_ADDRESS  
SERVER_PORT
```

- параметры получателя:

```
CSP_PROFILE_USER2  
MY_DN_USER2
```

- локальный каталог:

```
SENT_FOLDER  
RECEIVED_FOLDER
```

2. Результат:

```

Sign Count=1
-----
SERIALNUMBER=IIN000000000003;UID=BIN001001001013;CN=USER1.VISTA-TESTMARKLEN.NPCK.KZ;OU=BIN001001001013;O=Бұл менің байтақ елім ! Қазақстанда көптеген әдемі қалалар бар;C=KZ
20240517120107Z
1.2.398.3.10.1.3.3
1.2.398.3.10.1.1.2.3.2
SIGN OK
-----
CArchListArchFile...
CArchListArchFile OK
file1.txt
file2.txt
CArchRegCallbaker...
CArchRegCallbaker OK
CArchGetReferens...
CArchGetReferens OK

Referens: 5MBXX53YMEPEYCRU17DWZM7TKXPNKG0K
CArchGetEnvelopInfo...
CArchGetEnvelopInfo OK
Sender: SERIALNUMBER=IIN000000000003;UID=BIN001001001013;CN=USER1.VISTA-TESTMARKLEN.NPCK.KZ;OU=BIN001001001013;O=Бұл менің байтақ елім ! Қазақстанда көптеген әдемі қалалар бар;C=KZ
Recipients(count=1):
SERIALNUMBER=IIN000000000006;UID=BIN001001001016;CN=USER2.VISTA-TESTMARKLEN.NPCK.KZ;OU=BIN001001001016;O=Please agree on proposals for forms in the server monitoring system;C=KZ

```

Рисунок 18. Пример проверки работоспособности основных функций API

### 6.4.8 Получение сообщения об ошибке

Пример получения сообщения об ошибке, используя стандартный и расширенный код ошибки FASTi2 (CArchGetErrorString).

#### Назначение примера

Пример демонстрирует получение ошибки в случае попытки распаковки вложения получателем, для которого не предназначено данное сообщение: «Невозможно получить вложение, предназначенное другому получателю».

Пример демонстрирует методы:

- Открытия тестового архива;
- Установки параметров работы с хранилищем сертификатов и модулем криптографии;
- Чтения комментария к архиву;
- Получения информации об ЭЦП;
- Чтения содержимого архива;
- Регистрации функции визуализации;
- Вычисления референса архива;
- Получения информации об отправителе и получателях;
- Попытки получить вложение. Ошибка: «Невозможно получить вложение, предназначенное другому получателю»;
- Закрытие дескриптора архива.

#### Компиляция и выполнение примера

1. Перед компиляцией примера убедитесь, что файл `\include\common_params.h` корректно настроен. В данном случае важно указать:

```

CSP_PROFILE_USER1
MY_DN_USER1
SENT_FOLDER

```



**Важное замечание:** Для успешного результата необходимо чтобы в каталоге `SENT_FOLDER` были CMS-файлы.

2. Результат:

```

D:\projects\sdk_2024\bin64>carch_get_error_string.exe
vcOpenLocal, OK
vcList...
lic check license C:\Program Files (x86)\GammaamaTech\TumarCSP\lib64\bcctoken.plg
lic check license C:\Program Files (x86)\GammaamaTech\TumarCSP\lib64\itoken.plg
lic check license C:\Program Files (x86)\GammaamaTech\TumarCSP\lib64\ktoken.plg
lic check license C:\Program Files (x86)\GammaamaTech\TumarCSP\lib64\newoids.plg
Comment:Test comment...

Sign Count=2
-----
SERIALNUMBER=IIN0000000000003;UID=BIN001001001013;CN=USER1.VISTA-TESTMARKLEN.NPCK.KZ;OU=BIN001001001013;O=Бұл менің байтақ елім ! Қазақстанда көптеген әдемі
каналар бар;C=KZ
20240517055705Z
1.2.398.3.10.1.3.3
1.2.398.3.10.1.1.2.3.2
SIGN OK
-----
SERIALNUMBER=IIN0000000000006;UID=BIN001001001016;CN=USER2.VISTA-TESTMARKLEN.NPCK.KZ;OU=BIN001001001016;O=Please agree on proposals for forms in the server m
onitoring system;C=KZ
20240517060537Z
1.2.398.3.10.1.3.3
1.2.398.3.10.1.1.2.3.2
SIGN OK
-----
file1.txt
file2.txt

Referens: 39EFWJP4P9DCC6FA4PHYVNH90CINW05DT
Sender: SERIALNUMBER=IIN0000000000003;UID=BIN001001001013;CN=USER1.VISTA-TESTMARKLEN.NPCK.KZ;OU=BIN001001001013;O=Бұл менің байтақ елім ! Қазақстанда көптеген
әдемі каналар бар;C=KZ
Recipients(count=1):
SERIALNUMBER=IIN0000000000006;UID=BIN001001001016;CN=USER2.VISTA-TESTMARKLEN.NPCK.KZ;OU=BIN001001001016;O=Please agree on proposals for forms in the server mon
itoring system;C=KZ

Error of CMS format (ext=44 (Not specified or invalid recipient in loaded message))

```

Рисунок 19. Пример получения сообщения об ошибке

### 6.4.9 Отправка сообщения на сервер транспортного узла

Пример отправки сообщения на сервер транспортного узла FASTi2 (`vcclient_send`).

Исходные сообщения для данного примера можно создать, используя пример создания зашифрованного и подписанного ЭЦП сообщения.

#### Назначение примера

Пример демонстрирует методы:

- Открытия соединения с файловой системой;
- Получение списка сообщений с файловой системы;
- Открытия соединения с сервером транспортного узла;
- Отправка сообщений с файловой системой на сервер транспортного узла;
- Закрытия соединений.

Для правильной работы примера необходимо, чтобы были заполнены параметры раздела примера «Тестовые параметры» для данного узла.

#### Компиляция и выполнение примера

1. Перед компиляцией примера убедитесь, что файл `\include\common_params.h` корректно настроен. В данном случае важно указать:

- параметры отправителя:

```
CSP_PROFILE_USER1
MY_DN_USER1
```

- локальный каталог:

```
SENT_FOLDER
```

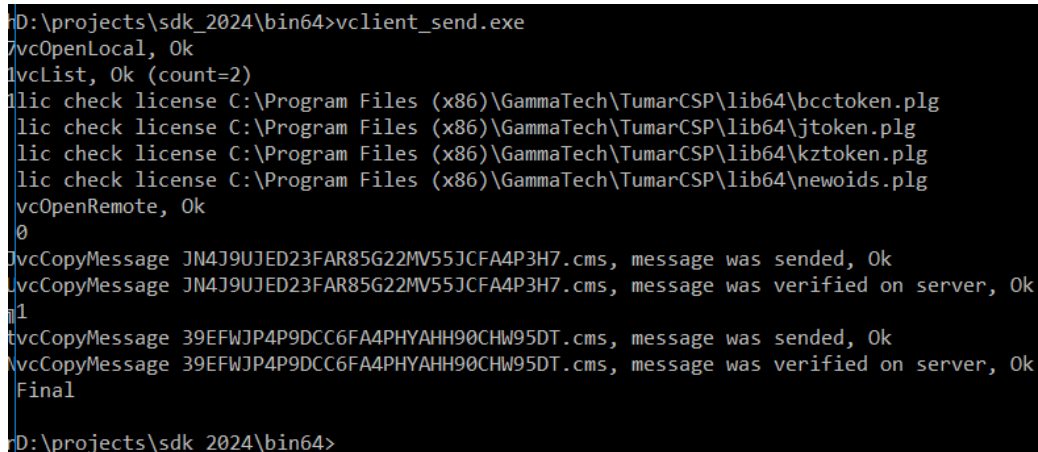
- параметры хранилища сертификатов:

```
LOCAL_STORE_HOST
LOCAL_STORE_PORT
```

- параметры сервера:

```
SERVER_HOST_ADDRESS
SERVER_PORT
```

2. Результат: файлы из каталога SENT\_FOLDER будут отправлены на сервер и удалены из данного каталога.



