Сервер Модуля Согласования

Руководство пользователя

Киев, 2021

СОДЕРЖАНИЕ

ПРИЛОЖЕНИЕ Б. ЗАГОЛОВОЧНЫЙ ФАЙЛ *GOHUB.CLIENT.ERRORS.H* **227**

ПРИЛОЖЕНИЕ В. ФАЙЛ ОПИСАНИЯ COM-ИНТЕРФЕЙСОВ GOHUBCLIENTCOM.IDL 230

Список сокращений

АС	–	автоматизированная система
СМС	–	Сервер Модуля Согласования
АРМ ГО	–	автоматизированное рабочее место грузоотправителя
ЭПД	–	электронный перевозочный документ
ЭД	–	электронные данные документа
ЭЦП	–	электронно-цифровая подпись
ЦСК	–	центр сертификации ключей
XML	–	extensible markup language
DLL	–	динамически линкуемая библиотека
УЗ	–	Украинские железные дороги
ПИ	–	Предварительное информирование

1. Концепция информационного взаимодействия

1.1.Общие положения

Сервер Модуля Согласования – модуль, позволяющий прикладному программному обеспечению взаимодействовать с АС «Клиент УЗ». Пользователи получают возможность формировать перевозочные документы при помощи собственных программ с последующей передачей их в АС «Клиент УЗ». Кроме того, *Сервер Модуля Согласования* позволяет пользовательским программам синхронизовать состояние документов (ЭПД) электронных данных (ЭД) и черновиков пользовательской программной системы с состоянием соответствующих объектов системы АС «Клиент УЗ».

Физически *Сервер Модуля Согласования* представляет собой службу Windows, запущенную на одном ПК, к которой подключаются *Клиенты СМС*.

Для облегчения реализации взаимодействия прикладных программ пользователей с *Клиентом СМС* в состав поставки включено несколько вариантов библиотек *Клиента*, предоставляющие прикладным программам дружественный программный интерфейс для следующих языков программирования: C, C++, COM/OLE, C#, др. языки платформы *.NET*.

На рисунке 1 представлена примерная схема взаимодействия прикладной программы пользователя с АС «Клиент УЗ» с использованием *Сервера Модуля Согласования*.

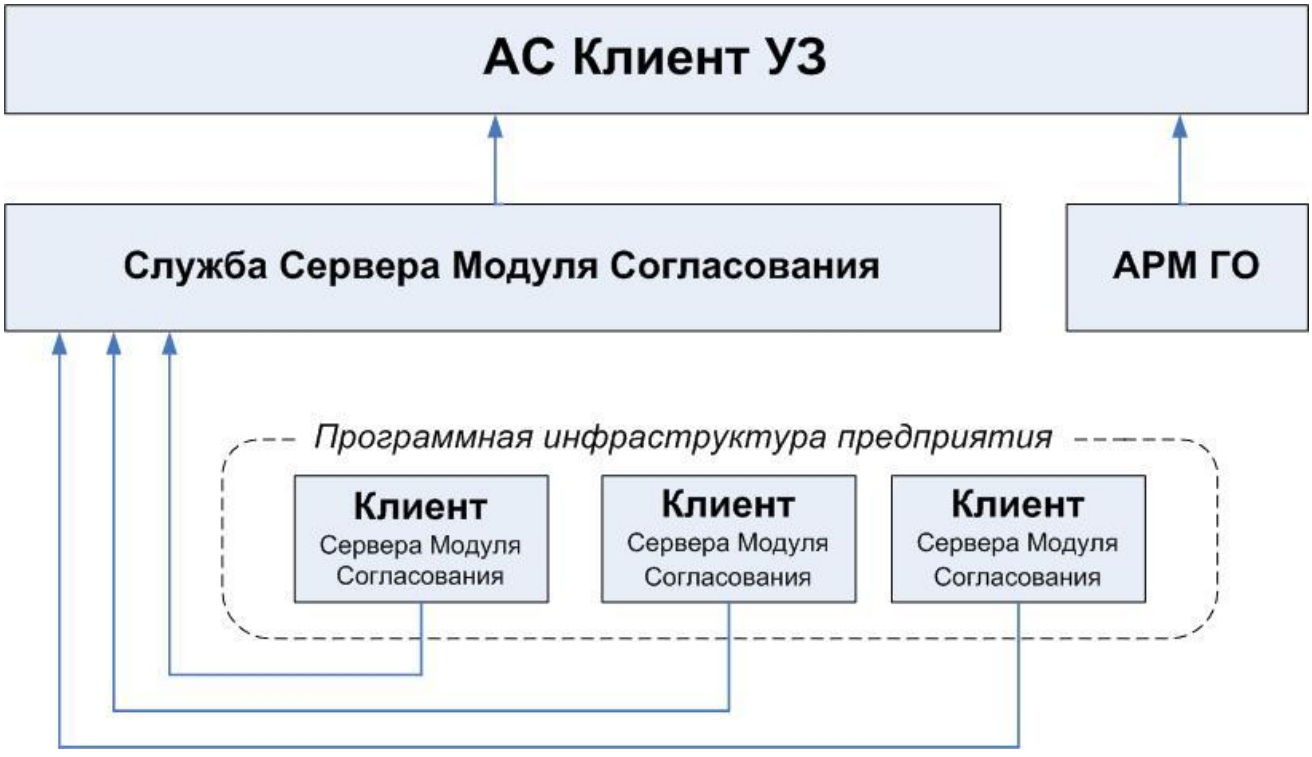


Рисунок 1

1.2. Функции модуля согласования

- передача перевозочных и сопроводительных документов, созданных в пользовательских программах, в АС «Клиент УЗ»;
- получение информации об изменениях (и списка изменений), произошедших в ЭПД, ЭД, черновиках и сопроводительных документах с конкретного момента;
- получение отдельных ЭПД, ЭД, черновиков и сопроводительных документов по идентификатору из АС «Клиент УЗ»;
- передача дозаполненных в пользовательских программах ЭПД и ЭД по прибытию в АС «Клиент УЗ»;
- выполнение наложения ЭЦП;
- выполнение проверки ЭЦП;
- отзыв документов и сопроводительных документов;
- удаление черновиков и сопроводительных документов;
- подключение файлов электронных ключей и управление подключенными ключами;
- получение заявок из АС «Месплан»;
- получение суточных перечней из АС «Клиент УЗ».

2. Руководство по установке и настройке

2.1. Инструкция по установке *Службы Сервера Модуля Согласования*

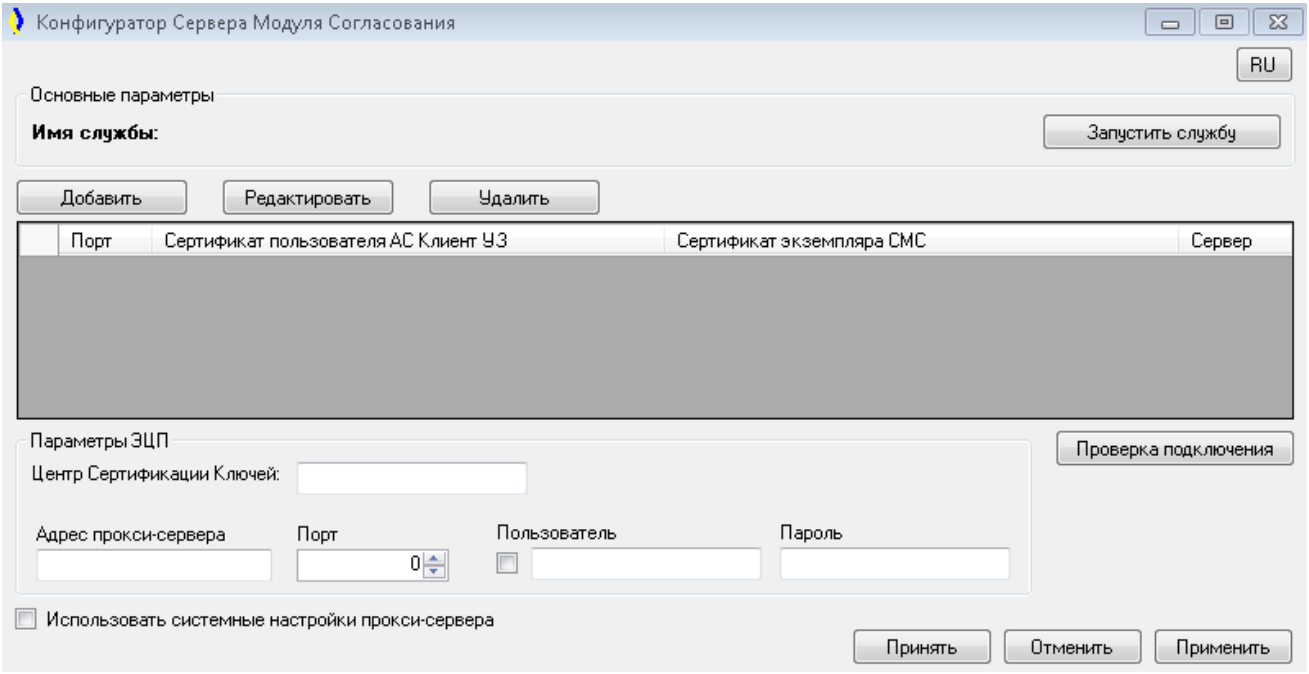
Для установки *службы Сервера Модуля Согласования* нужно запустить инсталлятор *tmsoft.gohub.service.setup.msi*.

В процессе инсталляции будет предложено выбрать директорию установки, по умолчанию - *"C:\Program Files\KPD-UZ\Сервер Модуля Узгодження"*.

В результате установки *Служба СМС* будет инсталлирована в систему как служба Windows с именем *TMSoft.Gohub (в списке служб отображается с именем «Модуль Согласования»)*. Средствами операционной системы можно выбрать режим запуска службы (*ручной или автоматический, по умолчанию - автоматический*) и стартовать службу.

2.2.Настройка *Службы Сервера Модуля Согласования*

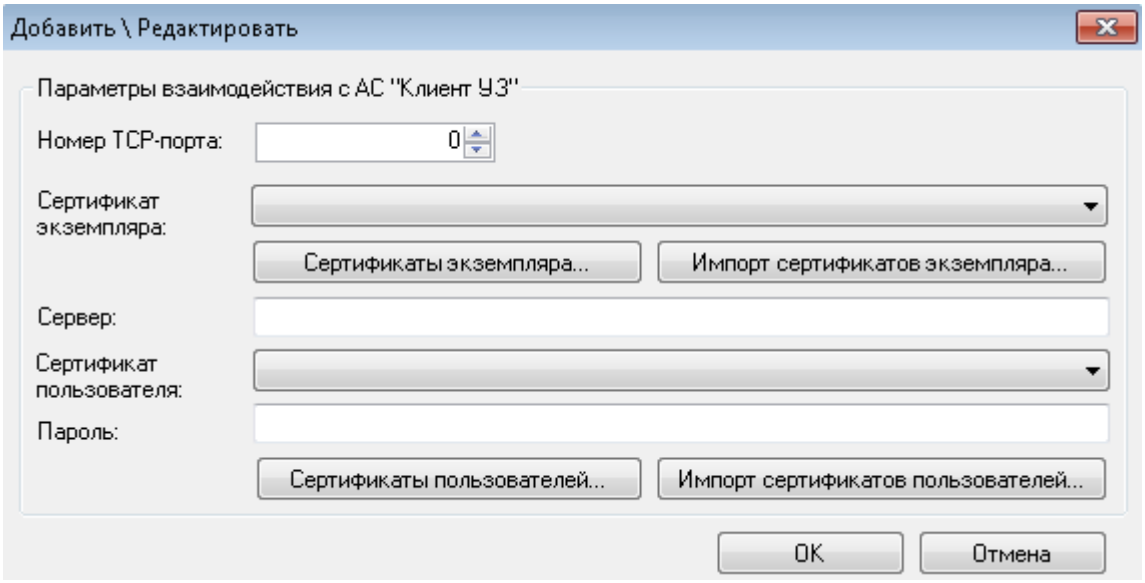
Для настройки *Службы СМС* в комплект поставки включена утилита *Конфигуратор Сервера Модуля Согласования*, которая запускается по завершении инсталляции, а также доступна, через системное меню: *Пуск -> KPD-UZ -> Сервер Модуль Согласования -> Конфигуратор*.



Основное окно: *Конфигуратор Сервера Модуля Согласования*

Конфигуратор позволяет настроить следующие свойства:

- Запуск Службы СМС: кнопка «Запустить службу» / «Остановить службу» запускает / останавливает Службу СМС;
- Создавать, редактировать и удалять данные о подключениях Сервера Модуля Согласования к АС «Клиент» УЗ;
- Параметры подключения к ЭЦП:
    - Адрес сервера Центра Сертификации Ключей (поумолчанию – *«csk.uz.gov.ua»*);
    - Настройки прокси-сервера – адрес и порт;
    - Настройки пользователя локальной сети – имя и пароль.



Окно добавления/редактирования данных о подключении

- Номер TCP-порта, по которому осуществляется взаимодействие *Службы* и *Клиента СМС*);
- Параметры взаимодействия с АС «Клиент УЗ»:
    - Сертификат экземпляра – сертификат предоставляющий возможность работы Службе СМС и взаимодействовать с АС «Клиент УЗ»;
    - Кнопка «Импорт…» предоставляет возможность внести данные о сертификате экземпляра из файла сертификата;
    - В поле «Сервер» задается адрес сервера АС «Клиент УЗ», который автоматически устанавливается из сертификата пользователя АС «Клиент УЗ»;
    - В поле «Имя пользователя» выбирается пользователь АС «Клиент УЗ», который устанавливается из сертификата пользователя АС «Клиент УЗ»;
    - В поле «Пароль пользователя» задается пароль доступа к АС «Клиент УЗ»;
    - Кнопка «Сертификаты…» открывает браузер сертификатов, относящихся пользователям Службы СМС, от которых идет обращение к АС «Клиент УЗ» и хранящихся в операционной системе;
    - Кнопка «Импорт сертификатов пользователей…» предоставляет возможность внести данные о пользователе, от которого идет обращение к АС «Клиент УЗ» из файла сертификата;

2.3.Описание установки *Клиента Сервера Модуля Согласования*

Для установки *Клиента Сервера Модуля Согласования* нужно запустить инсталлятор *tmsoft.gohub.client.setup.msi.*

В процессе инсталляции будет предложено выбрать директорию установки, по умолчанию - *C:\Program Files\ KPD-UZ \Клиент Модуля Согласования*.

В результате установки *Клиент СМС* в папке установки размещается следующие файлы:

- Папка ***bin*** содержит запускающийся файл консоли (его можно увидеть и в разделе «Пуск/Все программы/*KPD-UZ*/Клиент модуля согласования» как «*Консоль Модуля Согласования*») и три файла разных вариантов библиотеки Клиента:
    - ***TMSoft.gohub.client.dll*** - *динамически линкуемая библиотека* (DLL) для семейства операционных систем Windows, начиная с Windows 2000 и выше;
    - ***TMSoft.gohub.client.com.dll*** – *COM/OLE объект*;
    - ***TMSoft.Gohub.Client.Net.dll*** - *.NET версия библиотеки*.

    Примечание: библиотеки можно свободно перемещать, копировать и т.д., кроме библиотеки COM/OLE, которая при установке регистрируется в системе с указанием пути размещения файла.
- Папка ***doc*** содержит два файла – Руководство пользователя и *Changes.txt* с описаниями изменений в предыдущих версиях;
- Папка ***inc*** содержит файлы для линкования библиотек – заголовочные файлы *gohub.client.h* и *gohub.client.errors.h*, для динамически линкуемой библиотеки и файл *gohubclientcom.idl* с объявлением интерфейсов для *COM/OLE* версии библиотеки;
- Папка ***lib*** содержит два файла: статическая библиотека TMSoft.gohub.client.lib и библиотека типов TMSoft.gohub.client.com.tlb для приложений, использующих *COM/OLE* версию.
- Папка ***samples*** содержит xml-файлы образцов сообщений для формирования электронных перевозочных документов разных видов отправок, а так же файл *Gohub.Client.Test.xls*, с макросами-примерами использования *COM/OLE* версии библиотеки;
- Папка ***schemas*** содержит файлы *xml-схем* (*.xsd*) с формализованным описанием структур xml-сообщений электронных перевозочных документов.

3. Общее описание библиотеки

Прикладная библиотека «*Клиент Сервера Модуля Согласования*» (далее - *Библиотека*) предназначена для упрощения взаимодействия прикладных программ со *Службой Сервера Модуля Согласования*. *Библиотека* реализована в соответствии с требованиями, изложенными в «Концепции информационного взаимодействия» *Сервера Модуля Согласования*.

Физически *Библиотека* состоит из нескольких вариантов библиотеки *TMSoft.gohub.client*, под разные языки программирования:

- *TMSoft.gohub.client.dll - динамически линкуемой библиотекой* (DLL) для семейства операционных систем Windows, начиная с Windows 2000 и выше;
- *TMSoft.gohub.client.com.dll – COM/OLE объект;*
- *TMSoft.Gohub.Client.Net.dll - .NET версия библиотеки.*

Библиотека предоставляет набор простых и в то же время эффективных средств диагностики ошибок, возникающих при ее использовании.

Программный интерфейс *Библиотеки* предоставляет прикладным программам возможность решать следующие задачи:

- Отправлять в АС «Клиент УЗ» перевозочные документы, созданные вне системы АС «Клиент УЗ».
- Отслеживать появление в АС «Клиент УЗ» новых документов и модификацию существующих.
- Запрашивать из АС «Клиент УЗ» отдельные документы по их уникальным идентификаторам.
- Выполнять перекодировку документов и получать подробную информацию о возможных ошибках, возникающих при работе с *Библиотекой.*
- Отзывать документы и удалять черновики.
- Накладывать ЭЦП на перевозочные документы.
- Производить проверку ЭЦП, наложенной на перевозочные документы.
- Получать заявки из АС «Месплан»
- Получать суточные перечни.

Для взаимодействия с АС «Клиент УЗ» прикладная программа должна подключиться к *Серверу Модуля Согласования* при помощи функции *«Создать_соединение»*. Результатом вызова этой функции является дескриптор (указатель) объекта или объект соединения, который будет использоваться во всех операциях взаимодействия с АС «Клиент УЗ»: отправки и запроса перевозочных документов, а также запроса списков модификаций.

Для отправки черновика перевозочного документа в АС «Клиент УЗ» прикладная программа сначала должна создать объект документа при помощи функции «Создать_документ», либо загрузить документ из файла при помощи функции

«Загрузить_документ». После этого можно отправить документ в АС «Клиент УЗ» при помощи функции «Отправить_документ». Формат черновиков перевозочных документов, отправляемых в АС «Клиент УЗ», должен соответствовать требованиям изложенным в документе «Структура, состав и формат атрибутов электронного перевозочного документа», опубликованном на официальном сайте УЗ.

В результате отправки в АС «Клиент УЗ» документ получает уникальный идентификатор по которому можно в будущем запросить документ из АС «Клиент УЗ» на протяжении всего его жизненного цикла. Для запроса документа из АС «Клиент УЗ» служит функция «Запросить_документ».

Полученные документы можно сохранить в файл при помощи функции «Сохранить_документ», либо получить текст документа в виде строки при помощи функции «Текст_документа». Эта функция возвращает текст документа (без xml-заголовка) в текущей кодировке, которая может быть задана при помощи функции «Установить_кодировку» (по умолчанию - 1251).

Документы, полученные из АС «Клиент УЗ» по умолчанию имеют кодировку *utf-8* (кодовая страница 65001). При сохранении документа в файл при помощи функции «Сохранить_документ» вы можете указать кодовую страницу в которой хотите сохранить документ.

Помимо уникального идентификатора, каждый документ в системе АС «Клиент УЗ» имеет такое важное свойство как номер ревизии. Номер ревизии, так же как и идентификатор, назначаются документу при первом попадании в систему. Однако, в отличие от идентификатора, номер ревизии не является постоянным свойством и изменяет свое значение при каждой модификации документа. Все модификации по всем документам системы АС «Клиент УЗ» имеют сквозную нумерацию.

Номера ревизии позволяют внешним системам отслеживать изменения в документах, находящихся в АС «Клиент УЗ». При помощи функции «Запросить_следующий_документ» приложения могут получать документы в соответствии с последовательностью их последовательности их модификаций.

Все названия функций приведены на русском языке специально, так как семантика в разных *Библиотеках* отличается, но сущность логики действий и команд не меняется.

4. Динамически линкуемая библиотека tmsoft.gohub.client.dll

4.1. Краткое описание библиотеки

Gohub.client.dll является *динамически линкуемой библиотекой* (DLL) для семейства операционных систем Windows, начиная с Windows 2000 и выше. Для работы первых версий *Библиотеки* потребуются также платформа *.NET Framework* версии не ниже 2.0, а также библиотеки времени исполнения *Microsoft CRT*. Дистрибутивы этих дополнительных компонентов входят в комплект поставки *Библиотеки*.

Дополнительно в комплект поставки включены: статическая библиотека *gohub.client.lib* для линковки с *Библиотекой* программ, написанных на языке *Microsoft Visual C/C++*, и заголовочный файл *gohub.client.h*, содержащий декларации всех функций, типов и констант, составляющих программный интерфейс *Библиотеки*.

В *Библиотеке* есть ряд функций работающих со строчными данными исключительно в юникодовой кодировке UTF-16 Little Indian *(далее - UTF-16)*, все они содержат в названии букву ***W***.

Более детальное описание программного интерфейса библиотеки описано в следующих пунктах. Так же можно просмотреть заголовочные файлы gohub.client.h (Приложение А) и gohub.client.errors.h (Приложение Б).

Для облегчения и ускорения знакомства с программным интерфейсом Библиотеки служат примеры ее использования, приведенные в п.4.5.

4.2. Типы данных

`GohubBool`	Булева переменная
`GohubWChar`	Символ представленный в кодировке UTF-16
`GohubConnection`	Соединения с *Модулем Согласования*
`GohubDocument`	Перевозочные документы
`GohubAttachment`	Сопроводительные документы
`GohubEData`	Электронные данные предварительного информирования
`GohubPiPackage`	Пакеты предварительного информирования
`GohubPiPackageToEData`	Информация об электронных данных в пакете предварительного информирования
`GohubFdu92`	Документы ФДУ-92
`GohubGu46`	Документы ГУ-46
`GohubGu45`	Документы ГУ-45
`GohubInformServicesDoc`	Документы информационных услуг
`GohubDispatchInfo`	Список № Заказа для ПД

`GohubBool`

Булева переменная.

Объявление:

```
typedef int GohubBool;
```

`GohubWChar`

Строчная переменная в кодировке UTF-16.

Объявление:

```
typedef unsigned char GohubWChar;
```

GohubConnection

Объекты этого типа служат для представления подключения прикладной программы к службе *Модуля Согласования.*

Объявление:

```
typedef struct GohubConnection GohubConnection;
```

Примечание:

Полное определение этого типа недоступно прикладной программе. Объекты этого типа, а также процедуры их создания и уничтожения инкапсулированы в Библиотеке. Взаимодействие с объетами этого типа возможно только через открытые функции *Библиотеки,* которые оперуют указателями на эти объекты.

GohubDocument

Объекты этого типа служат для представления электронных перевозочных документов.

Объявление:

```
typedef struct GohubDocument GohubDocument;
```

Примечание:

Полное определение этого типа недоступно прикладной программе. Объекты этого типа, а также процедуры их создания и уничтожения инкапсулированы в Библиотеке. Взаимодействие с объетами этого типа возможно только через открытые функции *Библиотеки,* которые оперуют указателями на эти объекты.

GohubAttachment

Объекты этого типа служат для представления электронных сопроводительных документов.

Объявление:

```
typedef struct GohubAttachment GohubAttachment;
```

Примечание:

Полное определение этого типа недоступно прикладной программе. Объекты этого типа, а также процедуры их создания и уничтожения инкапсулированы в Библиотеке. Взаимодействие с объетами этого типа возможно только через открытые функции *Библиотеки,* которые оперуют указателями на эти объекты.

GohubEData

Объекты этого типа служат для представления электронных данных предварительного информирования.

Объявление:

```
typedef struct GohubEData GohubEData;
```

Примечание:

Полное определение этого типа недоступно прикладной программе. Объекты этого типа, а также процедуры их создания и уничтожения инкапсулированы в Библиотеке. Взаимодействие с объетами этого типа возможно только через открытые функции *Библиотеки,* которые оперуют указателями на эти объекты.

GohubPiPackage

Объекты этого типа служат для представления пакетов предварительного информирования.

Объявление:

```
typedef struct GohubPiPackage GohubPiPackage;
```

Примечание:

Полное определение этого типа недоступно прикладной программе. Объекты этого типа, а также процедуры их создания и уничтожения инкапсулированы в Библиотеке. Взаимодействие с объетами этого типа возможно только через открытые функции *Библиотеки,* которые оперуют указателями на эти объекты.

GohubPiPackageToEData

Объекты этого типа служат для представления информации об электронных данных в пакете предварительного информирования.

Объявление:

```
typedef struct GohubPiPackageToEData GohubPiPackageToEData;
```

Примечание:

Полное определение этого типа недоступно прикладной программе. Объекты этого типа, а также процедуры их создания и уничтожения инкапсулированы в Библиотеке. Взаимодействие с объетами этого типа возможно только через открытые функции *Библиотеки,* которые оперуют указателями на эти объекты.

GohubFdu92

Объекты этого типа служат для представления электронных накопительных карточек ФДУ-92.

Объявление:

```
typedef struct GohubFdu92 GohubFdu92;
```

Примечание:

Полное определение этого типа недоступно прикладной программе. Объекты этого типа, а также процедуры их создания и уничтожения инкапсулированы в Библиотеке. Взаимодействие с объетами этого типа возможно только через открытые функции Библиотеки, которые оперуют указателями на эти объекты.

GohubGu46

Объекты этого типа служат для представления электронных ведомостей о пользовании вагонами/контейнерами ГУ-46.

Объявление:

```
typedef struct GohubGu46 GohubGu46;
```

Примечание:

Полное определение этого типа недоступно прикладной программе. Объекты этого типа, а также процедуры их создания и уничтожения инкапсулированы в Библиотеке. Взаимодействие с объетами этого типа возможно только через открытые функции Библиотеки, которые оперуют указателями на эти объекты.

GohubGu45

Объекты этого типа служат для представления электронных памяток о подаче/уборке вагонов и выдаче/приёме контейнеров ГУ-45.

Объявление:

```
typedef struct GohubGu45 GohubGu45;
```

Примечание:

Полное определение этого типа недоступно прикладной программе. Объекты этого типа, а также процедуры их создания и уничтожения инкапсулированы в Библиотеке.

Взаимодействие с объетами этого типа возможно только через открытые функции Библиотеки, которые оперуют указателями на эти объекты.

GohubInformServicesDoc

Объекты этого типа служат для представления документов информационных услуг.

Объявление:

```
typedef struct GohubInformServicesDoc GohubInformServicesDoc;
```

Примечание:

Полное определение этого типа недоступно прикладной программе. Объекты этого типа, а также процедуры их создания и уничтожения инкапсулированы в Библиотеке. Взаимодействие с объетами этого типа возможно только через открытые функции Библиотеки, которые оперуют указателями на эти объекты.

GohubDispatchInfo

Объекты этого типа служат для получения перечня информации о заказе на согласование перевозки по данным календаря планирования перевозок зерновых грузов (за последние 5 дней от текущей даты).

Объявление:

```
typedef struct GohubDispatchInfo GohubDispatchInfo;
```

Примечание:

Полное определение этого типа недоступно прикладной программе. Объекты этого типа, а также процедуры их создания и уничтожения инкапсулированы в Библиотеке. Взаимодействие с объетами этого типа возможно только через открытые функции Библиотеки, которые оперуют указателями на эти объекты.

GohubPSTDInfo

Объекты этого типа служат для получения перечня информации о № заказа Согласования сокращения сроком доставки.

Объявление:

```
typedef struct GohubPSTDInfo GohubPSTDInfo;
```

Примечание:

Полное определение этого типа недоступно прикладной программе. Объекты этого типа, а также процедуры их создания и уничтожения инкапсулированы в Библиотеке. Взаимодействие с объетами этого типа возможно только через открытые функции Библиотеки, которые оперуют указателями на эти объекты.

4.3. Статусы документов

4.3.1. Состояния перевозочного документа

Коды состояния перевозочного документа в *Библиотеке* представлены перечислением (*enum*) `GohubDocumentStatus`, содержащим набор констант.

Сводная таблица кодов состояний перевозочного документа

`gohub_document_status_unknown = 0`	Статус неизвестен
`gohub_document_status_draft = 1`	Черновик
`gohub_document_status_sending = 2`	Документ передается товарному кассиру
`gohub_document_status_registered = 3`	Документ передан товарному кассиру
`gohub_document_status_reclaiming = 4`	Документ отзывается от товарного кассира

gohub_document_status_accepted = 5	Груз принят к перевозке
gohub_document_status_delivered = 6	Груз прибыл
gohub_document_status_recieved = 7	Груз получен получателем
gohub_document_status_uncredited = 8	Документ раскредитован товарным кассиром
gohub_document_status_recieved_draft = 9	Груз получен получателем и редактируется
gohub_document_status_recieved_sending = 10	Груз получен получателем и передан товарному кассиру
gohub_document_status_recieved_reclaiming = 11	Груз получен получателем и отзывается от товарного кассира
gohub_document_status_canceled = 12	Документ испорчен товарным кассиром
gohub_document_status_locked = 13	Документ заблокирован

Более детальное описание программного интерфейса библиотеки можно получить из заголовочного файла gohub.client.h (Приложение А).

4.3.2. Состояния накопительной карточки ФДУ-92

Коды состояния накопительной карточки ФДУ-92 в *Библиотеке* представлены перечислением (*enum*) `GohubFdu92Status`, содержащим набор констант.

Сводная таблица кодов состояний накопительной карточки ФДУ-92

gohub_fdu92_status_unknown = -1	Статус неизвестен
gohub_fdu92_status_approoving = 0	Документ получен на согласование
gohub_fdu92_status_approving_modified = 1	Документ получен на согласование и отредактирован пользователем
gohub_fdu92_status_confirmed = 2	Документ согласован и подписан
gohub_fdu92_status_canceled = 3	Документ отменен
gohub_fdu92_status_agreed_noted_sending = 4	Документ согласован с правками, отправляется
gohub_fdu92_status_agreed = 5	Документ согласован
gohub_fdu92_status_agreed_noted = 6	Документ согласован с правками
gohub_fdu92_status_expired = 7	Документ не согласован вовремя
gohub_fdu92_status_agreed_sending = 8	Документ согласован, отправляется
gohub_fdu92_status_confirmed_paper = 9	Документ подтвержден товарным кассиром
gohub_fdu92_status_rejecting = 10	Документ инициированный бумажный, отправляется
gohub_fdu92_status_rejected = 11	Документ инициированный бумажный
gohub_fdu92_status_paper = 60	Бумажный документ

4.3.3. Состояния ведомостей о пользовании вагонами/контейнерами ГУ-46

Коды состояния ведомостей о пользовании вагонами/контейнерами ГУ-46 в *Библиотеке* представлены перечислением *(enum)* `GohubGU46Status`, содержащим набор констант.

Сводная таблица кодов состояний ведомости пользования вагонами/контейнерами ГУ-46

gohub_gu46_status_unknown = -1	Статус неизвестен
gohub_gu46_status_approoving = 0	Документ получен на согласование
gohub_gu46_status_approving_modified = 1	Документ получен на согласование и

	отредактирован пользователем
gohub_gu46_status_confirmed = 2	Документ согласован и подписан
gohub_gu46_status_canceled = 3	Документ отменен
gohub_gu46_status_agreed_noted_sending = 4	Документ согласован с правками, отправляется
gohub_gu46_status_agreed = 5	Документ согласован
gohub_gu46_status_agreed_noted = 6	Документ согласован с правками
gohub_gu46_status_expired = 7	Документ не согласован вовремя
gohub_gu46_status_agreed_sending = 8	Документ согласован, отправляется
gohub_gu46_status_confirmed_paper = 9	Документ подтвержден товарным кассиром
gohub_gu46_status_rejecting = 10	Документ инициированный бумажный, отправляется
gohub_gu46_status_rejected = 11	Документ инициированный бумажный
gohub_gu46_status_paper = 60	Бумажный документ

4.3.4. Состояния памятки о подаче/уборке вагонов и выдаче/приёме контейнеров ГУ-45

Коды состояния памяток о подаче/уборке вагонов и выдачу/приём контейнеров ГУ-45 в *Библиотеке* представлены перечислением *(enum)* `GohubGU45Status`, содержащим набор констант.

Сводная таблица кодов состояний памятки о подаче/уборке вагонов и выдаче/приёме контейнеров ГУ-45

gohub_gu45_status_unknown = -1	Статус неизвестен
gohub_gu45_status_confirmed = 2	Документ согласован и подписан
gohub_gu45_status_canceled = 3	Документ отменен
gohub_gu45_status_confirmed_paper = 9	Документ подтвержден товарным кассиром
gohub_gu45_status_paper = 60	Бумажный документ

4.3.5. Состояние перевозочного документа при запросе статуса документа в подписи за индексом

Коды состояния перевозочного документа в *Библиотеке* представлены перечислением (*enum*) `UzRwcDocStatus`, содержащим набор констант.

Сводная таблица кодов состояний перевозочного документа

none = 0	Статус неизвестен
project = 1	ЕПД, передаваемых грузоотправителем
accepted = 2	ЕПД, принятые к перевозке
resend = 3	ЕПД, которые подверглись изменениям во время следования груза
arrived = 4	ЕПД, прибывшие на станцию назначения
reviewed = 5	ЕПД, передаваемых грузополучателем
uncredited = 6	Раскредитованные ЕПД
foreign = 7	ЕПД, полученные от иностранной ЖД
entered = 8	ЕПД, обработанные по входу на УЗ
exited = 9	ЕПД, обработанные по выходу из УЗ

Более детальное описание программного интерфейса библиотеки можно получить из заголовочного файла gohub.client.h (Приложение А).

4.4. Основные функции

Более детальное описание программного интерфейса библиотеки можно получить из заголовочного файла gohub.client.h (Приложение А).

4.4.1. Работа с кодовыми страницами

`gohub_set_codepage`	Установить кодировку по ее числовому обозначению
`gohub_encoding_codepage`	Получить кодировку по ее названию
`gohub_encoding_codepage_w`	Получить кодировку по ее названию `(параметр в UTF-16)`

`gohub_set_codepage`

Установить текущую кодовую страницу для всех строчных параметров *(кроме тех, что в UTF-16).*

Объявление:

```
int gohub_set_codepage(int codepage);
```

Параметры:

`codepage`

Кодировка страницы;

Результат:

Устанавливает кодировочную страницу для передаваемых строчных данных.

`gohub_encoding_codepage`

Получить цифровое обозначение кодовой страницы по ее текстовому названию.

Объявление:

```
int gohub_encoding_codepage(const char* encodingName);
```

Параметры:

`encodingName`

Название кодовой страницы;

Результат:

В случае успеха - Возвращает цифровую кодировку обозначающую запрошенную страницу. В случае ошибки – возвращает ноль. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др.`

`gohub_encoding_codepage_w`

Получить цифровое обозначение кодовой страницы по ее текстовому названию в кодировке UTF-16.

Объявление:

```
int gohub_encoding_codepage_w(
const GohubWChar* encodingName);
```

Параметры:

`encodingName`

Название кодовой страницы;

Результат:

В случае успеха - Возвращает цифровую кодировку обозначающую запрошенную страницу. В случае ошибки – возвращает ноль. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др.`

4.4.2. Подключение к Модулю Согласования

`gohub_connect`	Создать соединение с *Модулем Согласования*
`gohub_connect_w`	Создать соединение с *Модулем Согласования* `(параметр в UTF-16)`
`gohub_disconnect`	Закрыть соединение с *Модулем Согласования*

`gohub_connect`

Установить соединение с *Модулем Согласования.*

Объявление:

```
GohubConnection* gohub_connect(
      const char* host,
      int port);
```

Параметры:

`host`

DNS-имя или IP адрес компьютера, на котором установлен *Модуль Согласования*;

`port`

Номер порта для установления TCP соединения с *Модулем Согласования.*

Результат:

В случае успеха - Указатель соединения с *Модулем Согласования*. В случае ошибки – нулевой указатель. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

Примечание:

По окончании работы с соединением, его необходимо закрыть при помощи функции *`gohub_disconnect`.*

`gohub_connect_w`

Установить соединение с *Модулем Согласования.*

Объявление:

```
GohubConnection* gohub_connect(
      const GohubWChar* host,
      int port);
```

Параметры:

`host`

DNS-имя или IP адрес компьютера, на котором установлен *Модуль Согласования* *`(в UTF-16)`*;

`port`

Номер порта для установления TCP соединения с *Модулем Согласования.*

Результат:

В случае успеха - Указатель соединения с *Модулем Согласования*. В случае ошибки – нулевой указатель. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

Примечание:

По окончании работы с соединением, его необходимо закрыть при помощи функции *`gohub_disconnect`.*

gohub_disconnect

Закрыть соединение с Модулем Согласования.