```

D:\projects\sdk_2024\bin64>vclient_send.exe
7vcOpenLocal, Ok
7vcList, Ok (count=2)
lic check license C:\Program Files (x86)\GammaTech\TumarCSP\lib64\bcctoken.plg
lic check license C:\Program Files (x86)\GammaTech\TumarCSP\lib64\jtoken.plg
lic check license C:\Program Files (x86)\GammaTech\TumarCSP\lib64\kztoken.plg
lic check license C:\Program Files (x86)\GammaTech\TumarCSP\lib64\newoids.plg
7vcOpenRemote, Ok
0
7vcCopyMessage JN4J9UJED23FAR85G22MV55JCFA4P3H7.cms, message was sent, Ok
7vcCopyMessage JN4J9UJED23FAR85G22MV55JCFA4P3H7.cms, message was verified on server, Ok
1
7vcCopyMessage 39EFWJP4P9DCC6FA4PHYANH90CHW95DT.cms, message was sent, Ok
7vcCopyMessage 39EFWJP4P9DCC6FA4PHYANH90CHW95DT.cms, message was verified on server, Ok
Final
D:\projects\sdk_2024\bin64>

```

Рисунок 20. Пример отправки сообщения на сервер транспортного узла

#### 6.4.10 Получение сообщения с сервера транспортного узла

Пример получения сообщения с сервера транспортного узла FASTi2 (vclient\_recv).

##### Назначение примера

Пример демонстрирует методы:

- Открытия соединения с сервером транспортного узла;
- Получения списка сообщений с сервера транспортного узла;
- Открытия соединения с файловой системой;
- Получения сообщений с транспортного узла на файловую систему;
- Закрытия соединений.

Для правильной работы примера необходимо, чтобы были заполнены параметры раздела примера «Тестовые параметры» для данного узла.

В результате:

- будет принято сообщение с сервера;
- принятое сообщение будет записано в каталог ..\API for Win\bin.

##### Компиляция и выполнение примера

1. Перед компиляцией примера убедитесь, что файл \include\common\_params.h корректно настроен. В данном случае важно указать:

- параметры получателя:

```
CSP_PROFILE_USER2
MY_DN_USER2
```

- локальный каталог:

```
RECEIVED_FOLDER
```

- параметры хранилища сертификатов:

```
LOCAL_STORE_HOST
```

```
LOCAL_STORE_PORT
```

- параметры сервера:

```
SERVER_HOST_ADDRESS
SERVER_PORT
```

2. Результат: в каталоге RECEIVED\_FOLDER появятся сообщения (\*.CMS).

```
D:\projects\sdk_2024\bin64>vclient_recv.exe
lic check license C:\Program Files (x86)\GammaTech\TumarCSP\lib64\bcctoken.plg
lic check license C:\Program Files (x86)\GammaTech\TumarCSP\lib64\jtoken.plg
lic check license C:\Program Files (x86)\GammaTech\TumarCSP\lib64\kztoken.plg
lic check license C:\Program Files (x86)\GammaTech\TumarCSP\lib64\newoids.plg
vcOpenRemote, Ok
vcList, Ok (count=2)
vcOpenLocal, Ok
vcCopyMessage 5SWX73CAW51CH1C6JRS6J2HP6MX07VMC.cms, message was received, Ok
vcCopyMessage 5SWX73CAW51CH1C6JRS6J2HP6MX07VMC.cms, message was verified, Ok
vcCopyMessage 5SWX73CAW51CH1C6JRS6J2HP6MX07VMC.cms, size of sended confirmation = 922
vcCopyMessage 5SWX73CAW51CH1C6JRS6J2HP6MX07VMC.cms, conf_mess = 5SWX73CAW51CH1C6JRS6J2HP6MX07VMC.cms
OK
vcCopyMessage JN4J9UJED23FAR85G22MV55JCF4P3H7.cms, message was received, Ok
vcCopyMessage JN4J9UJED23FAR85G22MV55JCF4P3H7.cms, message was verified, Ok
vcCopyMessage JN4J9UJED23FAR85G22MV55JCF4P3H7.cms, size of sended confirmation = 922
vcCopyMessage JN4J9UJED23FAR85G22MV55JCF4P3H7.cms, conf_mess = JN4J9UJED23FAR85G22MV55JCF4P3H7.cms
OK
Final
```

Рисунок 21. Пример получения сообщения с сервера транспортного узла

#### 6.4.11 Работа с сообщением (carch\_work\_msg)

Пример работы с сообщением FASTi2 (carch\_work\_msg). При извлечении файлов из архива вложения сообщения будут записаны в каталоге ..\tmp.

##### Назначение примера

Пример демонстрирует методы:

- Открытия тестового архива;
- Установка параметров работы с хранилищем сертификатов и модулем криптографии;
- Чтения комментария к архиву;
- Получения информации об ЭЦП;
- Чтения содержимого архива;
- Регистрации функции визуализации;
- Вычисления референса архива;
- Получения информации об отправителе и получателях;
- Извлечения файлов из архива;
- Закрытия дескриптора архива.

##### Компиляция и выполнение примера

1. Перед компиляцией примера убедитесь, что файл \include\common\_params.h корректно настроен. В данном случае важно указать:

```
CSP_PROFILE_USER2
MY_DN_USER2
RECEIVED_FOLDER
```

2. Результат:

```

D:\projects\sdk_2024\bin64>carch_work_msg.exe
avcOpenLocal, Ok
vclist...
Processing the file : ../received/S7297XU2Y2CBFAK0FWEAFFT2P2E90PRH.cms
lic check license C:\Program Files (x86)\GammaTech\TumarCSP\lib64\bctoken.plg
lic check license C:\Program Files (x86)\GammaTech\TumarCSP\lib64\jtoken.plg
lic check license C:\Program Files (x86)\GammaTech\TumarCSP\lib64\kztoken.plg
lic check license C:\Program Files (x86)\GammaTech\TumarCSP\lib64\newoids.plg
Comment:Test comment...

Sign Count=1
-----
SERIALNUMBER=IIN000000000003;UID=BIN001001001013;CN=USER1.VISTA-TESTMARKLEN.NPCK.KZ;OU=BIN001001001013;O=Бұл менің байтақ елім ! Қазақстанда көптеген әдемі
қалалар бар;C=KZ
20240516064244Z
1.2.398.3.10.1.3.3
1.2.398.3.10.1.1.2.3.2
SIGN OK
-----
file1.txt
file2.txt

Referens: S7297XU2Y2CBFAK0FWEAFFT2P2E90PRH
Sender: SERIALNUMBER=IIN000000000003;UID=BIN001001001013;CN=USER1.VISTA-TESTMARKLEN.NPCK.KZ;OU=BIN001001001013;O=Бұл менің байтақ елім ! Қазақстанда көптеген
әдемі қалалар бар;C=KZ
Recipients(count=1):
SERIALNUMBER=IIN000000000006;UID=BIN001001001016;CN=USER2.VISTA-TESTMARKLEN.NPCK.KZ;OU=BIN001001001016;O=Please agree on proposals for forms in the server mon
itoring system;C=KZ