Объявление:

```
GohubBool gohub_disconnect(
      GohubConnection* connection);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

4.4.3. Работа с документами

gohub_load_document	Загрузить документ
gohub_load_document_w	Загрузить документ (в UTF-16)
gohub_create_document	Создать документ
gohub_create_document_w	Создать документ (в UTF-16)
gohub_query_document	Запросить документ
gohub_query_document_w	Запросить документ (в UTF-16)
gohub_query_next_document	Запросить следующий документ
gohub_document_id	Запросить ID документа
gohub_document_id_w	Запросить ID документа (в UTF-16)
gohub_document_revision	Запросить ревизию документа
gohub_document_text	Запросить текст документа
gohub_document_text_w	Запросить текст документа (в UTF-16)
gohub_document_data_text	Запросить текст электронных данных документа
gohub_document_data_text_w	Запросить текст электронных данных документа (в UTF-16)
gohub_document_status	Запросить статус документа
gohub_document_size	Запросить предполагаемый размер документа представленный в текущей кодовой странице
gohub_document_measure_equip_num	Получение сведений вагоноизмерительной техники
gohub_document_measure_equip_num_w	Получение сведений вагоноизмерительной техники (в UTF-16)
gohub_document_set_measure_equip_num_w	Установка сведений вагоноизмерительной техники
gohub_document_set_verified_empty_weight_for_wagon	Установка уточненного веса тары вагона
gohub_document_get_verified_empty_weight_for_wagon	Получение уточненного веса тары вагона
gohub_document_business_unit_num	Получение номера филиала ЧАО УЗ
gohub_document_business_unit_num_w	Получение номера филиала ЧАО УЗ (в UTF-16)
gohub_document_set_business_unit_num	Установка номера филиала ЧАО УЗ
gohub_document_set_business_unit_num_w	Установка номера филиала ЧАО УЗ (в UTF-16)
gohub_document_get_foreign_not_accept	Получение отметки возвращения непринятых пограничными станциями иностранных железных дорог вагонов на

	территорию Украины
gohub_send_document	Отправить документ
gohub_save_document	Сохранить документ
gohub_save_document_w	Сохранить документ (в UTF-16)
gohub_save_document_data	Сохранить электронные данные документа
gohub_save_document_data_w	Сохранить электронные данные документа (в UTF-16)
gohub_close_document	Закрыть документ
gohub_reclaim_document	Отозвать документ
gohub_reclaim_document_w	Отозвать документ (в UTF-16)
gohub_delete_document	Удалить документ
gohub_delete_document_w	Удалить документ (в UTF-16)
gohub_send_received_document	Отправить документ по прибытию
gohub_send_received_document_w	Отправить документ по прибытию (в UTF-16)
gohub_document_get_otpr	Сохранить актуальный текст документа
gohub_document_get_otpr_w	Сохранить актуальный текст документа (в кодировке UTF-8)
gohub_document_get_otpr_string	Получить актуальный текст перевозочного документа
gohub_document_get_otpr_string_w	Получить актуальный текст перевозочного документа (в кодировке UTF-8)
gohub_query_and_save_document_printable_form	Запросить печатную форму документа
gohub_query_and_save_document_printable_form_w	Запросить печатную форму документа (в UTF-16)

gohub_load_document

Создание объекта документа (GohubDocument) из xml-файла.

Объявление:

```
GohubDocument* gohub_load_document(
        const char* path);
```

Параметры:

path

путь к xml-файлу, из которого загружается документ;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_load_document_w

Создание объекта документа (GohubDocument) из xml-файла.

Объявление:

```
GohubDocument* gohub_load_document_w(
        const GohubWChar* path);
```

Параметры:

path

путь к xml-файлу в кодировке UTF-16, из которого загружается документ;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_create_document

Создание объекта документа (`GohubDocument`) из строки с xml-структурой.

Объявление:

```
GohubDocument* gohub_create_document(
      const char* content);
```

Параметры:

`content`

строка с xml-структурой с содержанием документа;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_create_document_w

Создание объекта документа (`GohubDocument`) из строки с xml-структурой.

Объявление:

```
GohubDocument* gohub_create_document_w(
      const GohubWChar* content);
```

Параметры:

`content`

строка с xml-структурой с содержанием документа в кодировке UTF-16;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_query_document

Запрос документа с сервера АС «Клиент УЗ» по его ID.

Объявление:

```
GohubDocument* gohub_query_document(
      GohubConnection* connection,
      const char* documentId);
```

Параметры:

`connection`

Указатель соединения с Модулем Согласования, полученный ранее при помощи функции `gohub_connect`;

`documentId`

уникальный идентификатор документа, который запрашивается;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_query_document_w

Запрос документа с сервера АС «Клиент УЗ» по его ID.

Объявление:

```
GohubDocument* gohub_query_document_w(
      GohubConnection* connection,
      const GohubWChar* documentId);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`documentId`

уникальный идентификатор документа (в кодировке UTF-16), который запрашивается;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_query_next_document

Запрос документа с сервера АС «Клиент УЗ» следующего по списку ревизий, начиная от ревизии переданной параметром.

Объявление:

```
GohubDocument* gohub_query_next_document(
      GohubConnection* connection,
      int lastRevision);
```

Параметры:

`connection`

Указатель соединения с Модулем Согласования, полученный ранее при помощи функции `gohub_connect;`

`lastRevision`

ревизия документа, от которой начинается поиск следующего документа.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_query_next_document2

Запрос документа с сервера АС «Клиент УЗ» следующего по списку ревизий, начиная от ревизии переданной параметром.

Объявление:

```
GohubDocument* gohub_query_next_document2(
      GohubConnection* connection,
      int lastRevision);
```

Параметры:

`connection`

Указатель соединения с Модулем Согласования, полученный ранее при помощи функции `gohub_connect;`

`lastRevision`

ревизия документа, от которой начинается поиск следующего документа.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_document_id

Запрос уникального ID документа с сервера АС «Клиент УЗ».

Объявление:

```
GohubDocument* gohub_query_document(
      GohubConnection* connection,
      const char* documentId);
```

Параметры:

`connection`

Указатель соединения с Модулем Согласования, полученный ранее при помощи функции `gohub_connect`;

`documentId`

уникальный идентификатор документа, который запрашивается;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_document_id_w

Запрос документа с сервера АС «Клиент УЗ» по его ID.

Объявление:

```
GohubDocument* gohub_query_document_w(
      GohubConnection* connection,
      const GohubWChar* documentId);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`documentId`

уникальный идентификатор документа (в кодировке UTF-16), который запрашивается;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_document_revision

Запрос ревизии документа.

Объявление:

```
int gohub_document_revision(
      GohubDocument* document);
```

Параметры:

`document`

Объект документа (GohubDocument), ревизия которого запрашивается;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_document_text

Запрос текста документа из объекта документ (GohubDocument).

Объявление:

```
const char* gohub_document_text(
      GohubDocument* document);
```

Параметры:

document

Объект документа (GohubDocument), текст которого запрашивается;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_document_text_w

Запрос текста документа из объекта документ (GohubDocument) в кодировке UTF-16.

Объявление:

```
const GohubWChar* gohub_document_text_w(
      GohubDocument* document);
```

Параметры:

document

Объект документа (GohubDocument), текст которого запрашивается;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_document_data_text

Запрос текста электронных данных документа из объекта документ (GohubDocument) в заданной версии ЭПД.

Объявление:

```
const char* gohub_document_text(
      GohubDocument* document,
      int epdVersion);
```

Параметры:

document

Объект документа (GohubDocument), текст электронных данных которого запрашивается;

epdVersion

версия ЭПД в которой получить электронный данные документа (принимаемые значения 10, 11, 12, 13, 14, что соответствует версии ЭПД 1.0, 1.1, 1.2, 1.3, 1.4);

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_document_data_text_w

Запрос текста электронных данных документа (в кодировке UTF-16) из объекта документ (GohubDocument) в заданной версии ЭПД.

Объявление:

```
const GohubWChar* gohub_document_text_w(
      GohubDocument* document,
      int epdVersion);
```

Параметры:

document

Объект документа (GohubDocument), текст электронных данных которого запрашивается;

epdVersion

версия ЭПД в которой получить электронный данные документа (принимаемые значения 10, 11, 12, 13, 14, что соответствует версии ЭПД 1.0, 1.1, 1.2, 1.3, 1.4);

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_document_status

Запрос статуса документа.

Объявление:

```
GohubDocumentStatus gohub_document_status(
      GohubDocument* document);
```

Параметры:

document

Объект документа (GohubDocument), статус которого запрашивается;

Результат:

В случае успеха – значение из перечисления GohubDocumentStatus. В случае ошибки – «-1». Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_document_size

Запрос предполагаемого размера файла, который получится при сохранении документа в текущей кодировке (GohubDocument).

Объявление:

```
int gohub_document_size(
      GohubDocument* document);
```

Параметры:

document

Объект документа (GohubDocument), текст которого запрашивается;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_document_measure_equip_num

Получение сведений вагоноизмерительной техники

Объявление:

```
const char* gohub_document_measure_equip_num(
      GohubDocument* document);
```

Параметры:

`document`

Объект документа (`GohubDocument`), который отправляется на сервер;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_document_measure_equip_num_w

Получение сведений вагоноизмерительной техники в кодировке UTF-16.

Объявление:

```
const GohubWChar* gohub_document_measure_equip_num_w(
      GohubDocument* document);
```

Параметры:

`document`

Объект документа (`GohubDocument`), который отправляется на сервер;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_document_set_measure_equip_num

Установка сведений вагоноизмерительной техники.

Объявление:

```
GohubBool gohub_document_set_measure_equip_num(
      GohubDocument* document, const char* value);
```

Параметры:

`document`

Объект документа (`GohubDocument`), который отправляется на сервер;

`value`

устанавливаемое значение сведений вагоноизмерительной;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_document_set_measure_equip_num_w

Установка сведений вагоноизмерительной техники в кодировке UTF-16.

Объявление:

```
GohubBool gohub_document_set_measure_equip_num_w(
      GohubDocument* document, const GohubWChar* value);
```

Параметры:

`document`

Объект документа (`GohubDocument`), который отправляется на сервер;

`value`

устанавливаемое значение сведений вагоноизмерительной техники (в кодировке UTF-16);

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_document_get_verified_empty_weight_for_wagon

Получение уточненного веса тары вагона

Объявление:

```
int gohub_document_ get_verified_empty_weight_for_wagon (
      GohubDocument* doc, int wagonIndex);
```

Параметры:

`document`

Объект документа (`GohubDocument`), который отправляется на сервер;

`wagonIndex`

Индекс вагона уточненный вес тары которого необходимо получить. Индексация вагонов начинается с нуля;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_document_set_verified_empty_weight_for_wagon

Установка уточненного веса тары вагона

Объявление:

```
GohubBool gohub_document_ set_verified_empty_weight_for_wagon (
      GohubDocument* doc, int wagonIndex, int weight);
```

Параметры:

`document`

Объект документа (`GohubDocument`), который отправляется на сервер;

`wagonIndex`

Индекс вагона уточненный вес тары которого необходимо изменить. Индексация вагонов начинается с нуля;

`weight`

Уточненный вес тары который необходимо установить

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_document_warrant_type

Позволяет определить тип оснований для получения груза

Объявление:

```
int gohub_document_warrant_type(GohubDocument* document);
```

Параметры:

`document`

Объект документа (`GohubDocument`), который отправляется на сервер;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – -1. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_document_set_warrant_type

Позволяет задать тип оснований для получения груза

Объявление:

```
GohubBool gohub_document_set_warrant_type(GohubDocument* document,
int val);
```

Параметры:

`document`

Объект документа (`GohubDocument`), который отправляется на сервер;

`val`

Тип оснований для получения груза (0 - доверенность, 1 - приказ)

Результат:

В случае успеха – true. В случае ошибки – false. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_document_business_unit_num

Получение номера филиала ЧАО УЗ

Объявление:

```
const char* gohub_document_ business_unit_num(
      GohubDocument* document);
```

Параметры:

`document`

Объект документа (`GohubDocument`), который отправляется на сервер;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_document_business_unit_num_w

Получение номера филиала ЧАО УЗ в кодировке UTF-16.

Объявление:

```
const GohubWChar* gohub_document_ business_unit_num_w(
```

```
        GohubDocument* document);
```

Параметры:

`document`

Объект документа (`GohubDocument`), который отправляется на сервер;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_document_set_business_unit_num

Установка номера филиала ЧАО УЗ.

Объявление:

```
GohubBool gohub_document_set_ business_unit_num(
        GohubDocument* document, const char* value);
```

Параметры:

`document`

Объект документа (`GohubDocument`), который отправляется на сервер;

`value`

устанавливаемое значение номера филиала ЧАО УЗ;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_document_set_business_unit_num_w

Установка номера филиала ЧАО УЗ в кодировке UTF-16.

Объявление:

```
GohubBool gohub_document_set_business_unit_num_w(
        GohubDocument* document, const GohubWChar* value);
```

Параметры:

`document`

Объект документа (`GohubDocument`), который отправляется на сервер;

`value`

устанавливаемое значение номера филиала ЧАО УЗ (в кодировке UTF-16);

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_document_get_foreign_not_accept

Получение отметки возвращения непринятых пограничными станциями иностранных железных дорог вагонов на территорию Украины

Объявление:

```
bool gohub_document_get_foreign_not_accept(GohubDocument* document);
```

Параметры:

`document`

Объект документа (`GohubDocument`)

Результат:

В случае успеха – значение true или false. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции gohub_last_error и др.

gohub_send_document

Отправление документа (`GohubDocument`) на сервер АС «Клиент УЗ» и далее – товарному кассиру.

Объявление:

```
GohubBool gohub_send_document(
      GohubConnection* connection,
      GohubDocument* document);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`document`

Объект документа (`GohubDocument`), который отправляется на сервер;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_save_document

Сохранение документа (`GohubDocument`) на диск, по указанному пути и с указанной кодировкой.

Объявление:

```
GohubBool gohub_save_document(
      GohubDocument* document,
      const char* path,
      int codePage);
```

Параметры:

`document`

Объект документа (`GohubDocument`), который отправляется на сервер;

`path`

путь, куда сохраняется файл;

`codePage`

числовое обозначение страницы кодировки, в которой сохраняется файл;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_save_document_w

Сохранение документа (GohubDocument) на диск, по указанному пути и с указанной кодировкой.

Объявление:

```
GohubBool gohub_save_document_w(
      GohubDocument* document,
      const GohubWChar* path,
      int codePage);
```

Параметры:

document

Объект документа (GohubDocument), который отправляется на сервер;

path

Путь, куда сохраняется файл (в кодировке UTF-16);

codePage

Числовое обозначение страницы кодировки, в которой сохраняется файл;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_save_document_data

Сохранение электронных данных документа (GohubDocument) на диск, по указанному пути и с указанной кодировкой.

Объявление:

```
GohubBool gohub_save_document_data(
      GohubDocument* document,
      const char* path,
      int codePage,
      int epdVersion);
```

Параметры:

document

Объект документа (GohubDocument), электронные данные которого запрашиваются;

path

путь, куда сохраняется файл;

codePage

числовое обозначение страницы кодировки, в которой сохраняется файл;

epdVersion

версия ЭПД в которой получить электронный данные документа (принимаемые значения 10, 11, 12, 13, 14, что соответствует версии ЭПД 1.0, 1.1, 1.2, 1.3, 1.4);

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_save_document_data_w

Сохранение электронных данных документа (GohubDocument) на диск, по указанному пути и с указанной кодировкой.

Объявление:

```
GohubBool gohub_save_document_data_w(
      GohubDocument* document,
      const GohubWChar* path,
      int codePage,
      int epdVersion);
```

Параметры:

document

Объект документа (GohubDocument), электронные данные которого запрашиваются;

path

Путь, куда сохраняется файл (в кодировке UTF-16);

codePage

Числовое обозначение страницы кодировки, в которой сохраняется файл;

epdVersion

версия ЭПД в которой получить электронный данные документа (принимаемые значения 10, 11, 12, 13, 14, что соответствует версии ЭПД 1.0, 1.1, 1.2, 1.3, 1.4);

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_close_document

Закрытие документа (GohubDocument).

Объявление:

```
GohubBool gohub_close_document(
      GohubDocument* document);
```

Параметры:

document

Объект документа (GohubDocument).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_reclaim_document

Отзыв документа.

Объявление:

```
GohubBool gohub_reclaim_document(
      GohubConnection* connection,
      const char* documentId);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`documentId`

уникальный идентификатор документа, который запрашивается;

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_reclaim_document_w

Отзыв документа.

Объявление:

```
GohubBool gohub_reclaim_document_w(
      GohubConnection* connection,
      const GohubWChar* documentId);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`documentId`

уникальный идентификатор документа (в кодировке UTF-16), который запрашивается;

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_delete_document

Удаление документа.

Объявление:

```
GohubBool gohub_delete_document(
      GohubConnection* connection,
      const char* documentId);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`documentId`

уникальный идентификатор документа, который запрашивается.

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_delete_document_w

Удаление документа.

Объявление:

```
GohubBool gohub_delete_document_w(
      GohubConnection* connection,
      const GohubWChar* char documentId);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`documentId`

уникальный идентификатор документа (в кодировке UTF-16), который запрашивается.

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_send_received_document

Отправление документа по прибытию (`GohubDocument`) на сервер АС «Клиент УЗ» и далее – товарному кассиру.

Объявление:

```
GohubBool gohub_send_document(
      GohubConnection* connection,
      GohubDocument* document
      const char* documentId);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`document`

Объект документа (`GohubDocument)`, который отправляется на сервер;

`documentId`

уникальный идентификатор документа (в кодировке UTF-16), который отправляется.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_send_received_document_w

Отправление документа по прибытию (`GohubDocument`) на сервер АС «Клиент УЗ» и далее – товарному кассиру.

Объявление:

```
GohubBool gohub_send_document(
      GohubConnection* connection,
      GohubDocument* document
      const GohubWChar* char documentId);
```

Параметры:

connection

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции gohub_connect;

document

Объект документа (GohubDocument), который отправляется на сервер;

documentId

уникальный идентификатор документа (в кодировке UTF-16), который отправляется.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_document_get_otpr

Сохранение актуального текста перевозочного документа (GohubDocument) в формате xml на диск, по указанному пути.

Объявление:

```
GohubBool gohub_document_get_otpr(
      GohubDocument* document,
      const char* path);
```

Параметры:

document

Объект документа (GohubDocument), электронные данные которого запрашиваются;

path

путь, куда сохраняется файл;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_document_get_otpr_w

Сохранение актуального текста перевозочного документа (GohubDocument) в формате xml на диск, по указанному пути (в кодировке UTF-8).

Объявление:

```
GohubBool gohub_document_get_otpr_w(
      GohubDocument* document,
      const GohubWChar* path);
```

Параметры:

document

Объект документа (GohubDocument), электронные данные которого запрашиваются;

path

Путь, куда сохраняется файл (в кодировке UTF-16);

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_document_get_otpr_string

Позволяет получить актуальный текст перевозочного документа (GohubDocument) в виде строки.

Объявление:

```
const char* gohub_document_get_otpr_string(GohubDocument* document);
```

Параметры:

document

Объект документа (GohubDocument), электронные данные которого запрашиваются;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_document_get_otpr_string_w

Позволяет получить актуальный текст перевозочного документа (GohubDocument) в виде строки (в кодировке UTF-8).

Объявление:

```
GohubBool const GohubWChar* gohub_document_get_otpr_string_w(
GohubDocument* document);
```

Параметры:

document

Объект документа (GohubDocument), электронные данные которого запрашиваются;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_document_warning

Запрос текста предупреждение после отправки документа.

Объявление:

```
const char* gohub_document_warning(GohubDocument* document);
```

Параметры:

document

Объект документа (GohubDocument), электронные данные которого запрашиваются;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_document_warning_w

Запрос текста предупреждение после отправки документа (в кодировке UTF-16).

Объявление:

```
const GohubWChar*  gohub_document_warning_w(GohubDocument* document);
```

Параметры:

document

Объект документа (GohubDocument), электронные данные которого запрашиваются;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_query_and_save_document_printable_form

Запросить печатную форму документа по его ID и сохранить ее в файл.

Объявление:

```
GohubBool gohub_query_and_save_document_printable_form(
      GohubConnection* connection,
      const char* documentId,
      const char* path);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect;

`documentId`

уникальный идентификатор документа;

`path`

путь, по которому сохранить запрошенную печатную форму.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_query_and_save_document_printable_form_w

Запросить печатную форму документа по его ID (в кодировке UTF-16) и сохранить ее в файл.

Объявление:

```
GohubBool gohub_query_and_save_document_printable_form_w(
      GohubConnection* connection,
      const GohubWChar* documentId,
      const GohubWChar* path);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect;

`documentId`

уникальный идентификатор документа (в кодировке UTF-16);

`path`

путь, по которому сохранить запрошенную печатную форму.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

4.4.4. Работа с сопроводительными документами

gohub_load_attachment	Загрузить сопроводительный документ, для бланка ГУ и ЦИМ
gohub_load_attachment_with_user_data	Загрузка сопроводительного документа (для бланков ГУ и ЦИМ) с электронными данными пользователя.
gohub_load_attachment_w	Загрузить сопроводительный документ для бланка ГУ и ЦИМ (в UTF-16)
gohub_load_attachment_with_user_data_w	Загрузка сопроводительного документа (для бланков ГУ и ЦИМ) с электронными данными пользователя (в UTF-16)
gohub_load_smgs_attachment	Загрузить сопроводительный документ для бланка СМГС и ЦИМ/СМГС
gohub_load_smgs_attachment_with_user_data	Загрузка сопроводительного документа (для бланков СМГС и ЦИМ/СМГС) с электронными данными пользователя.
gohub_load_smgs_attachment_w	Загрузить сопроводительный документ для бланка СМГС и ЦИМ/СМГС (в UTF-16)
gohub_load_smgs_attachment_with_user_data_w	Загрузка сопроводительного документа (для бланков СМГС и ЦИМ/СМГС) с электронными данными пользователя. (в UTF-16)
gohub_send_attachment	Отправить сопроводительный документ
gohub_query_attachment	Запросить сопроводительный документ
gohub_query_attachment_w	Запросить сопроводительный документ (в UTF-16)
gohub_save_attachment	Сохранить сопроводительный документ
gohub_save_attachment_with_user_data	Сохранение пользовательских электронных данных сопроводительного документа
gohub_save_attachment_w	Сохранить сопроводительный документ (в UTF-16)
gohub_save_attachment_with_user_data_w	Сохранение пользовательских электронных данных сопроводительного документа (в UTF-16).
gohub_delete_attachment	Удалить сопроводительный документ
gohub_delete_attachment_w	Удалить сопроводительный документ (в UTF-16)
gohub_close_attachment	Закрыть сопроводительный документ
gohub_attachment_id	Запросить ID сопроводительного документа
gohub_attachment_id_w	Запросить ID сопроводительного документа (в UTF-16)
gohub_attachment_type_code	Запросить код типа сопроводительного документа
gohub_attachment_type_code_w	Запросить код типа сопроводительного документа (в UTF-16)
gohub_attachment_name	Запросить имя сопроводительного документа
gohub_attachment_name_w	Запросить имя сопроводительного документа (в UTF-16)
gohub_attachment_reg_number	Запросить регистрационный номер сопроводительного документа
gohub_attachment_reg_number_w	Запросить регистрационный номер сопроводительного документа (в UTF-16)
gohub_attachment_reg_date	Запросить дату регистрации сопроводительного документа
gohub_attachment_reg_date_w	Запросить дату регистрации сопроводительного документа (в UTF-16)
gohub_attachment_valid_from	Запросить дату начала действия сопроводительного документа
gohub_attachment_valid_from_w	Запросить дату начала действия сопроводительного документа (в UTF-16)
gohub_attachment_valid_to	Запросить дату окончания действия сопроводительного документа

gohub_attachment_valid_to_w	Запросить дату окончания действия сопроводительного документа (в UTF-16)
gohub_attachment_description	Запросить дескриптор сопроводительного документа
gohub_attachment_description_w	Запросить дескриптор сопроводительного документа в UTF-16)
gohub_attachment_count	Запросить количество приложенных к перевозочному документу сопроводительных документов
gohub_attachment_id_by_index	Запросить ID сопроводительного документа, по его индексу в списке приложенных к перевозочному документу сопроводительных документов
gohub_attachment_id_by_index_w	Запросить ID сопроводительного документа, по его индексу в списке приложенных к перевозочному документу сопроводительных документов (в UTF-16)

gohub_load_attachment

Загрузка сопроводительного документа (для бланков ГУ и ЦИМ).

Объявление:

```
GohubAttachment* gohub_load_attachment(
      const char* typeCode,
      const char* name,
      const char* regNumber,
      const char* regDate,
      const char* validFrom,
      const char* validTo,
      const char* path);
```

Параметры:

typeCode

Код типа сопроводительного документа;

name

имя сопроводительного документа;

regNumber

регистрационный номер сопроводительного документа;

regDate

дата регистрации сопроводительного документа (формат даты ‘dd.MM.yyyy’);

validFrom

дата начала срока действия сопроводительного документа (формат даты ‘dd.MM.yyyy’);

validTo

дата окончания срока действия сопроводительного документа (формат даты ‘dd.MM.yyyy’);

path

путь к файлу с отсканированным текстом сопроводительного документа.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_load_attachment_with_user_data

Загрузка сопроводительного документа (для бланков ГУ и ЦИМ) с электронными данными пользователя.

Объявление:

```
GohubAttachment* gohub_load_attachment_with_user_data(
      const char* typeCode,
      const char* name,
      const char* regNumber,
      const char* regDate,
      const char* validFrom,
      const char* validTo,
      const char* path ,
      const char* pathUserData);
```

Параметры:

`typeCode`

Код типа сопроводительного документа;

`name`

имя сопроводительного документа;

`regNumber`

регистрационный номер сопроводительного документа;

`regDate`

дата регистрации сопроводительного документа (формат даты ‘dd.MM.yyyy’);

`validFrom`

дата начала срока действия сопроводительного документа (формат даты ‘dd.MM.yyyy’);

`validTo`

дата окончания срока действия сопроводительного документа (формат даты ‘dd.MM.yyyy’);

`path`

путь к файлу с отсканированным текстом сопроводительного документа.

`pathUserData`

путь к файлу с электронными данными сопроводительного документа.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_load_attachment_w

Загрузка сопроводительного документа (для бланков ГУ и ЦИМ) (в UTF-16).

Объявление:

```
GohubAttachment* gohub_load_attachment_w(
      const GohubWChar* typeCode,
      const GohubWChar* name,
      const GohubWChar* regNumber,
      const GohubWChar* regDate,
      const GohubWChar* validFrom,
      const GohubWChar* validTo,
      const GohubWChar* path);
```

Параметры:

`typeCode`

Код типа сопроводительного документа в кодировке UTF-16;

`name`

имя сопроводительного документа в кодировке UTF-16;

`regNumber`

регистрационный номер сопроводительного документа в кодировке UTF-16;

`regDate`

дата регистрации сопроводительного документа (формат даты ‘dd.MM.yyyy’) в кодировке UTF-16;

`validFrom`

дата начала срока действия сопроводительного документа (формат даты ‘dd.MM.yyyy’) в кодировке UTF-16;

`validTo`

дата окончания срока действия сопроводительного документа (формат даты ‘dd.MM.yyyy’) в кодировке UTF-16;

`path`

путь к файлу с отсканированным текстом сопроводительного документа в кодировке UTF-16.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др.`

gohub_load_attachment_with_user_data_w

Загрузка сопроводительного документа (для бланков ГУ и ЦИМ) с электронными данными пользователя `(в UTF-16).`

Объявление:

```
GohubAttachment* gohub_load_attachment_with_user_data_w(
      const GohubWChar* typeCode,
      const GohubWChar* name,
      const GohubWChar* regNumber,
      const GohubWChar* regDate,
      const GohubWChar* validFrom,
      const GohubWChar* validTo,
```

```
        const GohubWChar* path,
        const GohubWChar* pathUserData);
```

Параметры:

`typeCode`

Код типа сопроводительного документа в кодировке UTF-16;

`name`

имя сопроводительного документа в кодировке UTF-16;

`regNumber`

регистрационный номер сопроводительного документа в кодировке UTF-16;

`regDate`

дата регистрации сопроводительного документа (формат даты ‘dd.MM.yyyy’) в кодировке UTF-16;

`validFrom`

дата начала срока действия сопроводительного документа (формат даты ‘dd.MM.yyyy’) в кодировке UTF-16;

`validTo`

дата окончания срока действия сопроводительного документа (формат даты ‘dd.MM.yyyy’) в кодировке UTF-16;

`path`

путь к файлу с отсканированным текстом сопроводительного документа в кодировке UTF-16.

`pathUserData`

путь к файлу с электронными данными сопроводительного документа в кодировке UTF-16.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

`gohub_load_smgs_attachment`

Загрузка сопроводительного документа (для бланков СМГС и ЦИМ/СМГС).

Объявление:

```
GohubAttachment* gohub_load_smgs_attachment(
        const char* smgsTypeCode,
        const char* name,
        const char* regNumber,
        const char* regDate,
        const char* validFrom,
        const char* validTo,
        const char* path);
```

Параметры:

`smgsTypeCode`

Код типа сопроводительного документа согласно информационного руководства СМГС;

`name`

имя сопроводительного документа;

`regNumber`

регистрационный номер сопроводительного документа;

`regDate`

дата регистрации сопроводительного документа (формат даты ‘dd.MM.yyyy’);

`validFrom`

дата начала срока действия сопроводительного документа (формат даты ‘dd.MM.yyyy’);

`validTo`

дата окончания срока действия сопроводительного документа (формат даты ‘dd.MM.yyyy’);

`path`

путь к файлу с отсканированным текстом сопроводительного документа.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_load_smgs_attachment_with_user_data

Загрузка сопроводительного документа (для бланков СМГС и ЦИМ/СМГС) с электронными данными пользователя.

Объявление:

```
GohubAttachment* gohub_load_smgs_attachment_with_user_data(
      const char* smgsTypeCode,
      const char* name,
      const char* regNumber,
      const char* regDate,
      const char* validFrom,
      const char* validTo,
      const char* path,
      const char* pathUserData);
```

Параметры:

`smgsTypeCode`

Код типа сопроводительного документа согласно информационного руководства СМГС;

`name`

имя сопроводительного документа;

`regNumber`

регистрационный номер сопроводительного документа;

`regDate`

дата регистрации сопроводительного документа (формат даты ‘dd.MM.yyyy’);

`validFrom`

дата начала срока действия сопроводительного документа (формат даты ‘dd.MM.yyyy’);

`validTo`

дата окончания срока действия сопроводительного документа (формат даты ‘dd.MM.yyyy’);

`path`

путь к файлу с отсканированным текстом сопроводительного документа.

`pathUserData`

путь к файлу с электронными данными сопроводительного документа.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_load_smgs_attachment_w

Загрузка сопроводительного документа (для бланков СМГС и ЦИМ/СМГС) в UTF-16.

Объявление:

```
GohubAttachment* gohub_load_smgs_attachment_w(
      const GohubWChar* smgsTypeCode,
      const GohubWChar* name,
      const GohubWChar* regNumber,
      const GohubWChar* regDate,
      const GohubWChar* validFrom,
      const GohubWChar* validTo,
      const GohubWChar* path);
```

Параметры:

`smgsTypeCode`

Код типа сопроводительного документа в кодировке UTF-16 согласно информационного руководства СМГС;

`name`

имя сопроводительного документа в кодировке UTF-16;

`regNumber`

регистрационный номер сопроводительного документа в кодировке UTF-16;

`regDate`

дата регистрации сопроводительного документа (формат даты ‘dd.MM.yyyy’) в кодировке UTF-16;

`validFrom`

дата начала срока действия сопроводительного документа (формат даты ‘dd.MM.yyyy’) в кодировке UTF-16;

`validTo`

дата окончания срока действия сопроводительного документа (формат даты ‘dd.MM.yyyy’) в кодировке UTF-16;

`path`

путь к файлу с отсканированным текстом сопроводительного документа в кодировке UTF-16.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции gohub_last_error и др.

gohub_load_smgs_attachment_with_user_data_w

Загрузка сопроводительного документа (для бланков СМГС и ЦИМ/СМГС) с электронными данными пользователя (в UTF-16).

Объявление:

```
GohubAttachment* gohub_load_smgs_attachment_with_user_data_w(
      const GohubWChar* smgsTypeCode,
      const GohubWChar* name,
      const GohubWChar* regNumber,
      const GohubWChar* regDate,
      const GohubWChar* validFrom,
      const GohubWChar* validTo,
      const GohubWChar* path,
      const GohubWChar* pathUserData);
```

Параметры:

`smgsTypeCode`

Код типа сопроводительного документа в кодировке UTF-16 согласно информационного руководства СМГС;

`name`

имя сопроводительного документа в кодировке UTF-16;

`regNumber`

регистрационный номер сопроводительного документа в кодировке UTF-16;

`regDate`

дата регистрации сопроводительного документа (формат даты ‘dd.MM.yyyy’) в кодировке UTF-16;

`validFrom`

дата начала срока действия сопроводительного документа (формат даты ‘dd.MM.yyyy’) в кодировке UTF-16;

`validTo`

дата окончания срока действия сопроводительного документа (формат даты ‘dd.MM.yyyy’) в кодировке UTF-16;

`path`

путь к файлу с отсканированным текстом сопроводительного документа в кодировке UTF-16.

`pathUserData`

путь к файлу с электронными данными сопроводительного документа в кодировке UTF-16.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции gohub_last_error и др.

gohub_send_attachment

Отправление сопроводительного документа.

Объявление:

```
GohubBool gohub_send_attachment(
      GohubConnection* connection,
      GohubAttachment* attachment);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`attachment`

объект документа (`GohubAttachment)`, который отправляется на сервер.

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_query_attachment

Запрос сопроводительного документа.

Объявление:

```
GohubAttachment* gohub_query_attachment(
      GohubConnection* connection,
      const char* attachmentId);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`attachmentId`

уникальный идентификатор сопроводительного документа.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_query_attachment_w

Запрос сопроводительного документа.

Объявление:

```
GohubAttachment* gohub_query_attachment_w(
      GohubConnection* connection,
      const GohubWChar* attachmentId);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`attachmentId`

уникальный идентификатор сопроводительного документа (в кодировке UTF-16).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_query_attachment_with_user_data

Запрос сопроводительного документа с электронными данными пользователя.

Объявление:

```
GohubAttachment* gohub_query_attachment_with_user_data(
      GohubConnection* connection,
      const char* attachmentId);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`attachmentId`

уникальный идентификатор сопроводительного документа.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_query_attachment_with_user_data_w

Запрос сопроводительного документа.

Объявление:

```
GohubAttachment* gohub_query_attachment_with_user_data_w(
      GohubConnection* connection,
      const GohubWChar* attachmentId);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`attachmentId`

уникальный идентификатор сопроводительного документа (в кодировке UTF-16).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_save_attachment

Сохранение сопроводительного документа.

Объявление:

```
GohubBool gohub_save_attachment(
      GohubAttachment* attachment,
      const char* path);
```

Параметры:

`attachment`

Объект документа (`GohubAttachment`), который сохраняется на диск;

`path`

путь, куда сохраняется файл.

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_save_attachment_with_user_data

Сохранение пользовательских электронных данных сопроводительного документа.

Объявление:

```
GohubBool gohub_save_attachment_with_user_data(
      GohubAttachment* attachment,
      const char* path);
```

Параметры:

`attachment`

Объект документа (`GohubAttachment`),электронные данные которого сохраняются на диск;

`path`

путь, куда сохраняется файл.

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_save_attachment_w

Сохранение сопроводительного документа (в UTF-16).

Объявление:

```
GohubBool gohub_save_attachment_w(
      GohubAttachment* attachment,
      const GohubWChar* path);
```

Параметры:

`attachment`

Объект документа (`GohubAttachment`), который сохраняется на диск;

`path`

путь, куда сохраняется файл (в кодировке UTF-16).

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_save_attachment_with_user_data_w

Сохранение пользовательских электронных данных сопроводительного документа (в UTF-16).

Объявление:

```
GohubBool gohub_save_attachment_with_user_data_w(
      GohubAttachment* attachment,
      const GohubWChar* path);
```

Параметры:

`attachment`

Объект документа (`GohubAttachment`),электронные данные которого сохраняются на диск;

`path`

путь, куда сохраняется файл (в кодировке UTF-16).

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_delete_attachment

Удаление сопроводительного документа.

Объявление:

```
GohubBool gohub_delete_attachment(
      GohubConnection* connection,
      const char* attachmentId);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`attachmentId`

уникальный идентификатор сопроводительного документа, который удаляется.

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_delete_attachment_w

Запрос сопроводительного документа.

Объявление:

```
GohubBool gohub_delete_attachment_w(
      GohubConnection* connection,
      const GohubWChar* attachmentId);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`attachmentId`

уникальный идентификатор сопроводительного документа (в кодировке UTF-16), который удаляется.

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_close_attachment

Закрытие сопроводительного документа.

Объявление:

```
GohubBool gohub_close_attachment(
      GohubAttachment* attachment);
```

Параметры:

`attachment`

Объект документа (`GohubAttachment).`

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_attachment_id

Запрос ID сопроводительного документа.

Объявление:

```
const char* gohub_attachment_id(
      GohubAttachment* attachment);
```

Параметры:

`attachment`

Объект документа (`GohubAttachment).`

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_attachment_id_w

Запрос ID сопроводительного документа (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_attachment_id_w(
      GohubAttachment* attachment);
```

Параметры:

`attachment`

Объект документа (`GohubAttachment).`

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_attachment_type_code

Запрос кода типа сопроводительного документа.

Объявление:

```
const char* gohub_attachment_type_code(
      GohubAttachment* attachment);
```

Параметры:

`attachment`

Объект документа (`GohubAttachment`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_attachment_type_code_w

Запрос кода типа сопроводительного документа (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_attachment_type_code_w(
      GohubAttachment* attachment);
```

Параметры:

`attachment`

Объект документа (`GohubAttachment`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_attachment_name

Запрос имени сопроводительного документа.

Объявление:

```
const char* gohub_attachment_name(
      GohubAttachment* attachment);
```

Параметры:

`attachment`

Объект документа (`GohubAttachment`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_attachment_name_w

Запрос имени сопроводительного документа (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_attachment_name_w(
      GohubAttachment* attachment);
```

Параметры:

`attachment`

Объект документа (`GohubAttachment`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_attachment_reg_number

Запрос регистрационного номера сопроводительного документа.

Объявление:

```
const char* gohub_attachment_reg_number(
      GohubAttachment* attachment);
```

Параметры:

`attachment`

Объект документа (`GohubAttachment`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_attachment_reg_number_w

Запрос регистрационного номера сопроводительного документа (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_attachment_reg_number_w(
      GohubAttachment* attachment);
```

Параметры:

`attachment`

Объект документа (`GohubAttachment`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_attachment_reg_date

Запрос даты регистрации сопроводительного документа.

Объявление:

```
const char* gohub_attachment_reg_date(
      GohubAttachment* attachment);
```

Параметры:

`attachment`

Объект документа (`GohubAttachment`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_attachment_reg_date_w

Запрос даты регистрации сопроводительного документа (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_attachment_reg_date_w(
      GohubAttachment* attachment);
```

Параметры:

`attachment`

Объект документа (`GohubAttachment`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_attachment_valid_from

Запрос даты начала действия сопроводительного документа.

Объявление:

```
const char* gohub_attachment_valid_from(
      GohubAttachment* attachment);
```

Параметры:

`attachment`

Объект документа (`GohubAttachment`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_attachment_valid_from_w

Запрос даты начала действия сопроводительного документа (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_attachment_valid_from_w(
      GohubAttachment* attachment);
```

Параметры:

`attachment`

Объект документа (`GohubAttachment`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_attachment_valid_to

Запрос даты окончания действия сопроводительного документа.