[      7/      7/ 0%] file1.txt
[      7/      7/ 0%] file2.txt
FINAL OK

```

Рисунок 22. Пример работы с сообщением (carch\_work\_msg)

### 6.4.12 Работа с сообщением (work\_msg\_mem)

Пример работы с сообщением FASTi2 (work\_msg\_mem).

Исходный файл для данного примера можно создать, используя пример создания зашифрованного и подписанного ЭЦП сообщения.

При извлечении файлов из архива, их содержимое будет выводиться на экран в консоли.

#### Назначение примера

Пример демонстрирует методы:

- Открытия тестового архива;
- Установки параметров работы с хранилищем сертификатов и модулем криптографии;
- Чтения комментария к архиву;
- Получения информации об ЭЦП;
- Чтения содержимого архива;
- Регистрации функций визуализации;
- Вычисления референса архива;
- Получения информации об отправителе и получателях;
- Извлечения файлов из архива через оперативную память;
- Закрытия дескриптора архива.

#### Компиляция и выполнение примера

1. Перед компиляцией примера убедитесь, что файл `\include\common_params.h` корректно настроен. В данном случае важно указать:

```

CSP_PROFILE_USER2
MY_DN_USER2
RECEIVED_FOLDER

```

2. Результат: в каталоге `TMP_FOLDER` появятся файлы из сообщения.

```

D:\projects\sdk_2024\bin64>carch_work_msg_mem.exe
vcOpenLocal, Ok
vcList...
Processing a file : ../received/S7297XU2Y2CBFAK0FEAFFT2P2E90PRH.cms...
lic check license C:\Program Files (x86)\GammamaTech\TumarCSP\lib64\bcctoken.plg
lic check license C:\Program Files (x86)\GammamaTech\TumarCSP\lib64\jtoken.plg
lic check license C:\Program Files (x86)\GammamaTech\TumarCSP\lib64\kztoken.plg
lic check license C:\Program Files (x86)\GammamaTech\TumarCSP\lib64\newoids.plg
Comment:Test comment...

Sign Count=1
-----
SERIALNUMBER=IIN0000000000003;UID=BIN001001001013;CN=USER1.VISTA-TESTMARKLEN.NPCK.KZ;OU=BIN001001001013;O=Бұл менің байтақ елім ! Қазақстанда көптеген әдемі
қалалар бар;C=KZ
20240516064244Z
1.2.398.3.10.1.3.3
1.2.398.3.10.1.1.2.3.2
SIGN OK
-----
file1.txt
file2.txt

Referens: S7297XU2Y2CBFAK0FEAFFT2P2E90PRH
Sender: SERIALNUMBER=IIN0000000000003;UID=BIN001001001013;CN=USER1.VISTA-TESTMARKLEN.NPCK.KZ;OU=BIN001001001013;O=Бұл менің байтақ елім ! Қазақстанда көптеген
әдемі қалалар бар;C=KZ
Recipients(count=1):
SERIALNUMBER=IIN0000000000006;UID=BIN001001001016;CN=USER2.VISTA-TESTMARKLEN.NPCK.KZ;OU=BIN001001001016;O=Please agree on proposals for forms in the server mon
itoring system;C=KZ
13:28 7/ 7/ 0%] file1.txt
-----
13:28 7/ 7/ 0%] file2.txt
-----
FINAL OK

```

Рисунок 23. Пример работы с сообщением (work\_msg\_mem)

### 6.4.13 Извлечение информации из сообщения

Пример извлечения информации из сообщения подтверждения о доставке сообщения FASTi2 (carch\_work\_conf).

#### Назначение примера

Пример демонстрирует методы:

- Открытия тестового архива;
- Установки параметров работы с хранилищем сертификатов и модулем криптографии;
- Чтения содержимого;
- Получения информации о сообщении;
- Закрытия дескриптора архива.

#### Компиляция и выполнение примера

1. Перед компиляцией примера убедитесь, что файл \include\common\_params.h корректно настроен. В данном случае важно указать:

```

CSP_PROFILE_USER2
MY_DN_USER2
RECEIVED_FOLDER

```

2. Результат: при извлечении файлов из архива вложения сообщения будут записаны в каталоге TMP\_FOLDER.

### 6.4.14 Формирование запроса на смену сертификата

Пример формирования запроса на смену сертификата, используя функции CryptoAPI (CAPI\_CrtRequest).

Для правильной работы примера необходимо, чтобы были заполнены параметры раздела примера «Тестовые параметры» для данного узла.

Дополнительно необходимо, чтобы были настроены ключи с импортированным сертификатом для подписи запроса.

Параметры, передаваемые в функцию CreatePKCS10 (генерация ключей и формирование PKCS10):

key\_type – ключ подписи/ключевой обмен;



sizeKey – размер ключа.

- Генерация ключа ГОСТ на подпись:

```
key_type = AT_SIGNATURE
sizeKey = 512
```

- Генерация ключа ГОСТ на ключевой обмен:

```
key_type = AT_KEYEXCHANGE
sizeKey = 512
```

- Генерация ключа RSA 1024:

```
sizeKey = 1024
```

- Генерация ключа RSA 1536:

```
sizeKey = 1536
```

- Генерация ключа RSA 2048:

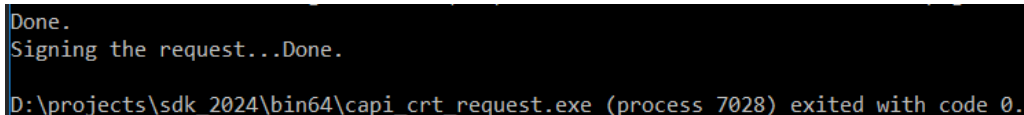
```
sizeKey = 2048
```

### Компиляция и выполнение примера

1. Перед компиляцией примера убедитесь, что файл `\include\common_params.h` корректно настроен. В данном случае важно указать:

```
CSP_PROFILE_USER1
MY_DN_USER1
MY_TEMPLATE="c=KZ;o=Template;cn=Default;"
```

2. Результат: запрос будет сохранен в файл `REQUEST` в каталоге `OUTPUT_FOLDER`.



```
Done.
Signing the request...Done.
D:\projects\sdk_2024\bin64\capi crt_request.exe (process 7028) exited with code 0.
```

Рисунок 24. Пример формирования запроса на смену сертификата

## 6.4.15 Получение параметров PKCS#7-запроса, используя функции CryptoAPI

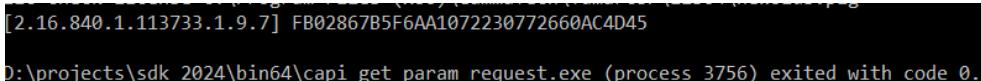
Пример получения параметров PKCS#7-запроса (`CAPI_GetParamReq`), используя функции CryptoAPI. Исходным файлом для данного примера является сформированный запрос `REQUEST` в каталоге `OUTPUT_FOLDER`.

### Компиляция и выполнение примера

1. Перед компиляцией примера убедитесь, что файл `\include\common_params.h` корректно настроен. В данном случае важно указать:

```
CSP_PROFILE_USER1
MY_DN_USER1
```

2. Результат:



```
[2.16.840.1.113733.1.9.7] FB02867B5F6AA1072230772660AC4D45
D:\projects\sdk_2024\bin64\capi get_param_request.exe (process 3756) exited with code 0.
```

Рисунок 25. Пример получения параметров PKCS#7-запроса

## 6.4.16 Получение параметров PKCS#7-ответа, используя функции CryptoAPI

Пример получения параметров PKCS#7-ответа от сервера, используя функции CryptoAPI (CAPI\_GetParamResp). Исходным файлом для данного примера является сформированный ответ от сервера \API for Win\bin\response.

### Компиляция и выполнение примера

1. Перед компиляцией примера убедитесь, что файл \include\common\_params.h корректно настроен. В данном случае важно указать:

```
CSP_PROFILE_USER1
```

2. В каталоге \API for Win\CAPI\_GetParamResp находится три файла, которые необходимо поочередно копировать в каталог \API for Win\bin и назвать скопированный файл response.
3. Результат:

```
C:\SDK\API for Win\bin>capi_get_param_resp.exe  
[2.16.840.1.113733.1.9.3] 0  
[2.16.840.1.113733.1.9.7] E9878C8D136D87D48E4AA39B54D05536
```

Рисунок 26. Пример получения параметров PKCS#7-ответа от сервера. Результат 0 – запрос успешно обработан

2 – запрос отклонен

```
C:\SDK\API for Win\bin>capi_get_param_resp.exe  
[2.16.840.1.113733.1.9.3] 2  
[2.16.840.1.113733.1.9.4] 2  
[2.16.840.1.113733.1.9.7] 9282E1D8C1BC2DC7FFCF19C02E03291B
```

Рисунок 27. Пример получения параметров PKCS#7-ответа от сервера. Результат 2 – запрос отклонен

3 – запрос в обработке

```
C:\SDK\API for Win\bin>capi_get_param_resp.exe  
[2.16.840.1.113733.1.9.3] 3  
[2.16.840.1.113733.1.9.7] C0471E892F1E50F555C13FE4CA5E6101
```

Рисунок 28. Пример получения параметров PKCS#7-ответа от сервера. Результат 3 – запрос в обработке

## 6.4.17 Установка сертификата в ключевой контейнер

Пример установки сертификата в ключевой контейнер, используя функции CryptoAPI (CAPI\_SetCert). Исходный файл, содержащий ответ сервера в кодировке DER, для данного примера нужно поместить в \API for Win\bin\response.

**Компиляция и выполнение примера**

1. Перед запуском примера необходимо настроить makefile в каталоге с примером \API for Win \CAPI\_SetCert (в данном случае важно указать):

```
CSP_PROFILE_USER1
```

2. Для демонстрации работы примера необходимо с помощью примера vGetCrtRequest создать 2 запроса на выпуск сертификата key\_ex и key\_sig.
3. Далее выполнить отправку обоих запросов в обработку в УЦ с помощью html-формы отправки запроса \API for Win\vGetCrtRequest\CrtRequest.html (см. [пример формирования запроса на выпуск сертификата](#) ).
4. После отправки запроса на выпуск сертификата в успешном случае на html странице отобразится кнопка Установить сертификат. Необходимо вызвать контекстное меню на свободном участке данной html-формы и выбрать Просмотреть код страницы.
5. Выделить содержимое с ответом от сервера – сертификатом и сохранить его в файл \API for Win \bin\response в DER-кодировке.

```
<HTML><HEAD><TITLE>Центр сертификации CERTEX.</TITLE>
<META http-equiv=Content-Type content="text/html; charset=windows-1251"></HEAD>
<BODY bgcolor=#F6F8F6>
<hr size=3 width="100%" noshade color=#008060>
<b>Запрос на сертификат.</b><br><hr size=3 width="100%" noshade color=#008060><br>
<li>Тип запроса = 1 : requestCertificate (size=1488, count=1) = 0.</li>
<li>Запрос успешно обработан : provType=25, keySpec=2, size=1460.</li>
<OBJECT classid="clsid:127698E4-E730-4E5C-A2b1-21490A70C8A1" codebase="xenroll.dll" id=Enroll > </OBJECT>
<SCRIPT language=VBScript>
Function InstallPKCS7 ()
Enroll.KeySpec = 2
Enroll.ProviderType = 25
Enroll.acceptPKCS7 (repform.Certificate.value)
If 0=Err.Number Then
MsgBox("InstallPKCS7 OK")
InstallPKCS7=0
Else
MsgBox("InstallPKCS7 error = " & Err.Number)
InstallPKCS7 = Err.Number
End If
End Function
</SCRIPT>
<FORM id=repform name=repform method=post onsubmit=InstallPKCS7()>
<INPUT type=hidden value="MIIEQwYJKoZIhvcNAQcCoIIENDCCBDACAQExEDAOBggrBgEEAbURAQIBBQAwNwYJKoZIhvcNAQcBoCoEKI
```

Рисунок 29. Пример установки сертификата в ключевой контейнер

6. Результат: в ключевой контейнер должен быть установлен сертификат.

### 6.4.18 Удаление ключей из ключевого контейнера, используя функции CryptoAPI

Пример удаления ключей из ключевого контейнера, имеющих статус **Новый**, используя функции CryptoAPI (CAPI\_DelKey).

**Компиляция и выполнение примера**

1. Перед компиляцией примера убедитесь, что файл \include\common\_params.h корректно настроен. В данном случае важно указать:

```
CSP_PROFILE_USER1
```

2. Результат:

```
key_count = 4
deleting keys...
state : ACTIVE oid : 1.2.398.3.10.1.1.2.2.1 - skip
state : ACTIVE oid : 1.2.398.3.10.1.4.1.2.1 - skip
state : WAIT oid : 1.3.6.1.4.1.6801.1.5.8 - deleting
state : WAIT oid : 1.3.6.1.4.1.6801.1.8.8 - deleting

D:\projects\sdk_2024\bin64\capi_del_key.exe (process 7848) exited with code 0.
```

Рисунок 30. Пример удаления ключей из ключевого контейнера

Все ключи со статусом **Новый** по указанному профайлу будут удалены.

#### 6.4.19 Удаление ключей из ключевого контейнера с помощью библиотеки

Пример удаления ключей из ключевого контейнера, имеющих статус Новый с помощью библиотеки `cptumar.dll` (`csp_delete_key`). Для правильной работы примера необходимо указать путь к библиотеке `cptumar.dll` в коде программы.

##### Компиляция и выполнение примера

1. Перед компиляцией примера убедитесь, что файл `\include\common_params.h` корректно настроен. В данном случае важно указать:

```
CSP_PROFILE_USER1
```

2. Дополнительно средствами конфигулятора необходимо создать новые ключи (для их последующего удаления).
3. Результат: все ключи со статусом **Новый** по указанному профайлу будут удалены.

```
key_count = 4
Deleting STATE_WAIT keys...
state : ACTIVE oid : 1.2.398.3.10.1.1.2.2.1 - Skip
state : ACTIVE oid : 1.2.398.3.10.1.4.1.2.1 - Skip
state : WAIT oid : 1.3.6.1.4.1.6801.1.5.8 - deleting
state : WAIT oid : 1.3.6.1.4.1.6801.1.8.8 - deleting

D:\projects\sdk_2024\bin64\csp_delete_key.exe (process 5060) exited with code 0.
```

Рисунок 31. Пример удаления ключей из ключевого контейнера с помощью библиотеки

#### 6.4.20 Формирование запроса на выпуск сертификата

Пример формирования запроса на выпуск сертификата, используя библиотеку `libvista` (`vGetCrtRequest`). Пример демонстрирует метод формирования запроса на выпуск сертификата, используя библиотеку `libvista.dll`. Запросы сохраняются в файлах `\API for Win\bin\key_ex` и `\API for Win\bin\key_sig`.

##### Компиляция и выполнение примера

1. Перед компиляцией примера убедитесь, что файл `\include\common_params.h` корректно настроен. В данном случае важно указать:

```
CSP_PROFILE_USER1
MY_DN_USER1
TMP_FOLDER
```

2. Результат: в каталоге `TMP_FOLDER` будут записаны два файла – запроса на выпуск сертификата: `key_ex`, `key_sig`.

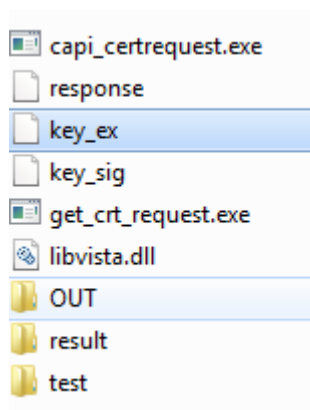


Рисунок 32. Результат примера формирования запроса на выпуск сертификата

#### 6.4.21 Получение параметров PKCS#7-ответа, используя библиотеку libvista

Пример получения параметров PKCS#7-ответа, полученного с сервера, используя библиотеку libvista (**vGetCmsAttribut**). Исходный файл, содержащий PKCS#7-ответ от сервера для данного примера нужно сохранить в \API for Win\bin\response.

##### Компиляция и выполнение примера

1. Перед компиляцией примера убедитесь, что файл \include\common\_params.h корректно настроен. В данном случае важно указать:

```
CSP_PROFILE_USER1  
MY_DN_USER1
```

2. Результат: в результате будет отображен в консоли номер транзакции и статус.

#### 6.4.22 Установка сертификата в ключевой контейнер пользователя

Пример установки сертификата в ключевой контейнер пользователя, используя библиотеку libvista (**vSetCrtToCont**). Исходные файлы, содержащие ответ сервера – файлы с сертификатами в кодировке DER, для данного примера нужно поместить в каталог ..\bin\key\_ex.req и ..\bin\key\_sig.req.

##### Компиляция и выполнение примера

1. Перед компиляцией примера убедитесь, что файл \include\common\_params.h корректно настроен. В данном случае важно указать:

```
CSP_PROFILE_USER1  
MY_USER_DN1
```

2. Результат: в ключевой контейнер должен быть установлен сертификат.

## Приложение 1 Коды ошибок

### 1. Базовые коды ошибок

Ошибки при работе с сообщениями (архивами)

Код	Возвращаемое значение	Описание
1	ERR_ARCH_HANDLE	Неверный дескриптор
2	ERR_ARCH_PARAM	Ошибка в параметрах
3	ERR_ARCH_TYPE	Ошибочный тип сообщения
4	ERR_ARCH_CREATE	Ошибка создания сообщения
5	ERR_ARCH_EXPORTKEY	Ошибка экспорта ключа
6	ERR_ARCH_LIST	Ошибка создания списка файлов
7	ERR_ARCH_ADDFILE	Ошибка добавления файла в сообщение
8	ERR_ARCH_OPENFILE	Ошибка открытия сообщения
9	ERR_ARCH_LISTFILE	Ошибка создания списка файлов в сообщении
10	ERR_ARCH_EXTRFILE	Ошибка извлечения файла из сообщения
11	ERR_ARCH_HASH	Ошибка целостности сообщения
12	ERR_ARCH_CMS	Ошибочный формат CMS
13	ERR_ARCH_CSP	Ошибка при работе с криптопровайдером
14	ERR_ARCH_LSTORE	Ошибка при работе с хранилищем сертификатов
15	ERR_ARCH_FORMAT	Ошибочный формат сообщения
16	ERR_ARCH_GET_CRL	Ошибка получения СОС
17	ERR_ARCH_SET_CA	Ошибка установки сертификата ЦС
18	ERR_ARCH_SET_CRL	Ошибка установки СОС

Ошибки при работе с сервером транспортного узла

Код	Возвращаемое значение	Описание
1000	VCLIENT_ERROR_CONNECT	Не удалось создать соединение
2001	VCLIENT_ERROR_CHANGE_DIR	Не удалось выполнить смену каталога
2002	VCLIENT_ERROR_USER_ABORT	Операция терминирована пользователем
2003	VCLIENT_ERROR_LIST	Не удалось получить список сообщений
2004	VCLIENT_ERROR_MESS_COPY	Ошибка при приёме/передаче сообщения
2005	VCLIENT_ERROR_MESS_OPEN	Ошибка при открытии сообщения
2006	VCLIENT_ERROR_MESS_READ	Ошибка при чтении сообщения
2007	VCLIENT_ERROR_MESS_CLOSE	Ошибка при закрытии сообщения
2008	VCLIENT_ERROR_MESS_HEADER	Ошибка при получении заголовка сообщения
2009	VCLIENT_ERROR_MESS_CONFIRM_COPY	Ошибка при копировании подтверждения на сообщение
2010	VCLIENT_ERROR_MESS_CONFIRM_OPEN	Ошибка при открытии подтверждения на сообщение
2011	VCLIENT_ERROR_MESS_CONFIRM_SIZE	Превышен размер подтверждения на сообщение
2012	VCLIENT_ERROR_MESS_CONFIRM_READ	Ошибка при чтении подтверждения на сообщение
2013	VCLIENT_ERROR_MESS_CONFIRM_WRITE	Ошибка при записи подтверждения на сообщение
2014	VCLIENT_ERROR_MESS_CONFIRM_CHECK	Ошибка при проверке ЭЦП подтверждения на сообщение
2015	VCLIENT_ERROR_MESS_CONFIRM_CREATE	Ошибка при создании подтверждения на сообщение
2016	VCLIENT_ERROR_MESS_CRYPT_OPEN	Ошибка при открытии криптографических блоков сообщения
2017	VCLIENT_ERROR_MESS_CRYPT_SIZE	Ошибка размера криптографических блоков сообщения
2018	VCLIENT_ERROR_MESS_CRYPT_READ	Ошибка при чтении криптографических блоков сообщения
2019	VCLIENT_ERROR_MESS_CRYPT_CLOSE	Ошибка при закрытии криптографических блоков сообщения
2020	VCLIENT_ERROR_COMM_DELETE	Ошибка при выполнении команды удаления сообщения
2021	VCLIENT_ERROR_COMM_CONFIRM_CREATE	Ошибка создания подтверждения для команды
2022	VCLIENT_ERROR_COMM_CONFIRM_OPEN	Ошибка открытия подтверждения для команды
2023	VCLIENT_ERROR_COMM_CONFIRM_WRITE	Ошибка записи подтверждения для команды

Код	Возвращаемое значение	Описание
2024	VCLIENT_ERROR_BLOCK_ENVELOPED	Неверная структура криптографического блока шифрования
2025	VCLIENT_ERROR_BLOCK_SIGNED	Неверная структура криптографического блока ЭЦП
2026	VCLIENT_ERROR_HANDLE	Неверный дескриптор соединения
2027	VCLIENT_ERROR_MESS_CHECK	Ошибка при проверке сообщения
2028	VCLIENT_ERROR_PARAM	Ошибка параметров
2029	VCLIENT_ERROR_MESS_CONFIRM_CLOSE	Ошибка при закрытии подтверждения на сообщение
2030	VCLIENT_ERROR_VERIFY_DELAYED	Проверка сообщения отложена

## 2. Расширенные коды ошибок

Ошибки при работе с хранилищем сертификатов

Код	Возвращаемое значение	Описание
1	CRTSIMPLE_ERROR_LOCAL_STORE_PATH	Не удалось определить каталог хранилища сертификатов
2	CRTSIMPLE_ERROR_LOCAL_STORE_BIND	Не удалось установить соединение с сервером хранилища сертификатов
3	CRTSIMPLE_ERROR_LOCAL_STORE_READ	Ошибка при чтении данных с сервера хранилища сертификатов
4	CRTSIMPLE_ERROR_LOCAL_STORE_UNBIND	Ошибка при отключении от сервера хранилища сертификатов
5	CRTSIMPLE_ERROR_CERT_NOT_FOUND	Сертификат не найден в хранилище сертификатов
6	CRTSIMPLE_ERROR_CERT_STRUCTURE	Ошибка в структуре полученного сертификата
7	CRTSIMPLE_ERROR_DN_SIZE	Превышен размер имени Distinguished Name в сертификате
8	CRTSIMPLE_ERROR_DN_STRUCTURE	Недопустимая структура имени Distinguished Name в сертификате
9	CRTSIMPLE_ERROR_DDE_INITIALIZED	Ошибка при инициализации DDE (только Windows)
10	CRTSIMPLE_ERROR_GET_STORE_PATH	Не удалось получить каталог хранилища сертификатов (только Windows)
11	CRTSIMPLE_ERROR_LOAD_CERTIFICATE	Ошибка при загрузке полученного сертификата
12	CRTSIMPLE_ERROR_CRL_NOT_FOUND	СОС не найден в хранилище сертификатов
14	CRTSIMPLE_ERROR_CERT_NOT_VALIDITY	Найденные сертификаты имеют неверный срок действия



Код	Возвращаемое значение	Описание
15	CRTSIMPLE_ERROR_DN_COMPARE	DN имена не совпадают

Ошибки при работе с CMS сообщениями

Код	Возвращаемое значение	Описание
21	TCMSSIMPLE_ERROR_CMS_NOT_LOADED	Сообщение не загружено
22	TCMSSIMPLE_ERROR_CMS_CREATE	Ошибка создания сообщения
23	TCMSSIMPLE_ERROR_CMS_PACKET_TYPE	Неверный тип сообщения
24	TCMSSIMPLE_ERROR_CMS_PACKET_ADD_CONTENT	Ошибка при добавлении содержимого в зашифрованное сообщение (CMS Enveloped)
25	TCMSSIMPLE_ERROR_CMS_PACKET_ADD_ORIGINATOR	Ошибка при добавлении данных об отправителе в сообщение
26	TCMSSIMPLE_ERROR_CMS_PACKET_ADD_RECIPIENT	Ошибка при добавлении данных о получателе в сообщение
27	TCMSSIMPLE_ERROR_CMS_PACKET_SET_UNPROT_ATTR	Ошибка при формировании неподписываемых атрибутов
28	TCMSSIMPLE_ERROR_CMS_PACKET_ADD_UNPROT_ATTR	Ошибка при добавлении неподписываемых атрибутов
29	TCMSSIMPLE_ERROR_CMS_SIGNED_ADD_CONTENT	Ошибка при добавлении содержимого в подписанное сообщение (CMS Signed)
30	TCMSSIMPLE_ERROR_CMS_SIGNED_TIME	Ошибка получения времени при формировании атрибутов подписанного сообщения
31	TCMSSIMPLE_ERROR_CMS_SIGNED_NAME	Ошибка получения имени из сертификата при формировании атрибутов подписанного сообщения
32	TCMSSIMPLE_ERROR_CMS_SIGNED_CONTENT_TYPE	Ошибка установки типа содержимого при формировании атрибутов подписанного сообщения

Код	Возвращаемое значение	Описание
33	TCMSSIMPLE_ERROR_CMS_SIGNED_DIGEST	Ошибка установки хеширования при формировании атрибутов подписанного сообщения
34	TCMSSIMPLE_ERROR_CMS_SIGNED_SIGNING_TIME	Ошибка установки времени подписи при формировании атрибутов подписанного сообщения
35	TCMSSIMPLE_ERROR_CMS_SIGNED_SENDER_NAME	Ошибка установки имени отправителя при формировании атрибутов подписанного сообщения
36	TCMSSIMPLE_ERROR_CMS_SIGNED_GET_ATTR	Ошибка получения атрибутов подписанного сообщения