Объявление:

```
const char* gohub_attachment_valid_to(
      GohubAttachment* attachment);
```

Параметры:

`attachment`

Объект документа (`GohubAttachment`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_attachment_valid_to_w

Запрос даты окончания сопроводительного документа (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_attachment_valid_to_w(
      GohubAttachment* attachment);
```

Параметры:

`attachment`

Объект документа (`GohubAttachment`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_attachment_description

Запрос дескриптора сопроводительного документа. Возвращается текстовая строка состоящая из названия типа документа, его регистрационного номера и даты.

Объявление:

```
const char* gohub_attachment_description(
      GohubAttachment* attachment);
```

Параметры:

`attachment`

Объект документа (`GohubAttachment`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_attachment_description_w

Запрос дескриптора сопроводительного документа. Возвращается текстовая строка состоящая из названия типа документа, его регистрационного номера и даты (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_attachment_description_w(
      GohubAttachment* attachment);
```

Параметры:

`attachment`

Объект документа (`GohubAttachment`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_attachment_count

Запросить количество присоединенных сопроводительных документов к перевозочному документу.

Объявление:

```
int gohub_attachment_count(
      GohubDocument* document);
```

Параметры:

`document`

Объект документа (`GohubDocument`).

Результат:

В случае успеха – не отрицательное число. В случае ошибки – «-1». Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др.`

gohub_attachment_id_by_index

Запросить ID сопроводительного документа по его индексу в списке присоединенных сопроводительных документов к перевозочному документу.

Объявление:

```
const char* gohub_attachment_id_by_index(
      GohubDocument* document,
      int attachmentIndex);
```

Параметры:

`document`

Объект документа (`GohubDocument`);

`attachmentIndex`

индекс документа в списке приложенных к перевозочному документу.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др.`

gohub_attachment_id_by_index_w

Запросить ID сопроводительного документа по его индексу в списке присоединенных сопроводительных документов к перевозочному документу (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_attachment_id_by_index_w(
      GohubDocument* document,
      int attachmentIndex);
```

Параметры:

`document`

Объект документа (`GohubDocument`);

`attachmentIndex`

индекс документа в списке приложенных к перевозочному документу.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др.`

4.4.5. Работа с электронными данными и пакетами предварительного информирования (ПИ)

gohub_load_edata	Загрузить электронные данные ПИ
gohub_load_edata_w	Загрузить электронные данные ПИ (в UTF-16)
gohub_load_edata_simple	Загрузить электронные данные ПИ без сопроводительного документа
gohub_load_edata_simple_w	Загрузить электронные данные ПИ без сопроводительного документа (в UTF-16)
gohub_send_edata	Отправить электронные данные ПИ
gohub_update_edata	Обновить имеющиеся электронные данные ПИ
gohub_query_edata	Запросить электронные данные ПИ
gohub_query_edata_w	Запросить электронные данные ПИ (в UTF-16)
gohub_query_next_edata	Запросить электронные данные ПИ по ревизии
gohub_save_edata	Сохранить электронные данные ПИ
gohub_save_edata_w	Сохранить электронные данные ПИ (в UTF-16)
gohub_edata_load_data	Загрузить данные из файла в имеющиеся электронные данные ПИ
gohub_edata_load_data_w	Загрузить данные из файла в имеющиеся электронные данные ПИ (в UTF-16)
gohub_close_edata	Закрыть электронные данные ПИ
gohub_edata_id	Запросить ID электронных данных ПИ
gohub_edata_id_w	Запросить ID электронных данных ПИ (в UTF-16)
gohub_edata_revision	Запросить ревизию электронных данных ПИ
gohub_edata_revision_date	Запросить дату ревизии электронных данных ПИ
gohub_edata_revision_date_w	Запросить дату ревизии электронных данных ПИ (в UTF-16)
gohub_edata_doc_type	Запросить код типа электронных данных ПИ (190 счет-фактура, 320 упаковочный лист)
gohub_edata_status	Запросить статус электронных данных ПИ
gohub_edata_version	Запросить версию электронных данных ПИ
gohub_edata_version_w	Запросить версию электронных данных ПИ (в UTF-16)
gohub_edata_attachment_id	Запросить ID сопроводительного дукумента электронных данных ПИ
gohub_edata_attachment_id_w	Запросить ID сопроводительного дукумента электронных данных ПИ (в UTF-16)
gohub_query_pi_package	Запросить пакет ПИ
gohub_query_pi_package_w	Запросить пакет ПИ (в UTF-16)
gohub_query_next_pi_package	Запросить пакет ПИ по ревизии
gohub_save_pi_package	Сохранить пакет ПИ
gohub_save_pi_package_w	Сохранить пакет ПИ (в UTF-16)
gohub_close_pi_package	Закрыть пакет ПИ
gohub_pi_package_id	Запросить ID пакета ПИ
gohub_pi_package_id_w	Запросить ID пакета ПИ (в UTF-16)
gohub_pi_package_revision	Запросить ревизию пакета ПИ
gohub_pi_package_revision_date	Запросить дату ревизии пакета ПИ
gohub_pi_package_revision_date_w	Запросить дату ревизии пакета ПИ (в UTF-16)
gohub_pi_package_consignment_id	Запросить ID перевозочного документа на который ссылается пакет ПИ
gohub_pi_package_consignment_id_w	Запросить ID перевозочного документа на который ссылается пакет ПИ (в UTF-16)
gohub_pi_package_status	Запросить статус пакета ПИ
gohub_pi_package_pipacktoed_count	Запросить количество элементов GohubPiPackageToEData содержащиеся в пакете ПИ
gohub_pi_package_pipacktoed	Запросить элемент GohubPiPackageToEData в пакете ПИ по индексу
gohub_pipacktoed_id	Запросить ID объекта GohubPiPackageToEData
gohub_pipacktoed_id_w	Запросить ID объекта GohubPiPackageToEData (в UTF-16)
gohub_pipacktoed_edata_id	Запросить ID электронных данных в объекте GohubPiPackageToEData

gohub_pipacktoed_edata_id_w	Запросить ID электронных данных в объекте GohubPiPackageToEData (в UTF-16)
gohub_pipacktoed_pi_package_id	Запросить ID пакета ПИ в объекте GohubPiPackageToEData
gohub_pipacktoed_pi_package_id_w	Запросить ID пакета ПИ в объекте GohubPiPackageToEData (в UTF-16)
gohub_pipacktoed_note	Запросить примечание в объекте GohubPiPackageToEData
gohub_pipacktoed_note_w	Запросить примечание в объекте GohubPiPackageToEData (в UTF-16)
gohub_pipacktoed_status	Запросить статус объекта GohubPiPackageToEData
gohub_pipacktoed_edata_version	Запросить версию электронных данных в объекте GohubPiPackageToEData
gohub_pipacktoed_edata_version_w	Запросить версию электронных данных в объекте GohubPiPackageToEData (в UTF-16)
gohub_add_edata_to_pi_package	Добавить электронные данные к пакету ПИ
gohub_add_edata_to_pi_package_w	Добавить электронные данные к пакету ПИ (в UTF-16)

gohub_load_edata

Загрузка электронных данных ПИ.

Объявление:

```
GohubEData* gohub_load_edata(
      unsigned int attachmentSmgsTypeCode,
      const char* xmlPath,
      const char* name,
      const char* regNumber,
      const char* regDate,
      const char* validFrom,
      const char* validTo,
      const char* pdfPath);
```

Параметры:

attachmentSmgsTypeCode

Код типа сопроводительного документа согласно информационного руководства СМГС. Допустимые значения: 325 Счет-проформа (Инвойс), 380 (Инвойс) счет-фактура, 935 Счет-фактура;

xmlPath

Путь к xml-файлу содержащий электронные данные ПИ;

name

имя сопроводительного документа;

regNumber

регистрационный номер сопроводительного документа;

regDate

дата регистрации сопроводительного документа (формат даты ‘dd.MM.yyyy’);

validFrom

дата начала срока действия сопроводительного документа (формат даты ‘dd.MM.yyyy’);

validTo

дата окончания срока действия сопроводительного документа (формат даты ‘dd.MM.yyyy’);

`pdfPath`

путь к pdf-файлу с отсканированным текстом сопроводительного документа.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_load_edata_w

Загрузка электронных данных ПИ (в кодировке UTF-16).

Объявление:

```
GohubEData* gohub_load_edata_w(
      unsigned int attachmentSmgsTypeCode,
      const GohubWChar* xmlPath,
      const GohubWChar* name,
      const GohubWChar* regNumber,
      const GohubWChar* regDate,
      const GohubWChar* validFrom,
      const GohubWChar* validTo,
      const GohubWChar* pdfPath);
```

Параметры:

`attachmentSmgsTypeCode`

Код типа сопроводительного документа согласно информационного руководства СМГС. Допустимые значения: 325 Счет-проформа (Инвойс), 380 (Инвойс) счет-фактура, 935 Счет-фактура;

`xmlPath`

Путь к xml-файлу в кодировке UTF-16 содержащий электронные данные ПИ;

`name`

имя сопроводительного документа в кодировке UTF-16;

`regNumber`

регистрационный номер сопроводительного документа в кодировке UTF-16;

`regDate`

дата регистрации сопроводительного документа (формат даты ‘dd.MM.yyyy’) в кодировке UTF-16;

`validFrom`

дата начала срока действия сопроводительного документа (формат даты ‘dd.MM.yyyy’) в кодировке UTF-16;

`validTo`

дата окончания срока действия сопроводительного документа (формат даты ‘dd.MM.yyyy’) в кодировке UTF-16;

`pdfPath`

путь к pdf-файлу с отсканированным текстом сопроводительного документа в кодировке UTF-16.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_load_edata_simple

Загрузить электронные данные ПИ без сопроводительного документа.

Объявление:

```
GohubEData* gohub_load_edata(
      unsigned int attachmentSmgsTypeCode,
      const char* xmlPath
      );
```

Параметры:

`attachmentSmgsTypeCode`

Код типа сопроводительного документа согласно информационного руководства СМГС. Допустимые значения: 325 Счет-проформа (Инвойс), 380 (Инвойс) счет-фактура, 935 Счет-фактура;

`xmlPath`

Путь к xml-файлу содержащий электронные данные ПИ;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_load_edata_simple_w

Загрузить электронные данные ПИ без сопроводительного документа (в кодировке UTF-16).

Объявление:

```
GohubEData* gohub_load_edata_w(
      unsigned int attachmentSmgsTypeCode,
      const GohubWChar* xmlPath);
```

Параметры:

`attachmentSmgsTypeCode`

Код типа сопроводительного документа согласно информационного руководства СМГС. Допустимые значения: 325 Счет-проформа (Инвойс), 380 (Инвойс) счет-фактура, 935 Счет-фактура;

`xmlPath`

Путь к xml-файлу в кодировке UTF-16 содержащий электронные данные ПИ;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_send_edata

Отправление электронных данных ПИ.

Объявление:

```
GohubBool gohub_send_edata(
      GohubConnection* connection,
      GohubEData* eData);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`eData`

объект электронных данных ПИ (`GohubEData)`, который отправляется на сервер.

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др.`

gohub_update_edata

Обновить имеющиеся электронные данные ПИ.

Объявление:

```
GohubBool gohub_update_edata(
      GohubConnection* connection,
      GohubEData* eData);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`eData`

объект электронных данных ПИ (`GohubEData)`, который будет обновляться на сервере.

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др.`

gohub_query_edata

Запрос электронных данных ПИ.

Объявление:

```
GohubEData* gohub_query_edata(
      GohubConnection* connection,
      const char* eDataId);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`eDataId`

уникальный идентификатор электронных данных ПИ.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_query_edata_w

Запрос электронных данных ПИ (в кодировке UTF-16).

Объявление:

```
GohubEData* gohub_query_edata_w(
      GohubConnection* connection,
      const GohubWChar* eDataId);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`eDataId`

уникальный идентификатор электронных данных ПИ (в кодировке UTF-16).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_query_edata_for_attachment

Запрос электронных данных ПИ по идентификатору сопроводительного документа.

Объявление:

```
GohubEData* gohub_query_edata(
      GohubConnection* connection,
      const char* attachmentId);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`attachmentId`

уникальный идентификатор сопроводительного документа.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_query_edata_for_attachment_w

Запрос электронных данных ПИ (в кодировке UTF-16) по идентификатору сопроводительного документа.

Объявление:

```
GohubEData* gohub_query_edata_w(
      GohubConnection* connection,
      const GohubWChar* attachmentId);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`attachmentId`

уникальный идентификатор сопроводительного документа (в кодировке UTF-16).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_query_next_edata

Запрос электронных данных ПИ по ревизии.

Объявление:

```
GohubEData* gohub_query_next_edata(
      GohubConnection* connection,
      unsigned __int64 lastRevision);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`lastRevision`

ревизия, с которой начинать поиск электронных данных ПИ.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_save_edata

Сохранение электронных данных ПИ.

Объявление:

```
GohubBool gohub_save_edata(
      GohubEData* eData,
      const char* path);
```

Параметры:

`eData`

Объект электронных данных ПИ (`GohubEData)`, который сохраняется на диск;

`path`

путь, куда сохраняется файл.

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_save_edata_w

Сохранение электронных данных ПИ (в кодировке UTF-16).

Объявление:

```
GohubBool gohub_save_edata_w(
      GohubEData* eData,
      const GohubWChar* path);
```

Параметры:

`eData`

Объект электронных данных ПИ (`GohubEData`), который сохраняется на диск;

`path`

путь, куда сохраняется файл (в кодировке UTF-16).

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_edata_load_data

Загрузить данные из файла в имеющиеся электронные данные ПИ.

Объявление:

```
GohubBool gohub_save_edata(
      GohubEData* eData,
      const char* path);
```

Параметры:

`eData`

Объект электронных данных ПИ (`GohubEData`), который сохраняется на диск;

`path`

путь, куда сохраняется файл.

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_edata_load_data_w

Загрузить данные из файла в имеющиеся электронные данные ПИ (в кодировке UTF-16).

Объявление:

```
GohubBool gohub_save_edata_w(
      GohubEData* eData,
      const GohubWChar* path);
```

Параметры:

`eData`

Объект электронных данных ПИ (`GohubEData`), который сохраняется на диск;

`path`

путь, куда сохраняется файл (в кодировке UTF-16).

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_close_edata

Закрытие электронных данных ПИ.

Объявление:

```
GohubBool gohub_close_edata(
      GohubEData* eData);
```

Параметры:

`eData`

Объект электронных данных ПИ (`GohubEData`).

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др`.

gohub_edata_id

Запрос ID электронных данных ПИ.

Объявление:

```
const char* gohub_edata_id(
      GohubEData* eData);
```

Параметры:

`eData`

Объект электронных данных ПИ (`GohubEData`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др`.

gohub_edata_id_w

Запрос ID электронных данных ПИ (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_edata_id_w(
      GohubEData* eData);
```

Параметры:

`eData`

Объект электронных данных ПИ (`GohubEData`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др`.

gohub_edata_revision

Запрос номера ревизии электронных данных ПИ.

Объявление:

```
unsigned __int64 gohub_edata_revision(
      GohubEData* eData);
```

Параметры:

`eData`

Объект электронных данных ПИ (`GohubEData`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_edata_revision_date

Запрос даты ревизии электронных данных ПИ.

Объявление:

```
const char* gohub_edata_revision_date(
      GohubEData* eData);
```

Параметры:

`eData`

Объект электронных данных ПИ (`GohubEData`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_edata_revision_date_w

Запрос даты ревизии электронных данных ПИ (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_edata_revision_date_w(
      GohubEData* eData);
```

Параметры:

`eData`

Объект электронных данных ПИ (`GohubEData`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_edata_doc_type

Запрос типа электронных данных ПИ.

Объявление:

```
unsigned int gohub_edata_doc_type(
      GohubEData* eData);
```

Параметры:

`eData`

Объект электронных данных ПИ (`GohubEData`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_edata_status

Запрос статуса электронных данных ПИ.

Объявление:

```
int gohub_edata_status(
      GohubEData* eData);
```

Параметры:

`eData`

Объект электронных данных ПИ (`GohubEData`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_edata_version

Запрос версии электронных данных ПИ.

Объявление:

```
const char* gohub_edata_version(
      GohubEData* eData);
```

Параметры:

`eData`

Объект электронных данных ПИ (`GohubEData`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_edata_version_w

Запрос версии электронных данных ПИ (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_edata_version_w(
      GohubEData* eData);
```

Параметры:

`eData`

Объект электронных данных ПИ (`GohubEData`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_edata_attachment_id

Запрос ID сопроводительного документа электронных данных ПИ.

Объявление:

```
const char* gohub_edata_attachment_id(
      GohubEData* eData);
```

Параметры:

`eData`

Объект электронных данных ПИ (`GohubEData`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_edata_attachment_id_w

Запрос ID сопроводительного документа электронных данных ПИ (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_edata_attachment_id_w(
      GohubEData* eData);
```

Параметры:

`eData`

Объект электронных данных ПИ (`GohubEData`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_query_pi_package

Запрос пакет ПИ.

Объявление:

```
GohubPiPackage* gohub_query_pi_package(
      GohubConnection* connection,
      const char* piPackageId);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`piPackageId`

уникальный идентификатор электронных данных ПИ.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_query_pi_package_w

Запрос пакет ПИ (в кодировке UTF-16).

Объявление:

```
GohubPiPackage* gohub_query_pi_package_w(
      GohubConnection* connection,
      const GohubWChar* piPackageId);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`piPackageId`

уникальный идентификатор сопроводительного документа (в кодировке UTF-16).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_query_next_pi_package

Запрос пакета ПИ по ревизии.

Объявление:

```
GohubPiPackage* gohub_query_next_pi_package(
      GohubConnection* connection,
      unsigned __int64 lastRevision);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`lastRevision`

ревизия, с которой начинать поиск пакета ПИ.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_save_pi_package

Сохранение пакета ПИ.

Объявление:

```
GohubBool gohub_save_pi_package(
      GohubPiPackage* piPackage,
      const char* path);
```

Параметры:

`piPackage`

Объект пакета ПИ (`GohubPiPackage),` который сохраняется на диск;

`path`

путь, куда сохраняется файл.

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_save_pi_package_w

Сохранение пакета ПИ (в кодировке UTF-16).

Объявление:

```
GohubBool gohub_save_pi_package_w(
      GohubPiPackage* piPackage,
      const GohubWChar* path);
```

Параметры:

`piPackage`

Объект пакета ПИ (`GohubPiPackage),` который сохраняется на диск;

`path`

путь, куда сохраняется файл (в кодировке UTF-16).

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_close_pi_package

Закрытие пакета ПИ.

Объявление:

```
GohubBool gohub_close_pi_package(
      GohubPiPackage* piPackage);
```

Параметры:

`piPackage`

Объект пакета ПИ (`GohubPiPackage`).

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_pi_package_id

Запрос ID пакета ПИ.

Объявление:

```
const char* gohub_pi_package_id(
      GohubPiPackage* piPackage);
```

Параметры:

`piPackage`

Объект пакета ПИ (`GohubPiPackage`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_pi_package_id_w

Запрос ID пакета ПИ (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_pi_package_id_w(
      GohubPiPackage* piPackage);
```

Параметры:

`piPackage`

Объект пакета ПИ (`GohubPiPackage`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_pi_package_revision

Запрос номера ревизии пакета ПИ.

Объявление:

```
unsigned __int64 gohub_pi_package_revision(
      GohubPiPackage* piPackage);
```

Параметры:

`piPackage`

Объект пакета ПИ (`GohubPiPackage`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др`.

gohub_pi_package_revision_date

Запрос даты ревизии пакета ПИ.