37	TCMSSIMPLE_ERROR_CMS_SIGNED_ADD_SIGN	Ошибка при добавлении ЭЦП в подписанное сообщение
38	TCMSSIMPLE_ERROR_CMS_LOAD_TYPE	Неверный тип загружаемого сообщения
39	TCMSSIMPLE_ERROR_CMS_LOAD_ORIGINATOR_COUNT	Не указано или неверное количество авторов в загружаемом сообщении
40	TCMSSIMPLE_ERROR_CMS_LOAD_ORIGINATOR_TYPE	Не указан или неверный тип автора в загружаемом сообщении
41	TCMSSIMPLE_ERROR_CMS_LOAD_ORIGINATOR_ISSUER	Не указан или неверный издатель сертификата автора в загружаемом сообщении
42	TCMSSIMPLE_ERROR_CMS_LOAD_ORIGINATOR_ISSUER_SIZE	Превышен размер данных издателя сертификата автора в загружаемом сообщении
43	TCMSSIMPLE_ERROR_CMS_LOAD_ORIGINATOR_ISSUER_SERIAL	Не указан или неверный серийный номер сертификата автора в загружаемом сообщении
44	TCMSSIMPLE_ERROR_CMS_LOAD_RECIPIENT	Не указан или неверный получатель в загружаемом сообщении
45	TCMSSIMPLE_ERROR_CMS_LOAD_RECIPIENT_TYPE	Не указан или неверный тип получателя в загружаемом сообщении
46	TCMSSIMPLE_ERROR_CMS_LOAD_RECIPIENT_SIZE	Превышен размер данных получателя в загружаемом сообщении

Код	Возвращаемое значение	Описание
47	TCMSSIMPLE_ERROR_CMS_LOAD_RECIPIENT_SERIAL	Не указан или неверный серийный номер сертификата получателя в загружаемом сообщении
48	TCMSSIMPLE_ERROR_CMS_LOAD_RECIPIENT_KEY	Не указан или неверный зашифрованный сессионный ключ получателя в загружаемом сообщении
49	TCMSSIMPLE_ERROR_CMS_LOAD_UNPROT_ATTR	Загружаемое сообщение не включает в себя поле незащищенных атрибутов
50	TCMSSIMPLE_ERROR_CMS_LOAD_UNPROT_ATTR_ORIG	Загружаемое сообщение не содержит значение атрибута автора
51	TCMSSIMPLE_ERROR_CMS_LOAD_UNPROT_ATTR_RECP	Загружаемое сообщение не содержит значение атрибута получателей
52	TCMSSIMPLE_ERROR_CMS_LOAD_UNPROT_ATTR_TIME	Загружаемое сообщение не содержит значение атрибута времени подписи
53	TCMSSIMPLE_ERROR_CMS_LOAD_CONTENT	Неверный параметр содержания в загружаемом сообщении
54	TCMSSIMPLE_ERROR_CMS_LOAD_ISSUER	Не указан или неверный издатель сертификата подписавшего загружаемое сообщение
55	TCMSSIMPLE_ERROR_CMS_LOAD_ISSUER_SIZE	Превышен размер данных издателя сертификата подписавшего загружаемое сообщение
56	TCMSSIMPLE_ERROR_CMS_LOAD_ISSUER_SERIAL	Не указан или неверный серийный номер сертификата подписавшего загружаемое сообщение
57	TCMSSIMPLE_ERROR_CMS_LOAD_SIGNATURE	Не удастся получить одну из подписей в загружаемом сообщении
58	TCMSSIMPLE_ERROR_CMS_LOAD_ATTR	Загружаемое сообщение не включает в себя поле подписываемых атрибутов
59	TCMSSIMPLE_ERROR_CMS_LOAD_ATTR_SIGNER	Не удастся получить подписываемые атрибуты загружаемого сообщения

Код	Возвращаемое значение	Описание
60	TCMSSIMPLE_ERROR_CMS_LOAD_ATTR_TIME	Загружаемое сообщение не содержит значение атрибута времени подписи
61	TCMSSIMPLE_ERROR_CMS_LOAD_ATTR_SENDER	Загружаемое сообщение не содержит значение атрибута автора
62	TCMSSIMPLE_ERROR_CMS_LOAD_ATTR_DIGEST	Загружаемое сообщение не содержит значение атрибута хеширования
63	TCMSSIMPLE_ERROR_CMS_LOAD_VERSION	Не удастся получить версию поля данных подписавшего загружаемое сообщение
64	TCMSSIMPLE_ERROR_CMS_LOAD_SIGNER_ID	Не удастся получить тип идентификатора сертификата подписавшего загружаемое сообщение
65	TCMSSIMPLE_ERROR_CMS_ADD_CERT	Ошибка при вложении сертификата в сообщение
66	TCMSSIMPLE_ERROR_CMS_ADD_CRL	Ошибка при вложении СОС в сообщение
67	TCMSSIMPLE_ERROR_CMS_CERT_NOT_FOUND	Вложенный сертификат не найден
68	TCMSSIMPLE_ERROR_CMS_DN_SENDER_NOT_SIGNER	Имя отправителя не соответствует имени подписывающего
69	TCMSSIMPLE_ERROR_CMS_INVALID_SIGNATURE	Неверная подпись (ЭЦП)
71	TCMS_LIGHT_ERROR_CMS_NOT_LOADED	Сообщение не загружено
72	TCMS_LIGHT_ERROR_INVALID_INDEX	Не удастся получить открытый ключ в сертификатах по номеру (превышен номер)
73	TCMS_LIGHT_ERROR_GET_OPEN_KEY	Не удастся получить открытый ключ в сертификатах по имени (имя не найдено)
74	TCMS_LIGHT_ERROR_CMS_COUNT	Количество получателей не соответствует количеству зашифрованных сессионных ключей
75	TCMS_LIGHT_ERROR_CMS_NOT_ATTR	Не установлены атрибуты для подписи

Криптографические ошибки в защищенном протоколе обмена данными

Код	Возвращаемое значение	Описание
81	AUTH_ERROR_CSP_PATH	Не удастся установить путь криптопровайдера
82	AUTH_ERROR_CSP_LOAD	Не удастся загрузить библиотеку криптопровайдера
83	AUTH_ERROR_CSP_NOT_LOADED	Библиотека криптопровайдера не загружена
84	AUTH_ERROR_CSP_CONTEXT	Не удастся получить контекст криптопровайдера
85	AUTH_ERROR_CSP_KEY_GEN	Ошибка при генерации сессионного ключа
86	AUTH_ERROR_CSP_KEY_EXPORT	Ошибка при экспорте ключей
87	AUTH_ERROR_CSP_KEY_IMPORT	Ошибка при импорте ключей
88	AUTH_ERROR_CSP_GET_KEY_USER	Не удастся получить ключ пользователя
89	AUTH_ERROR_SERIAL_CONVERT	Ошибка при конвертации серийного номера сертификата пользователя
90	AUTH_ERROR_MESS_SEND_RECIP_KEY	Не установлен сессионный ключ получателя при формировании защищенной команды
91	AUTH_ERROR_MESS_SEND_RECIP_HASH	Не удалось создать хеш объект при формировании защищенной команды
92	AUTH_ERROR_MESS_SEND_RECIP_HASH_DATA	Не удалось добавить данные в хеш объект при формировании защищенной команды
93	AUTH_ERROR_MESS_SEND_RECIP_HASH_DUPL	Не удалось дублировать хеш объект при формировании защищенной команды
94	AUTH_ERROR_MESS_SEND_RECIP_HASH_PARM	Не удалось получить параметры хеш объекта при формировании защищенной команды
95	AUTH_ERROR_MESS_CHECK_MY_KEY	Не установлен свой сессионный ключ при проверке защищенной команды
96	AUTH_ERROR_MESS_CHECK_MY_HASH	Не удалось создать хеш объект при проверке защищенной команды
97	AUTH_ERROR_MESS_CHECK_SIZE	Неверный размер защищенной команды
98	AUTH_ERROR_MESS_CHECK_FORMAT	Неверный формат защищенной команды
99	AUTH_ERROR_MESS_CHECK_MY_HASH_DATA	Не удалось добавить данные в хеш объект при проверке защищенной команды

Код	Возвращаемое значение	Описание
100	AUTH_ERROR_MESS_CHECK_MY_HASH_DUPL	Не удалось дублировать хеш объект при проверке защищенной команды
101	AUTH_ERROR_MESS_CHECK_MY_HASH_PARM	Не удалось получить параметры хеш объекта при проверке защищенной команды
102	AUTH_ERROR_MESS_CHECK_COMPARE	Ошибка аутентификации при проверке защищенной команды
103	AUTH_ERROR_MESS_CREATE_HASH_DATA	Не удалось вычислить хеш при создании сообщения
104	AUTH_ERROR_MESS_CREATE_SIGN_DATA	Не удалось вычислить ЭЦП при создании сообщения
105	AUTH_ERROR_MESS_CHECK_SIGNERS_COUNT	Неверное количество подписавших в проверяемом сообщении
106	AUTH_ERROR_MESS_CHECK_HASH_DATA	Не удалось вычислить хеш при проверке сообщения
107	AUTH_ERROR_MESS_CHECK_HASH_COMPARE	Неверный хеш в сообщении
108	AUTH_ERROR_MESS_CHECK_DATA_COMPARE	Неверная ЭЦП в сообщении
109	AUTH_ERROR_INVALID_PARAM	Неправильный параметр
110	AUTH_ERROR_CSP_LOAD_KEYS	Не удастся загрузить ключи