Объявление:

```
const char* gohub_pi_package_revision_date(
      GohubPiPackage* piPackage);
```

Параметры:

`piPackage`

Объект пакета ПИ (`GohubPiPackage`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др`.

gohub_pi_package_revision_date_w

Запрос даты ревизии пакета ПИ (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_pi_package_revision_date_w(
      GohubPiPackage* piPackage);
```

Параметры:

`piPackage`

Объект пакета ПИ (`GohubPiPackage`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др`.

gohub_pi_package_status

Запрос статуса электронных данных ПИ.

Объявление:

```
int gohub_pi_package_status(
      GohubPiPackage* piPackage);
```

Параметры:

`eData`

Объект электронных данных ПИ (`GohubEData`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_pi_package_consignment_id

Запросить ID перевозочного документа на который ссылается пакет ПИ.

Объявление:

```
const char* gohub_pi_package_consignment_id(
      GohubPiPackage* piPackage);
```

Параметры:

`piPackage`

Объект пакета ПИ (`GohubPiPackage`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_pi_package_consignment_id_w

Запросить ID перевозочного документа на который ссылается пакет ПИ (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_pi_package_consignment_id_w(
      GohubPiPackage* piPackage);
```

Параметры:

`piPackage`

Объект пакета ПИ (`GohubPiPackage`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_pi_package_pipacktoed_count

Запросить количество элементов GohubPiPackageToEData содержащиеся в пакете ПИ.

Объявление:

```
int gohub_pi_package_pipacktoed_count(
      GohubPiPackage* piPackage);
```

Параметры:

`piPackage`

Объект пакета ПИ (`GohubPiPackage`).

Результат:

Количество элементов GohubPiPackageToEData содержащиеся в пакете ПИ.

gohub_pi_package_pipacktoed

Запрос элемента GohubPiPackageToEData в пакете ПИ по индексу.

Объявление:

```
GohubPiPackageToEData* gohub_pi_package_pipacktoed(
      GohubPiPackage* piPackage,
      int index);
```

Параметры:

`piPackage`

Объект пакета ПИ (`GohubPiPackage`).

`index`

Номер индекса. Допустимые значения от 0 до gohub_pi_package_pipacktoed_count()-1.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_pipacktoed_id

Запросить ID объекта GohubPiPackageToEData.

Объявление:

```
const char* gohub_pipacktoed_id(
      GohubPiPackageToEData* piPackageToEData);
```

Параметры:

`piPackageToEData`

Объект `GohubPiPackageToEData`.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_pipacktoed_id_w

Запросить ID объекта GohubPiPackageToEData (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_pipacktoed_id_w(
      GohubPiPackageToEData* piPackageToEData);
```

Параметры:

`piPackageToEData`

Объект `GohubPiPackageToEData`.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_pipacktoed_edata_id

Запросить ID электронных данных в объекте GohubPiPackageToEData.

Объявление:

```
const char* gohub_pipacktoed_edata_id(
      GohubPiPackageToEData* piPackageToEData);
```

Параметры:

`piPackageToEData`

Объект `GohubPiPackageToEData`.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_pipacktoed_edata_id_w

Запросить ID электронных данных в объекте GohubPiPackageToEData (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_pipacktoed_edata_id_w(
      GohubPiPackageToEData* piPackageToEData);
```

Параметры:

`piPackageToEData`

Объект `GohubPiPackageToEData`.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_pipacktoed_pi_package_id

Запросить ID пакета ПИ в объекте GohubPiPackageToEData.

Объявление:

```
const char* gohub_pipacktoed_pi_package_id(
      GohubPiPackageToEData* piPackageToEData);
```

Параметры:

`piPackageToEData`

Объект `GohubPiPackageToEData`.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_pipacktoed_pi_package_id_w

Запросить ID пакета ПИ в объекте GohubPiPackageToEData (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_pipacktoed_pi_package_id_w(
      GohubPiPackageToEData* piPackageToEData);
```

Параметры:

`piPackageToEData`

Объект `GohubPiPackageToEData`.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_pipacktoed_note

Запросить примечание в объекте GohubPiPackageToEData.

Объявление:

```
const char* gohub_pipacktoed_note(
      GohubPiPackageToEData* piPackageToEData);
```

Параметры:

`piPackageToEData`

Объект `GohubPiPackageToEData`.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_pipacktoed_note_w

Запросить примечание в объекте GohubPiPackageToEData (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_pipacktoed_note_w(
      GohubPiPackageToEData* piPackageToEData);
```

Параметры:

`piPackageToEData`

Объект `GohubPiPackageToEData`.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_pipacktoed_edata_version

Запросить версию электронных данных в объекте GohubPiPackageToEData.

Объявление:

```
const char* gohub_pipacktoed_edata_version(
      GohubPiPackageToEData* piPackageToEData);
```

Параметры:

`piPackageToEData`

Объект `GohubPiPackageToEData`.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_pipacktoed_edata_version_w

Запросить версию электронных данных в объекте GohubPiPackageToEData (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_pipacktoed_edata_version_w(
      GohubPiPackageToEData* piPackageToEData);
```

Параметры:

`piPackageToEData`

Объект `GohubPiPackageToEData`.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_add_edata_to_pi_package

Добавить электронные данные к пакету ПИ.

Объявление:

```
GohubPiPackage* gohub_add_edata_to_pi_package(
      GohubConnection* connection,
      GohubEData* eData,
      const char* piPackageId);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`eData`

Объект электронных данных ПИ (`GohubEData).`

`piPackageId`

уникальный идентификатор электронных данных ПИ.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_add_edata_to_pi_package_w

Добавить электронные данные к пакету ПИ (в кодировке UTF-16).

Объявление:

```
GohubPiPackage* gohub_add_edata_to_pi_package_w(
      GohubConnection* connection,
      GohubEData* eData,
      const GohubWChar* piPackageId);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`eData`

Объект электронных данных ПИ (`GohubEData).`

`piPackageId`

уникальный идентификатор сопроводительного документа (в кодировке UTF-16).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

4.4.6. Работа с накопительными карточками ФДУ-92

gohub_query_fdu92	Запросить накопительную карточку
gohub_query_fdu92_w	Запросить накопительную карточку `(в UTF-16)`

gohub_query_fdu92_by_number	Запросить накопительную карточку за номером станции и номером накопительной карточки
gohub_query_fdu92_by_number_w	Запросить накопительную карточку за номером станции и номером накопительной карточки (в UTF-16)
gohub_query_next_fdu92	Запросить следующую накопительную карточку
gohub_create_fdu92	Создать накопительную карточку
gohub_create_fdu92_w	Создать накопительную карточку (в UTF-16)
gohub_load_fdu92	Загрузить накопительную карточку из файла xml
gohub_load_fdu92_w	Загрузить накопительную карточку из файла xml (UTF-16)
gohub_save_fdu92	Сохранить накопительную карточку
gohub_save_fdu92 _w	Сохранить накопительную карточку (в UTF-16)
gohub_send_fdu92	Отправить накопительную карточку
gohub_fdu92_id	Запросить ID накопительной карточки
gohub_fdu92_id_w	Запросить ID накопительной карточки (в UTF-16)
gohub_fdu92_status	Запросить статус накопительной карточки
fdu92_revision	Запросить номер ревизии накопительной карточки
gohub_fdu92_text	Запросить текст накопительной карточки
gohub_fdu92_text_w	Запросить текст накопительной карточки (в UTF-16)
fdu92_size	Запросить предполагаемый размер накопительной карточки представленный в текущей кодовой странице
gohub_fdu92_signer_info	Запросить информацию о подписанте
gohub_fdu92_signer_info_w	Запросить информацию о подписанте (в UTF-16)
gohub_fdu92_sign_time	Запросить информацию о времени подписания
gohub_fdu92_sign_time_w	Запросить информацию о времени подписания (в UTF-16)
gohub_fdu92_signer_name_w	Запросить имя подписанта (в UTF-16)
gohub_fdu92_has_signature	Проверить подписан ли документ
gohub_save_fdu92	Сохранить накопительную карточку в файл
gohub_save_fdu92_w	Сохранить накопительную карточку в файл (в UTF-16)
gohub_reject_fdu92	Отозвать накопительную карточку
gohub_reject_fdu92_w	Отозвать накопительную карточку (в UTF-16)
gohub_close_fdu92	Закрыть накопительную карточку
gohub_query_and_save_fdu92_printable_form	Запросить печатную форму накопительной карточки
gohub_query_and_save_fdu92_printable_form_w	Запросить печатную форму накопительной карточки (в UTF-16)

gohub_query_fdu92

Запросить накопительную карточку ФДУ-92.

Объявление:

```
GohubFdu92* gohub_query_fdu92(
GohubConnection* connection,
const char* fdu92Id);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`fdu92Id`

уникальный идентификатор накопительной карточки.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_query_fdu92_w

Запросить накопительную карточку ФДУ-92.

Объявление:

```
GohubFdu92* gohub_query_fdu92_w(
GohubConnection* connection,
const GohubWChar* fdu92Id);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`fdu92Id`

уникальный идентификатор накопительной карточки (в кодировке UTF-16).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_query_fdu92_by_number

Запросить накопительную карточку ФДУ-92.

Объявление:

```
GohubFdu92* gohub_query_fdu92(
GohubConnection* connection,
const char* registration_esr, const char* registration_num);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`registration_esr`

Номер станции.

`registration_num`

Номер накопительной карточки.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_query_fdu92_by_number_w

Запросить накопительную карточку ФДУ-92.

Объявление:

```
GohubFdu92* gohub_query_fdu92_w(
```

```
GohubConnection* connection,
const GohubWChar* registration_esr,
const GohubWChar* registration_num);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`registration_esr`

Номер станции (в кодировке UTF-16).

`registration_num`

`Номер накопительной карточки` (в кодировке UTF-16).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_create_fdu92

Создать накопительную карточку ФДУ-92.

Объявление:

```
GohubFdu92* gohub_create_fdu92(
const char* content);
```

Параметры:

`content`

строка с xml-структурой с содержанием документа.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_create_fdu92_w

Создать накопительную карточку ФДУ-92.

Объявление:

```
GohubFdu92* gohub_create_fdu92_w(
const GohubWChar* content);
```

Параметры:

`content`

строка с xml-структурой с содержанием документа в кодировке UTF-16;.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции gohub_last_error и др.

gohub_load_fdu92

Загрузить накопительную карточку ФДУ-92 из файла xml.

Объявление:

```
GohubFdu92* gohub_load_fdu92(
const char* id,
const char* path);
```

Параметры:

`id`

идентификатор карточки ФДУ-92 на сервере.

`path`

путь к файлу.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_load_fdu92_w

Загрузить накопительную карточку ФДУ-92 из файла xml.

Объявление:

```
GohubFdu92* gohub_load_fdu92_w(
const GohubWChar* id,
const GohubWChar* path);
```

Параметры:

`id`

идентификатор карточки ФДУ-92 на сервере.

`path`

путь к файлу в кодировке UTF-16.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_send_fdu92

Отправить накопительную карточку ФДУ-92.

Объявление:

```
GohubBool gohub_send_fdu92(
GohubConnection* connection,
GohubFdu92* fdu92);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect`

`fdu92`

Объект документа (`GohubFdu92`).;

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_query_next_fdu92

Запросить следующую накопительную карточку ФДУ-92.

Объявление:

```
GohubFdu92* gohub_query_next_fdu92(
GohubConnection* connection,
int lastRevision);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`lastRevision`

ревизия документа, от которой начинается поиск следующей накопительной карточки.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_fdu92_id

Запрос ID накопительной карточки ФДУ-92.

Объявление:

```
const char* gohub_fdu92_id(
GohubFdu92* fdu92);
```

Параметры:

`fdu92`

Объект документа (`GohubFdu92`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_fdu92_id_w

Запрос ID накопительной карточки ФДУ-92 (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_fdu92_id_w(
GohubFdu92* fdu92);
```

Параметры:

`fdu92`

Объект документа (`GohubFdu92`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_fdu92_status

Запрос статуса накопительной карточки ФДУ-92.

Объявление:

```
GohubFdu92Status gohub_fdu92_status(
GohubFdu92* fdu92);
```

Параметры:

`fdu92`

Объект документа (GohubFdu92), статус которого запрашивается;

Результат:

В случае успеха – значение из перечисления GohubFdu92Status. В случае ошибки – «-1». Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_fdu92_revision

Запрос номера ревизии накопительной карточки ФДУ-92.

Объявление:

```
int gohub_fdu92_revision(
GohubFdu92* fdu92);
```

Параметры:

connection

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции gohub_connect;

fdu92

объект документа (GohubFdu92).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_fdu92_text

Запрос текста накопительной карточки ФДУ-92.

Объявление:

```
const char* gohub_fdu92_text(
GohubFdu92* fdu92);
```

Параметры:

fdu92

Объект документа (GohubFdu92).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_fdu92_text_w

Запрос текста накопительной карточки ФДУ-92 (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_fdu92_text_w(
GohubFdu92* fdu92);
```

Параметры:

fdu92

Объект документа (GohubFdu92).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_fdu92_size

Запрос предполагаемого размера файла, который получится при сохранении документа в текущей кодировке (GohubFdu92).

Объявление:

```
int gohub_fdu92_size(
GohubFdu92* fdu92);
```

Параметры:

fdu92

Объект документа (GohubFdu92), текст которого запрашивается;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_fdu92_signer_info

Запрос информации о подписанта накопительной карточки ФДУ-92.

Объявление:

```
const char* gohub_fdu92_signer_info(
GohubFdu92* fdu92);
```

Параметры:

fdu92

Объект документа (GohubFdu92).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_fdu92_signer_info_w

Запрос информации о подписанта накопительной карточки ФДУ-92 (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_fdu92_signer_info_w(
GohubFdu92* fdu92);
```

Параметры:

fdu92

Объект документа (GohubFdu92).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_fdu92_sign_time

Запрос информации о подписанта накопительной карточки ФДУ-92.

Объявление:

```
const char* gohub_fdu92_sign_time(
```

```
GohubFdu92* fdu92);
```

Параметры:

`fdu92`

Объект документа (`GohubFdu92`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_fdu92_sign_time_w

Запрос информации о подписанта накопительной карточки ФДУ-92 (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_fdu92_sign_time_w(
GohubFdu92* fdu92);
```

Параметры:

`fdu92`

Объект документа (`GohubFdu92`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_fdu92_sign_name_w

Запрос имя подписанта накопительной карточки ФДУ-92 (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_fdu92_sign_name_w(
GohubFdu92* fdu92);
```

Параметры:

`fdu92`

Объект документа (`GohubFdu92`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_fdu92_has_signature

Проверить подписан ли документ.

Объявление:

```
const GohubWChar* gohub_fdu92_has_signature(
GohubFdu92* fdu92);
```

Параметры:

`fdu92`

Объект документа (`GohubFdu92`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_save_fdu92

Сохранение накопительной карточки ФДУ-92.

Объявление:

```
GohubBool gohub_save_fdu92(
GohubFdu92* fdu92,
const char* path,
int codePage);
```

Параметры:

`fdu92`

Объект документа (`GohubFdu92`), который сохраняется на диск;

`path`

путь, куда сохраняется файл;

`codePage`

Числовое обозначение страницы кодировки, в которой сохраняется файл.

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_save_fdu92_w

Сохранение накопительной карточки ФДУ-92.

Объявление:

```
GohubBool gohub_save_fdu92_w(
GohubFdu92* fdu92,
const GohubWChar* path,
int codePage);
```

Параметры:

`fdu92`

Объект документа (`GohubFdu92`), который сохраняется на диск;

`path`

путь, куда сохраняется файл (в кодировке UTF-16);

`codePage`

Числовое обозначение страницы кодировки, в которой сохраняется файл.

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_reject_fdu92

Отозвать накопительную карточку ФДУ-92.

Объявление:

```
GohubBool gohub_reject_fdu92(
GohubConnection* connection,
const char* fdu92Id);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`fdu92Id`

уникальный идентификатор накопительной карточки .

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_reject_fdu92_w

Отозвать накопительную карточку ФДУ-92.

Объявление:

```
GohubBool gohub_reject_fdu92_w(
GohubConnection* connection,
const GohubWChar* fdu92Id);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`fdu92Id`

уникальный идентификатор накопительной карточки (в кодировке UTF-16).

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_close_fdu92

Закрытие документа (`GohubFdu92).`

Объявление:

```
GohubBool gohub_close_fdu92(
GohubFdu92* fdu92);
```

Параметры:

`fdu92`

Объект документа (`GohubFdu92).`

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_query_and_save_fdu92_printable_form

Запросить печатную форму накопительной карточки ФДУ-92 по его ID и сохранить ее в файл.

Объявление:

```
GohubBool gohub_query_and_save_fdu92_printable_form(
```

```
        GohubConnection* connection,
        const char* fdu92Id,
        const char* path);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect;

`fdu92Id`

уникальный идентификатор накопительной карточки .

`path`

путь, по которому сохранить запрошенную печатную форму.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_query_and_save_fdu92_printable_form_w

Запросить печатную форму накопительной карточки ФДУ-92 по его ID (в кодировке UTF-16) и сохранить ее в файл.

Объявление:

```
GohubBool gohub_query_and_save_fdu92_printable_form_w(
        GohubConnection* connection,
        const GohubWChar* fdu92Id,
        const GohubWChar* path);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect;

`fdu92Id`

уникальный идентификатор накопительной карточки (в кодировке UTF-16);

`path`

путь, по которому сохранить запрошенную печатную форму.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

4.4.7. Работа с ведомостями о пользовании вагонами/контейнерами ГУ-46

gohub_query_gu46	Запросить ведомость о пользовании вагонами/контейнерами
gohub_query_gu46_w	Запросить ведомость о пользовании вагонами/контейнерами (в UTF-16)
gohub_query_next_gu46	Запросить следующую ведомость о пользовании вагонами/контейнерами

gohub_create_gu46	Создать ведомость о пользовании вагонами/контейнерами
gohub_create_gu46_w	Создать ведомость о пользовании вагонами/контейнерами
gohub_load_ gu46	Загрузить ведомость о пользовании вагонами/контейнерами из файла xml
gohub_load_ gu46_w	Загрузить ведомость о пользовании вагонами/контейнерами из файла xml (UTF-16)
gohub_send_gu46	Отправить ведомость о пользовании вагонами/контейнерами
gohub_gu46_id	Запросить ID ведомости о пользовании вагонами/контейнерами
gohub_gu46_id_w	Запросить ID ведомости о пользовании вагонами/контейнерами (в UTF-16)
gohub_gu46_status	Запросить статус ведомости о пользовании вагонами/контейнерами
gohub_gu46_revision	Запросить номер ревизии ведомости о пользовании вагонами/контейнерами
gohub_gu46_text	Запросить текст ведомости о пользовании вагонами/контейнерами
gohub_gu46_text_w	Запросить текст ведомости о пользовании вагонами/контейнерами (в UTF-16)
gohub_gu46_size	Запросить предполагаемый размер ведомости о пользовании вагонами/контейнерами представленный в текущей кодовой странице
gohub_gu46_signer_info	Запросить информацию о подписанте ведомости о пользовании вагонами/контейнерами
gohub_gu46_signer_info_w	Запросить информацию о подписанте ведомости о пользовании вагонами/контейнерами (в UTF-16)
gohub_gu46_sign_time	Запросить время подписания ведомости о пользовании вагонами/контейнерами
gohub_gu46_sign_time_w	Запросить время подписания ведомости о пользовании вагонами/контейнерами (в UTF-16)
gohub_gu46_signer_name_w	Запросить имя подписанта (в UTF-16)
gohub_gu46_has_signature	Проверить подписан ли документ
gohub_save_gu46	Сохранить ведомость о пользовании вагонами/контейнерами
gohub_save_gu46_w	Сохранить ведомость о пользовании вагонами/контейнерами (в UTF-16)
gohub_reject_gu46	Отозвать ведомость о пользовании вагонами/контейнерами
gohub_reject_gu46_w	Отозвать ведомость о пользовании вагонами/контейнерами (в UTF-16)
gohub_close_gu46	Закрыть ведомость о пользовании вагонами/контейнерами
gohub_query_gu46_by_number	Запросить ведомость о пользовании вагонами/контейнерами за номером станции и номером ведомости
gohub_query_gu46_by_number_w	Запросить ведомость о пользовании вагонами/контейнерами контейнерами за номером

	станции и номером ведомости (в UTF-16)
gohub_query_and_save_gu46_printable_form	Запросить печатную форму ведомости о пользовании вагонами/контейнерами
gohub_query_and_save_gu46_printable_form_w	Запросить печатную форму ведомости о пользовании вагонами/контейнерами (в UTF-16)

gohub_query_gu46

Запросить ведомость о пользовании вагонами/контейнерами ГУ-46.

Объявление:

```
GohubGu46* gohub_query_gu46(
GohubConnection* connection,
const char* gu46Id);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`gu46Id`

уникальный идентификатор ведомости о пользовании вагонами/контейнерами.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_query_gu46_w

Запросить ведомость о пользовании вагонами/контейнерами ГУ-46.

Объявление:

```
GohubGu46* gohub_query_gu46_w(
GohubConnection* connection,
const GohubWChar* gu46Id);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`gu46Id`

уникальный идентификатор ведомости о пользовании вагонами/контейнерами (в кодировке UTF-16).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_query_next_gu46

Запросить следующую ведомость о пользовании вагонами/контейнерами ГУ-46.

Объявление:

```
GohubGu46* gohub_query_next_gu46(
GohubConnection* connection,
```

```
int lastRevision);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`lastRevision`

ревизия документа, от которой начинается поиск следующей ведомости о пользовании вагонами/контейнерами.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_create_gu46

Создать ведомость о пользовании вагонами/контейнерами ГУ-46.

Объявление:

```
GohubGu46* gohub_create_gu46(
const char* gu46Id,
const char* content);
```

Параметры:

`gu46Id`

уникальный идентификатор ведомости о пользовании вагонами/контейнерами;

`content`

строка с xml-структурой с содержанием документа.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_create_gu46_w

Создать ведомость о пользовании вагонами/контейнерами ГУ-46.

Объявление:

```
GohubGu46* gohub_create_gu46_w(
const GohubWChar* gu46Id,
const GohubWChar* content);
```

Параметры:

`gu46Id`

уникальный идентификатор ведомости о пользовании вагонами/контейнерами (в кодировке UTF-16);

`content`

строка с xml-структурой с содержанием документа.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_load_gu46

Загрузить ведомость о пользовании вагонами/контейнерами ГУ-46 из файла xml.

Объявление:

```
GohubGu46* gohub_load_gu46(
const char* id,
const char* path);
```

Параметры:

`id`

идентификатор ведомости ГУ-46 на сервере.

`path`

путь к файлу.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_load_gu46_w

Загрузить ведомость о пользовании вагонами/контейнерами ГУ-46 из файла xml.

Объявление:

```
GohubGu46* gohub_load_gu46_w(
const GohubWChar* id,
const GohubWChar* path);
```

Параметры:

`id`

идентификатор ведомости ГУ-46 на сервере.

`path`

путь к файлу в кодировке UTF-16.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_send_gu46

Отправить ведомость о пользовании вагонами/контейнерами ГУ-46.

Объявление:

```
GohubBool gohub_send_gu46(
GohubConnection* connection,
GohubGu46* gu46);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`gu46`

Объект документа (`GohubGu46`).

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_gu46_id

Запрос ID ведомости о пользовании вагонами/контейнерами ГУ-46.

Объявление:

```
const char* gohub_gu46_id(
GohubGu46* gu46);
```

Параметры:

gu46

Объект документа (GohubGu46).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_gu46_id_w

Запрос ID ведомости о пользовании вагонами/контейнерами ГУ-46.

Объявление:

```
const GohubWChar* gohub_gu46_id_w(
GohubGu46* gu46);
```

Параметры:

gu46

Объект документа (GohubGu46).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_gu46_status

Запрос статуса ведомости о пользовании вагонами/контейнерами ГУ-46.

Объявление:

```
GohubGu46Status gohub_gu46_status(
GohubGu46* gu46);
```

Параметры:

gu46

Объект документа (GohubGu46), статус которого запрашивается;

Результат:

В случае успеха – значение из перечисления GohubGu46Status. В случае ошибки – «-1». Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_gu46_revision

Запрос номера ревизии ведомости о пользовании вагонами/контейнерами ГУ-46.

Объявление:

```
int gohub_gu46_revision(
```

```
GohubGu46* gu46);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`gu46`

объект документа (`GohubGu46).`

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_gu46_text

Запрос текста ведомости о пользовании вагонами/контейнерами ГУ-46.

Объявление:

```
const char* gohub_gu46_text(
GohubGu46* gu46);
```

Параметры:

`gu46`

Объект документа (`GohubGu46).`

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_gu46_text_w

Запрос текста ведомости о пользовании вагонами/контейнерами ГУ-46.

Объявление:

```
const GohubWChar* gohub_gu46_text_w(
GohubGu46* gu46);
```

Параметры:

`gu46`

Объект документа (`GohubGu46).`

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_gu46_size

Запрос предполагаемого размера файла, который получится при сохранении документа в текущей кодировке (`GohubGu46).`

Объявление:

```
int gohub_gu46_size(
GohubGu46* gu46);
```

Параметры:

gu46

Объект документа (GohubGu46), текст которого запрашивается;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_gu46_signer_info

Запрос информации о подписанте ведомости о пользовании вагонами/контейнерами ГУ-46.

Объявление:

```
const char* gohub_gu46_signer_info(
GohubGu46* gu46);
```

Параметры:

gu46

Объект документа (GohubGu46).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_gu46_signer_info_w

Запрос информации о подписанте ведомости о пользовании вагонами/контейнерами ГУ-46.

Объявление:

```
const GohubWChar* gohub_gu46_signer_info_w(
GohubGu46* gu46);
```

Параметры:

gu46

Объект документа (GohubGu46).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_gu46_sign_time

Запрос времени подписания ведомости о пользовании вагонами/контейнерами ГУ-46.

Объявление:

```
const char* gohub_gu46_sign_time(
GohubGu46* gu46);
```

Параметры:

gu46

Объект документа (GohubGu46).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_gu46_sign_time_w

Запрос времени подписания ведомости о пользовании вагонами/контейнерами ГУ-46.

Объявление:

```
const GohubWChar* gohub_gu46_sign_time_w(
GohubGu46* gu46);
```

Параметры:

`gu46`

Объект документа (`GohubGu46`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др.`

gohub_gu46_sign_name_w

Запрос имя подписанта ведомости о пользовании вагонами/контейнерами ГУ-46 (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_gu46_sign_name_w(
GohubGu46* gu46);
```

Параметры:

`gu46`

Объект документа (`GohubGu46`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др.`

gohub_gu46_has_signature

Проверить подписан ли документ.

Объявление:

```
const GohubWChar* gohub_gu46_has_signature(
GohubGu46* gu46);
```

Параметры:

`gu46`

Объект документа (`GohubGu46`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др.`

gohub_save_gu46

Сохранение ведомости о пользовании вагонами/контейнерами ГУ-46.

Объявление:

```
GohubBool gohub_save_gu46(
GohubGu46* gu46,
const char* path,
int codePage);
```

Параметры:

gu46

Объект документа (GohubGu46), который сохраняется на диск;

path

путь, куда сохраняется файл;

codePage

Числовое обозначение страницы кодировки, в которой сохраняется файл.

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_save_gu46_w

Сохранение ведомости о пользовании вагонами/контейнерами ГУ-46.

Объявление:

```
GohubBool gohub_save_gu46_w(
GohubGu46* gu46,
const GohubWChar* path,
int codePage);
```

Параметры:

gu46

Объект документа (GohubGu46), который сохраняется на диск;

path

путь, куда сохраняется файл (в кодировке UTF-16);

codePage

Числовое обозначение страницы кодировки, в которой сохраняется файл.

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_reject_gu46

Отозвать ведомость о пользовании вагонами/контейнерами ГУ-46.

Объявление:

```
GohubBool gohub_reject_gu46(
GohubConnection* connection,
const char* gu46Id);
```

Параметры:

connection

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции gohub_connect;

gu46Id

уникальный идентификатор ведомости о пользовании вагонами/контейнерами.

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_reject_gu46_w

Отозвать ведомость о пользовании вагонами/контейнерами ГУ-46.

Объявление:

```
GohubBool gohub_reject_gu46_w(
GohubConnection* connection,
const GohubWChar* gu46Id);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`gu46Id`

уникальный идентификатор ведомости о пользовании вагонами/контейнерами (в кодировке UTF-16).

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_close_gu46

Закрытие документа (`GohubGu46).`

Объявление:

```
GohubBool gohub_close_gu46(
GohubGu46* gu46);
```

Параметры:

`gu46`

Объект документа (`GohubGu46).`

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_query_gu46_by_number

Запросить ведомость о пользовании вагонами/контейнерами ГУ-46 за номером станции и номером ведомости.

Объявление:

```
GohubGu46* gohub_query_gu46_by_number(GohubConnection* connection,
const char* registration_esr, const char* registration_num);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

registration_esr

Станция формирования ведомости.

registration_num

Номер ведомости..

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_query_gu46_by_number_w

Запросить ведомость о пользовании вагонами/контейнерами ГУ-46 за номером станции и номером ведомости (в кодировке UTF-16)

Объявление:

```
GohubGu46* gohub_query_gu46_by_number_w(GohubConnection* connection,
const GohubWChar* registration_esr, const GohubWChar*
registration_num);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

registration_esr

Станция формирования ведомости. (в кодировке UTF-16).

registration_num

Номер ведомости.. (в кодировке UTF-16).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_query_and_save_gu46_printable_form

Запросить печатную форму ведомости о пользовании вагонами/контейнерами ГУ-46 по его ID и сохранить ее в файл.

Объявление:

```
GohubBool gohub_query_and_save_gu46_printable_form(
      GohubConnection* connection,
      const char* gu46Id,
      const char* path);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect;

`gu46Id`

уникальный идентификатор ведомости о пользовании вагонами/контейнерами.

`path`

путь, по которому сохранить запрошенную печатную форму.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_query_and_save_gu46_printable_form_w

Запросить печатную форму ведомости о пользовании вагонами/контейнерами ГУ-46 по его ID (в кодировке UTF-16) и сохранить ее в файл.

Объявление:

```
GohubBool gohub_query_and_save_gu46_printable_form_w(
      GohubConnection* connection,
      const GohubWChar* gu46Id,
      const GohubWChar* path);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect;

`gu46Id`

уникальный идентификатор ведомости о пользовании вагонами/контейнерами (в кодировке UTF-16);

`path`

путь, по которому сохранить запрошенную печатную форму.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

4.4.8. Работа с памятками о подаче/уборке вагонов и выдаче/приёме контейнеров ГУ-45

gohub_query_gu45	Запросить памятку о подаче/уборке вагонов и выдаче/приёме контейнеров
gohub_query_gu45_w	Запросить памятку о подаче/уборке вагонов и выдаче/приёме контейнеров (в UTF-16)
gohub_query_next_gu45	Запросить следующую памятку о подаче/уборке вагонов и выдаче/приёме контейнеров
gohub_gu45_id	Запросить ID памятки о подаче/уборке вагонов и выдаче/приёме контейнеров
gohub_gu45_id_w	Запросить ID памятки о подаче/уборке вагонов и выдаче/приёме контейнеров (в UTF-16)
gohub_gu45_status	Запросить статус памятку о подаче/уборке вагонов и выдаче/приёме контейнеров
gu45_revision	Запросить номер ревизии памятки о подаче/уборке вагонов и выдаче/приёме контейнеров
gohub_gu45_text	Запросить текст памятки о подаче/уборке вагонов и

	выдаче/приёме контейнеров
gohub_gu45_text_w	Запросить текст памятки о подаче/уборке вагонов и выдаче/приёме контейнеров (в UTF-16)
gohub_gu45_signer_info	Запросить информацию о подписанте памятки о подаче/уборке вагонов и выдаче/приёме контейнеров
gohub_gu45_signer_info_w	Запросить информацию о подписанте памятки о подаче/уборке вагонов и выдаче/приёме контейнеров (в UTF-16)
gohub_gu45_sign_time	Запросить время подписания памятки о подаче/уборке вагонов и выдаче/приёме контейнеров
gohub_gu45_sign_time_w	Запросить время подписания памятки о подаче/уборке вагонов и выдаче/приёме контейнеров (в UTF-16)
gohub_gu45_signer_name_w	Запросить имя подписанта (в UTF-16)
gohub_gu45_has_signature	Проверить подписан ли документ
gu45_size	Запросить предполагаемый размер памятки о подаче/уборке вагонов и выдаче/приёме контейнеров представленный в текущей кодовой странице
gohub_save_gu45	Сохранить памятку о подаче/уборке вагонов и выдаче/приёме контейнеров
gohub_save_gu45_w	Сохранить памятку о подаче/уборке вагонов и выдаче/приёме контейнеров (в UTF-16)
gohub_close_gu45	Закрыть памятку о подаче/уборке вагонов и выдаче/приёме контейнеров
gohub_query_gu45_by_number	Запросить памятку о подаче/уборке вагонов и выдаче/приёме контейнеров за номером станции, номером памятки и датой оформления
gohub_query_gu45_by_number_w	Запросить памятку о подаче/уборке вагонов и выдаче/приёме контейнеров за номером станции, номером памятки и датой оформления (в UTF-16)
gohub_query_and_save_gu45_printable_form	Запросить печатную памятки о подаче/уборке вагонов и выдаче/приёме контейнеров
gohub_query_and_save_gu45_printable_form_w	Запросить печатную форму памятки о подаче/уборке вагонов и выдаче/приёме контейнеров (в UTF-16)

gohub_query_gu45

Запросить памятку о подаче/уборке вагонов и выдаче/приёме контейнеров ГУ-45.

Объявление:

```
GohubGu45* gohub_query_gu45(
GohubConnection* connection,
const char* gu45Id);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`gu45Id`

уникальный идентификатор памятки о подаче/уборке вагонов и выдаче/приёме контейнеров.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_query_gu45_w

Запросить памятку о подаче/уборке вагонов и выдаче/приёме контейнеров ГУ-45.

Объявление:

```
GohubGu45* gohub_query_gu45_w(
GohubConnection* connection,
const GohubWChar* gu45Id);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`gu45Id`

уникальный идентификатор памятки о подаче/уборке вагонов и выдаче/приёме контейнеров (в кодировке UTF-16).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_query_next_gu45

Запросить следующую памятку о подаче/уборке вагонов и выдаче/приёме контейнеров ГУ-45.

Объявление:

```
GohubGu45* gohub_query_next_gu45(
GohubConnection* connection,
int lastRevision);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`lastRevision`

ревизия документа, от которой начинается поиск следующей памятки о подаче/уборке вагонов и выдаче/приёме контейнеров.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_gu45_id

Запрос ID памятки о подаче/уборке вагонов и выдаче/приёме контейнеров ГУ-45.

Объявление:

```
const char* gohub_gu45_id(
GohubGu45* gu45);
```

Параметры:

`gu45`

Объект документа (`GohubGu45`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др`.

gohub_gu45_id_w

Запрос ID памятки о подаче/уборке вагонов и выдаче/приёме контейнеров ГУ-45.

Объявление:

```
const GohubWChar* gohub_gu45_id_w(
GohubGu45* gu45);
```

Параметры:

`gu45`

Объект документа (`GohubGu45`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др`.

gohub_gu45_status

Запрос статуса памятки о подаче/уборке вагонов и выдаче/приёме контейнеров ГУ-45.

Объявление:

```
GohubGu45Status gohub_gu45_status(
GohubGu45* gu45);
```

Параметры:

`gu45`

Объект документа (`GohubGu45`), статус которого запрашивается;

Результат:

В случае успеха – значение из перечисления `GohubGu45Status`. В случае ошибки – «-1». Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др`.

gohub_gu45_revision

Запрос номера ревизии памятки о подаче/уборке вагонов и выдаче/приёме контейнеров ГУ-45.

Объявление:

```
int gohub_gu45_revision(
GohubGu45* gu45);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`gu45`

объект документа (`GohubGu45).`

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_gu45_text

Запрос текста памятки о подаче/уборке вагонов и выдаче/приёме контейнеров ГУ-45.

Объявление:

```
const char* gohub_gu45_text(
GohubGu45* gu45);
```

Параметры:

`gu45`

Объект документа (`GohubGu45).`

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_gu45_text_w

Запрос текста памятки о подаче/уборке вагонов и выдаче/приёме контейнеров ГУ-45.

Объявление:

```
const GohubWChar* gohub_gu45_text_w(
GohubGu45* gu45);
```

Параметры:

`gu45`

Объект документа (`GohubGu45).`

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_gu45_size

Запрос предполагаемого размера файла, который получится при сохранении документа в текущей кодировке (`GohubGu45).`

Объявление:

```
int gohub_gu45_size(
GohubGu45* gu45);
```

Параметры:

`gu45`

Объект документа (`GohubGu45), текст которого запрашивается;`

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_gu45_signer_info

Запрос информации о памятки о подаче/уборке вагонов и выдаче/приёме контейнеров ГУ-45.

Объявление:

```
const char* gohub_gu45_signer_info(
GohubGu46* gu45);
```

Параметры:

gu45

Объект документа (GohubGu46).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_gu45_signer_info_w

Запрос информации о памятки о подаче/уборке вагонов и выдаче/приёме контейнеров ГУ-45.

Объявление:

```
const GohubWChar* gohub_gu45_signer_info_w(
GohubGu46* gu45);
```

Параметры:

gu45

Объект документа (GohubGu46).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_gu45_sign_time

Запрос времени подписания памятки о подаче/уборке вагонов и выдаче/приёме контейнеров ГУ-45.

Объявление:

```
const char* gohub_gu45_sign_time(
GohubGu45* gu45);
```

Параметры:

gu45

Объект документа (GohubGu46).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_gu45_sign_time_w

Запрос времени подписания памятки о подаче/уборке вагонов и выдаче/приёме контейнеров ГУ-45.

Объявление:

```
const GohubWChar* gohub_gu45_sign_time_w(
GohubGu45* gu45);
```

Параметры:

`gu45`

Объект документа (`GohubGu45`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др.`

gohub_gu45_sign_name_w

Запрос имя подписанта памятки о подаче/уборке вагонов и выдаче/приёме контейнеров ГУ-45 (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_gu45_sign_name_w(
GohubGu45* gu45);
```

Параметры:

`gu45`

Объект документа (`GohubGu45`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др.`

gohub_gu45_has_signature

Проверить подписан ли документ.

Объявление:

```
const GohubWChar* gohub_gu45_has_signature(
GohubGu45* gu45);
```

Параметры:

`gu45`

Объект документа (`GohubGu45`).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др.`

gohub_save_gu45

Сохранение памятки о подаче/уборке вагонов и выдаче/приёме контейнеров ГУ-45.

Объявление:

```
GohubBool gohub_save_gu45(
GohubGu45* gu45,
const char* path,
int codePage);
```

Параметры:

`gu45`

Объект документа (`GohubGu45`), который сохраняется на диск;

`path`

путь, куда сохраняется файл;

`codePage`

Числовое обозначение страницы кодировки, в которой сохраняется файл.