Ошибки при работе с файловой системой и удаленным сервером

Код	Возвращаемое значение	Описание
121	ERR_OVN_TIMEOUT	Неверный параметр времени ожидания
122	ERR_OVN_CONNECT	Не удалось установить соединение с файловой системой или удалённым сервером
123	ERR_OVN_WRITE	Ошибка при выполнении операции записи/передачи данных
124	ERR_OVN_READ	Ошибка при выполнении операции чтения/получения данных
125	ERR_OVN_MEMORY	Ошибка при выполнении операции выделения памяти
126	ERR_OVN_HANDLE	Неверный дескриптор
127	ERR_OVN_ACCESS	Нет прав доступа на выполнение данной команды
128	ERR_OVN_NOMATCH	Удаляемый объект не существует
129	ERR_OVN_INIT_CSP	Не удастся инициализировать криптопровайдера

Код	Возвращаемое значение	Описание
130	ERR_OVN_SET_PROXY	Неверные параметры прокси сервера
131	ERR_OVN_REPL_SIZE	Размер ответа сервера превосходит допустимый размер
132	ERR_OVN_REPL_NEGATIVE	Отрицательный ответ сервера
133	ERR_OVN_CONNECTION_CLOSE	Соединение закрыто
134	ERR_OVN_NOT_CONNECTION	Соединение не установлено
135	ERR_OVN_CONNECTION_TIMEOUT	Соединение потеряно по тайм-ауту
136	ERR_OVN_GETHOSTBYNAME	Не удалось получить адрес сервера по имени
137	ERR_OVN_SERVER_NOT_AVAILABLE	Сервер недоступен
138	ERR_OVN_USER_BLOCKED	Пользователь заблокирован
139	ERR_OVN_USER_ADDRESS	Пользователь попытался соединиться с другого адреса

Ошибки при работе с криптопровайдером

Код	Возвращаемое значение	Описание
151	TSCSP_ERROR_LoadCSP	Ошибка загрузки криптопровайдера
152	TSCSP_ERROR_PARAM	Ошибка в параметрах
153	TSCSP_ERROR_hProv	Ошибочный дескриптор криптопровайдера
154	TSCSP_ERROR_hHash	Ошибочный дескриптор хеш-объекта
155	TSCSP_ERROR_ssKey	Ошибочный дескриптор сессионного ключа
156	TSCSP_ERROR_CPAcquireContext	Ошибка в методе криптопровайдера CPAcquireContext
157	TSCSP_ERROR_CPGenKey	Ошибка в методе криптопровайдера CPGenKey
158	TSCSP_ERROR_CPGetUserKey	Ошибка в методе криптопровайдера CPGetUserKey
159	TSCSP_ERROR_CPExportKey	Ошибка в методе криптопровайдера CPExportKey
160	TSCSP_ERROR_CPImportKey	Ошибка в методе криптопровайдера CPImportKey
161	TSCSP_ERROR_CPCreateHash	Ошибка в методе криптопровайдера CPCreateHash

Код	Возвращаемое значение	Описание
162	TSCSP_ERROR_CPHashData	Ошибка в методе криптопровайдера CPHashData
163	TSCSP_ERROR_CPGetHashParam	Ошибка в методе криптопровайдера CPGetHashParam
164	TSCSP_ERROR_CPSignHash	Ошибка в методе криптопровайдера CPSignHash
165	TSCSP_ERROR_CPEncrypt	Ошибка в методе криптопровайдера CPEncrypt
166	TSCSP_ERROR_CPDecrypt	Ошибка в методе криптопровайдера CPDecrypt
167	TSCSP_ERROR_CPVerifySignature	Ошибка в методе криптопровайдера CPVerifySignature
168	TSCSP_ERROR_CPGetKeyParam	Ошибка в методе криптопровайдера CPGetKeyParam
169	TSCSP_ERROR_CPSetKeyParam	Ошибка в методе криптопровайдера CPSetKeyParam

Ошибки при работе с сообщениями

Код	Возвращаемое значение	Описание
200	TArchive_ERROR_PARAM	Ошибка в параметрах
201	TArchive_ERROR_isOpen	Сообщение уже открыто
202	TArchive_ERROR_TYPE	Ошибочный тип сообщения
203	TArchive_ERROR_NOT_FOUND	Объект не найден
204	TArchive_ERROR_BZ2_bzopen	Ошибка открытия компрессора BZ2
205	TArchive_ERROR_BZ2_bzwrite	Ошибка записи в компрессор BZ2
220	TDeArchive_ERROR_PARAM	Ошибка в параметрах
221	TDeArchive_ERROR_isOpen	Сообщение уже открыто
222	TDeArchive_ERROR_isNotOpen	Сообщение не открыто
223	TDeArchive_ERROR_TYPE	Ошибочный тип сообщения
224	TDeArchive_ERROR_MAGIC	Ошибка MAGIC-заголовка сообщения
225	TDeArchive_ERROR_NOT_FOUND	Объект не найден
226	TDeArchive_ERROR_MHeadCRC	Ошибка CRC основного заголовка



Код	Возвращаемое значение	Описание
227	TDeArchive_ERROR_MemoCRC	Ошибка CRC заголовка комментария
228	TDeArchive_ERROR_DHeadCRC	Ошибка CRC заголовка каталога
229	TDeArchive_ERROR_DNameSize	Ошибочный размер имени каталога
230	TDeArchive_ERROR_DNameCRC	Ошибка CRC имени каталога
231	TDeArchive_ERROR_FHeadCRC	Ошибка CRC заголовка файла
232	TDeArchive_ERROR_FNameSize	Ошибочный размер имени файла
233	TDeArchive_ERROR_FNameCRC	Ошибка CRC имени файла
234	TDeArchive_ERROR_FileCRC	Ошибка CRC файла
235	TDeArchive_ERROR_OutPTR	Выход за границы сообщения
236	TDeArchive_ERROR_TSCSP	Не загружен класс TSCSP
237	TDeArchive_ERROR_CMSEnv	Ошибочный блок CMS-Envp
238	TDeArchive_ERROR_CMSSign	Ошибочный блок CMS-Sign
239	TDeArchive_ERROR_BZ2_bzopen	Ошибка открытия компрессора BZ2
240	TDeArchive_ERROR_BZ2_bzread	Ошибка чтения из компрессора BZ2
241	TDeArchive_ERROR_FileExists	Файл уже существует
261	CArch_smp_ERROR_LoadLib	Ошибка загрузки библиотеки libvista.dll
262	CArch_smp_ERROR_CMSSnotSignedType	Сообщение не подписано
270	CRTV_ERR_ALLOC_MEMORY	Ошибка выделения памяти
271	CRTV_ERR_BAD_CERT_BODY	Неправильная структура сертификата
272	CRTV_ERR_BAD_CRL_BODY	Неправильная структура СОС
273	CRTV_ERR_BAD_CERT_VALID	Неверный срок действия сертификата
274	CRTV_ERR_BAD_CRL_VALID	Неверный срок действия СОС
275	CRTV_ERR_FIND_CA_CERT	Подпись сертификата или СОС не соответствует СА
276	CRTV_ERR_VAR_BAD_SIGNALG	Неправильный алгоритм подписи

Код	Возвращаемое значение	Описание
277	CRTV_ERR_BAD_ASN_STRUCT	Неправильная структура ASN.1
278	CRTV_ERR_CERT_REVOKED	Сертификат отозван
279	CRTV_ERR_BAD_CERT_COMPATIBLE	Сертификат не соответствует CA
280	CRTV_ERR_BAD_CERT_SIGNATURE	Неправильная подпись (ЭЦП)
281	CRTV_ERR_FIND_FILE_CRL	Не найден файл СОС
282	CRTV_ERR_OPEN_FILE_CRL	Ошибка открытия файла СОС
283	CRTV_ERR_READ_FILE_CRL	Ошибка чтения файла СОС
284	CRTV_ERR_CERT_POLICY	Неправильные политики сертификата

## Предупреждения

Код	Возвращаемое значение	Описание
300	WARNING_CONFIRM_CONTENT_ERROR	Предупреждение: Подтверждение содержит информацию об ошибке
301	WARNING_MESSAGE_CONTENT_ERROR	Предупреждение: Ошибка проверки сообщения

## 3. Другие ошибки

Ошибки при работе с Центром регистрации

Код	Возвращаемое значение	Описание
321	RA_ERR_PARAM_URL	Неверно задан URL
322	RA_ERR_PARAM	Неверный параметр
323	RA_ERR_SSL_CREATE	Ошибка создания SSL-контекста
324	RA_ERR_SSL_CONNECT	Не удалось установить соединение по SSL
325	RA_ERR_LDAP_BIND	Не удалось установить соединение LDAP
326	RA_ERR_LDAP_EXTENSION	Не удалось отправить запрос LDAP
327	RA_ERR_CONNECT	Не удалось установить соединение
328	RA_ERR_NOT_CONNECTED	Соединение не установлено

Код	Возвращаемое значение	Описание
329	RA_ERR_SEND_REQUEST	Не удалось отправить запрос
330	RA_ERR_RECV_RESPONSE	Не удалось получить ответ

## Ошибки HTTP

Код	Возвращаемое значение	Описание
7	ERR_TIMEOUT	Время ожидания вышло
8	ERR_ONCLOSE	Клиент закрыл соединение
13	ERR_RECV_DATA	Ошибка чтения данных из сокета
14	ERR_SEND_DATA	Ошибка записи данных в сокет
15	ERR_NOTCONNECTION	Нет соединения
34	ERR_STRING_LIMIT	Слишком большая строка входных данных
35	ERR_ABORT	Сервер остановлен