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др.`

`gohub_save_gu45_w`

Сохранение памятки о подаче/уборке вагонов и выдаче/приёме контейнеров ГУ-45.

Объявление:

```
GohubBool gohub_save_gu45_w(
GohubGu45* gu45,
const GohubWChar* path,
int codePage);
```

Параметры:

`gu45`

Объект документа (`GohubGu45`), который сохраняется на диск;

`path`

путь, куда сохраняется файл (в кодировке UTF-16);

`codePage`

Числовое обозначение страницы кодировки, в которой сохраняется файл.

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др.`

`gohub_close_gu45`

Закрытие документа (`GohubGu45`).

Объявление:

```
GohubBool gohub_close_gu45(
GohubGu45* gu45);
```

Параметры:

`gu45`

Объект документа (`GohubGu45`).

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др.`

`gohub_query_gu45_by_number`

Запросить памятку о подаче/уборке вагонов и выдаче/приёме контейнеров за номером станции, номером памятки и датой оформления .

Объявление:

```
GohubGu45* gohub_query_gu45_by_number(GohubConnection* connection,
const char* registration_esr, const char* registration_num , const
char* registration_date);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

registration_esr

Станция формирования памятки.

registration_num

Номер памятки..

registration_date

Дата формирования памятки.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_query_gu45_by_number_w

Запросить памятку о подаче/уборке вагонов и выдаче/приёме контейнеров за номером станции, номером памятки и датой оформления (в UTF-16)

Объявление:

```
GohubGu45* gohub_query_gu45_by_number_w(GohubConnection* connection,
const GohubWChar* registration_esr, const GohubWChar*
registration_num , const GohubWChar* registration_date);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

registration_esr

Станция формирования памятки (в кодировке UTF-16).

registration_num

Номер памятки (в кодировке UTF-16).

registration_date

Дата формирования памятки (в кодировке UTF-16).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_query_and_save_gu45_printable_form

Запросить печатную форму о подаче/уборке вагонов и выдаче/приёме контейнеров ГУ-45 по его ID и сохранить ее в файл.

Объявление:

```
GohubBool gohub_query_and_save_gu45_printable_form(
      GohubConnection* connection,
      const char* gu45Id,
      const char* path);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect;

`gu45Id`

уникальный идентификатор памятки о подаче/уборке вагонов и выдаче/приёме контейнеров.

`path`

путь, по которому сохранить запрошенную печатную форму.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_query_and_save_gu45_printable_form_w

Запросить печатную форму о подаче/уборке вагонов и выдаче/приёме контейнеров ГУ-45 по его ID (в кодировке UTF-16) и сохранить ее в файл.

Объявление:

```
GohubBool gohub_query_and_save_gu45_printable_form_w(
      GohubConnection* connection,
      const GohubWChar* gu45Id,
      const GohubWChar* path);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect;

`gu45Id`

уникальный идентификатор памятки о подаче/уборке вагонов и выдаче/приёме контейнеров (в кодировке UTF-16);

`path`

путь, по которому сохранить запрошенную печатную форму.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

4.4.9. Работа с фильтрами для запрашиваемых документов

Ниже перечисленные функции устанавливают фильтры, влияющие на функции запроса документов из п. 4.4.3.

gohub_clear_all_filters	Сбросить все фильтры
gohub_set_filter_by_document_status	Установить фильтр по статусу документа
gohub_set_filter_by_document_number	Установить фильтр по номеру документа
gohub_set_filter_by_document_number_w	Установить фильтр по номеру документа (в UTF-16)
gohub_set_filter_by_wagon_number	Установить фильтр по номеру вагона
gohub_set_filter_by_wagon_number_w	Установить фильтр по номеру вагона (в UTF-16)
gohub_set_filter_by_departure_client	Установить фильтр по коду отправителя
gohub_set_filter_by_departure_client_w	Установить фильтр по коду отправителя (в UTF-16)
gohub_set_filter_by_departure_payer	Установить фильтр по коду плательщика по отправлению
gohub_set_filter_by_departure_payer_w	Установить фильтр по коду плательщика по отправлению (в UTF-16)
gohub_set_filter_by_departure_station	Установить фильтр по коду станции отправления
gohub_set_filter_by_departure_station_w	Установить фильтр по коду станции отправления (в UTF-16)
gohub_set_filter_by_arrival_client	Установить фильтр по коду получателя
gohub_set_filter_by_arrival_client_w	Установить фильтр по коду получателя (в UTF-16)
gohub_set_filter_by_arrival_payer	Установить фильтр по коду плательщика по прибытию
gohub_set_filter_by_arrival_payer_w	Установить фильтр по коду плательщика по прибытию (в UTF-16)
gohub_set_filter_by_arrival_station	Установить фильтр по коду станции назначения
gohub_set_filter_by_arrival_station_w	Установить фильтр по коду станции назначения (в UTF-16)
gohub_get_filter_by_document_status	Запросить значение фильтра по статусу документа
gohub_get_filter_by_document_number	Запросить значение фильтра по номеру документа
gohub_get_filter_by_document_number_w	Запросить значение фильтра по номеру документа (в UTF-16)
gohub_get_filter_by_wagon_number	Запросить значение фильтра по номеру вагона
gohub_get_filter_by_wagon_number_w	Запросить значение фильтра по номеру вагона (в UTF-16)
gohub_get_filter_by_departure_client	Запросить значение фильтра по коду отправителя
gohub_get_filter_by_departure_client_w	Запросить значение фильтра по коду отправителя (в UTF-16)
gohub_get_filter_by_departure_payer	Запросить значение фильтра по коду плательщика по отправлению
gohub_get_filter_by_departure_payer_w	Запросить значение фильтра по коду плательщика по отправлению (в UTF-16)
gohub_get_filter_by_departure_station	Запросить значение фильтра по коду станции отправления
gohub_get_filter_by_departure_station_w	Запросить значение фильтра по коду станции отправления (в UTF-16)
gohub_get_filter_by_arrival_client	Запросить значение фильтра по коду получателя
gohub_get_filter_by_arrival_client_w	Запросить значение фильтра по коду получателя (в UTF-16)
gohub_get_filter_by_arrival_payer	Запросить значение фильтра по коду плательщика по прибытию
gohub_get_filter_by_arrival_payer_w	Запросить значение фильтра по коду плательщика по

	прибытию (в UTF-16)
gohub_get_filter_by_arrival_station	Запросить значение фильтра по коду станции назначения
gohub_get_filter_by_arrival_station_w	Запросить значение фильтра по коду станции назначения (в UTF-16)

gohub_clear_all_filters

Сбросить все настройки фильтров.

Объявление:

```
GohubBool gohub_clear_all_filters(
      GohubConnection* connection);
```

Параметры:

connection

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_set_filter_by_document_status

Установить фильтр по статусу документа.

Объявление:

```
GohubBool gohub_set_filter_by_document_status(
      GohubConnection* connection,
      int documentStatusCode);
```

Параметры:

connection

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect;

documentStatusCode

код статуса документа, если указать 0 – фильтр будет сброшен на «все»;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_set_filter_by_document_number

Установить фильтр по номеру документа.

Объявление:

```
GohubBool gohub_set_filter_by_document_number(
      GohubConnection* connection,
      const char* documentNumber);
```

Параметры:

connection

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect;

`documentNumber`

номер документа, если передать пустой указатель или пустую строку – фильтр будет сброшен на «все».

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_set_filter_by_document_number_w

Установить фильтр по номеру документа (в кодировке UTF-16).

Объявление:

```
GohubBool gohub_set_filter_by_document_number_w(
      GohubConnection* connection,
      const GohubWChar* documentNumber);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect;

`documentNumber`

номер документа (в кодировке UTF-16), если передать пустой указатель или пустую строку – фильтр будет сброшен на «все».

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_set_filter_by_wagon_number

Установить фильтр по номеру вагона.

Объявление:

```
GohubBool gohub_set_filter_by_wagon_number(
      GohubConnection* connection,
      const char* wagonNumber);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect;

`wagonNumber`

номер вагона, если передать пустой указатель или пустую строку – фильтр будет сброшен на «все».

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_set_filter_by_wagon_number_w

Установить фильтр по номеру вагона (в кодировке UTF-16).

Объявление:

```
GohubBool gohub_set_filter_by_wagon_number_w(
      GohubConnection* connection,
      const GohubWChar* wagonNumber);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect;

`wagonNumber`

номер вагона (в кодировке UTF-16), если передать пустой указатель или пустую строку – фильтр будет сброшен на «все».

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

`gohub_set_filter_by_departure_client`

Установить фильтр по коду отправителя.

Объявление:

```
GohubBool gohub_set_filter_by_departure_client(
      GohubConnection* connection,
      const char* clientCode);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect;

`clientCode`

код отправителя, если передать пустой указатель или пустую строку – фильтр будет сброшен на «все».

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

`gohub_set_filter_by_departure_client_w`

Установить фильтр по коду отправителя (в кодировке UTF-16).

Объявление:

```
GohubBool gohub_set_filter_by_departure_client_w(
      GohubConnection* connection,
      const GohubWChar* clientCode);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect;

`clientCode`

код отправителя (в кодировке UTF-16), если передать пустой указатель или пустую строку – фильтр будет сброшен на «все».

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_set_filter_by_departure_payer

Установить фильтр по коду плательщика по отправлению.

Объявление:

```
GohubBool gohub_set_filter_by_departure_payer(
      GohubConnection* connection,
      const char* payerCode);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect;

`payerCode`

код плательщика по отправлению, если передать пустой указатель или пустую строку – фильтр будет сброшен на «все».

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_set_filter_by_departure_payer_w

Установить фильтр по коду плательщика по отправлению (в кодировке UTF-16).

Объявление:

```
GohubBool gohub_set_filter_by_departure_payer_w(
      GohubConnection* connection,
      const GohubWChar* payerCode);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect;

`payerCode`

код плательщика по отправлению (в кодировке UTF-16), если передать пустой указатель или пустую строку – фильтр будет сброшен на «все».

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_set_filter_by_departure_station

Установить фильтр по коду станции отправления.

Объявление:

```
GohubBool gohub_set_filter_by_departure_station(
      GohubConnection* connection,
      const char* stationCode);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect;

`stationCode`

код станции отправления, если передать пустой указатель или пустую строку – фильтр будет сброшен на «все».

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_set_filter_by_departure_station_W

Установить фильтр по коду станции отправления (в кодировке UTF-16).

Объявление:

```
GohubBool gohub_set_filter_by_departure_station_w(
      GohubConnection* connection,
      const GohubWChar* stationCode);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect;

`stationCode`

код станции отправления (в кодировке UTF-16), если передать пустой указатель или пустую строку – фильтр будет сброшен на «все».

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_set_filter_by_arrival_client

Установить фильтр по коду получателя.

Объявление:

```
GohubBool gohub_set_filter_by_arrival_client(
      GohubConnection* connection,
      const char* clientCode);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect;

`clientCode`

код получателя, если передать пустой указатель или пустую строку – фильтр будет сброшен на «все».

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_set_filter_by_arrival_client_W

Установить фильтр по коду получателя (в кодировке UTF-16).

Объявление:

```
GohubBool gohub_set_filter_by_arrival_client_w(
      GohubConnection* connection,
      const GohubWChar* clientCode);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect;

`clientCode`

код получателя (в кодировке UTF-16), если передать пустой указатель или пустую строку – фильтр будет сброшен на «все».

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_set_filter_by_arrival_payer

Установить фильтр по коду плательщика по прибытию.

Объявление:

```
GohubBool gohub_set_filter_by_arrival_payer(
      GohubConnection* connection,
      const char* payerCode);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect;

`payerCode`

код плательщика по прибытию, если передать пустой указатель или пустую строку – фильтр будет сброшен на «все».

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_set_filter_by_arrival_payer_w

Установить фильтр по коду плательщика по прибытию (в кодировке UTF-16).

Объявление:

```
GohubBool gohub_set_filter_by_arrival_payer_w(
      GohubConnection* connection,
      const GohubWChar* payerCode);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect;

`payerCode`

код плательщика по прибытию (в кодировке UTF-16), если передать пустой указатель или пустую строку – фильтр будет сброшен на «все».

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_set_filter_by_arrival_station

Установить фильтр по коду станции назначения.

Объявление:

```
GohubBool gohub_set_filter_by_arrival_station(
      GohubConnection* connection,
      const char* stationCode);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect;

`stationCode`

код станции назначения, если передать пустой указатель или пустую строку – фильтр будет сброшен на «все».

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_set_filter_by_arrival_station_w

Установить фильтр по коду станции назначения (в кодировке UTF-16).

Объявление:

```
GohubBool gohub_set_filter_by_arrival_station_w(
      GohubConnection* connection,
      const GohubWChar* stationCode);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect;

`stationCode`

код станции назначения (в кодировке UTF-16), если передать пустой указатель или пустую строку – фильтр будет сброшен на «все».

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_get_filter_by_document_status

Запросить значение фильтра по статусу документа. Возвращает значение целого типа, описанного перечислением `GohubDocumentStatus`.

Объявление:

```
GohubDocumentStatus gohub_get_filter_by_document_status(
      GohubConnection* connection);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect.

Результат:

В случае успеха – не пустая строка. Если результат пустая строка - значит фильтр не наложен или соединение не действительно. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_get_filter_by_document_number

Запросить значение фильтра по номеру документа.

Объявление:

```
const char* gohub_get_filter_by_document_number(
      GohubConnection* connection);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect.

Результат:

В случае успеха – не пустая строка. Если результат пустая строка - значит фильтр не наложен или соединение не действительно. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_get_filter_by_document_number_w

Запросить значение фильтра по номеру документа (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_get_filter_by_document_number_w(
      GohubConnection* connection);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect.

Результат:

В случае успеха – не пустая строка. Если результат пустая строка - значит фильтр не наложен или соединение не действительно. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_get_filter_by_wagon_number

Запросить значение фильтра по номеру вагона.

Объявление:

```
const char* gohub_get_filter_by_wagon_number(
      GohubConnection* connection);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect.

Результат:

В случае успеха – не пустая строка. Если результат пустая строка - значит фильтр не наложен или соединение не действительно. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_get_filter_by_wagon_number_w

Запросить значение фильтра по номеру вагона (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_get_filter_by_wagon_number_w(
      GohubConnection* connection);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect.

Результат:

В случае успеха – не пустая строка. Если результат пустая строка - значит фильтр не наложен или соединение не действительно. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_get_filter_by_departure_client

Запросить значение фильтра по коду отправителя.

Объявление:

```
const char* gohub_get_filter_by_departure_client(
      GohubConnection* connection);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect.

Результат:

В случае успеха – не пустая строка. Если результат пустая строка - значит фильтр не наложен или соединение не действительно. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_get_filter_by_departure_client_w

Запросить значение фильтра по коду отправителя (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_get_filter_by_departure_client_w(
      GohubConnection* connection);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect.

Результат:

В случае успеха – не пустая строка. Если результат пустая строка - значит фильтр не наложен или соединение не действительно. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_get_filter_by_departure_payer

Запросить значение фильтра по коду плательщика по отправлению.

Объявление:

```
const char* gohub_get_filter_by_departure_payer(
      GohubConnection* connection);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect.

Результат:

В случае успеха – не пустая строка. Если результат пустая строка - значит фильтр не наложен или соединение не действительно. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

`gohub_get_filter_by_departure_payer_w`

Запросить значение фильтра по коду плательщика по отправлению (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_get_filter_by_departure_payer_w(
      GohubConnection* connection);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect.

Результат:

В случае успеха – не пустая строка. Если результат пустая строка - значит фильтр не наложен или соединение не действительно. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

`gohub_get_filter_by_departure_station`

Запросить значение фильтра по коду станции отправления.

Объявление:

```
const char* gohub_get_filter_by_departure_station(
      GohubConnection* connection);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect.

Результат:

В случае успеха – не пустая строка. Если результат пустая строка - значит фильтр не наложен или соединение не действительно. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

`gohub_get_filter_by_departure_station_w`

Запросить значение фильтра по коду станции отправления (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_get_filter_by_departure_station_w(
```

```
            GohubConnection* connection);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect.

Результат:

В случае успеха – не пустая строка. Если результат пустая строка - значит фильтр не наложен или соединение не действительно. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_get_filter_by_arrival_client

Запросить значение фильтра по коду получателя.

Объявление:

```
const char* gohub_get_filter_by_arrival_client(
      GohubConnection* connection);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect.

Результат:

В случае успеха – не пустая строка. Если результат пустая строка - значит фильтр не наложен или соединение не действительно. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_get_filter_by_arrival_client_w

Запросить значение фильтра по коду получателя (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_get_filter_by_arrival_client_w(
      GohubConnection* connection);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect.

Результат:

В случае успеха – не пустая строка. Если результат пустая строка - значит фильтр не наложен или соединение не действительно. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_get_filter_by_arrival_payer

Запросить значение фильтра по коду плательщика по прибытию.

Объявление:

```
const char* gohub_get_filter_by_arrival_payer(
      GohubConnection* connection);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect.

Результат:

В случае успеха – не пустая строка. Если результат пустая строка - значит фильтр не наложен или соединение не действительно. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_get_filter_by_arrival_payer_w

Запросить значение фильтра по коду плательщика по прибытию (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_get_filter_by_arrival_payer_w(
      GohubConnection* connection);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect.

Результат:

В случае успеха – не пустая строка. Если результат пустая строка - значит фильтр не наложен или соединение не действительно. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_get_filter_by_arrival_station

Запросить значение фильтра по коду станции прибытия.

Объявление:

```
const char* gohub_get_filter_by_arrival_station(
      GohubConnection* connection);
const char* gohub_get_filter_by_arrival_payer(
      GohubConnection* connection);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect.

Результат:

В случае успеха – не пустая строка. Если результат пустая строка - значит фильтр не наложен или соединение не действительно. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_get_filter_by_arrival_station_w

Запросить значение фильтра по коду станции прибытия (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_get_filter_by_arrival_station_w(
      GohubConnection* connection);
```

Параметры:

`connection`

указатель соединения с Модулем Согласования, полученный ранее при помощи функции gohub_connect.

Результат:

В случае успеха – не пустая строка. Если результат пустая строка - значит фильтр не наложен или соединение не действительно. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

4.4.10. Проверка электронно-цифровой подписи

gohub_document_has_signature	Узнать, содержит ли документ электронно-цифровую подпись
gohub_document_check_signature	Выполнить проверку электронно-цифровой подписи документа
gohub_document_signer_name	Получить имя лица, подписавшего электронный документ
gohub_document_signer_name_w	Получить имя лица, подписавшего электронный документ (в UTF-16)
gohub_document_signer_info	Получить подробную информацию о лице, подписавшем электронный документ
gohub_document_signer_info_w	Получить подробную информацию о лице, подписавшем электронный документ (в UTF-16)
gohub_document_sign_time	Получить метку времени электронно-цифровой подписи
gohub_document_sign_time_w	Получить метку времени электронно-цифровой подписи (в UTF-16)
gohub_document_signer_info_by_index	Получить подробную информацию о лице, подписавшем электронный документ по индексу
gohub_document_signer_info_by_index_w	Получить подробную информацию о лице, подписавшем электронный документ по индексу (в UTF-16)
gohub_document_signer_name_by_index	Получить имя лица, подписавшего электронный документ по индексу
gohub_document_signer_name_by_index_w	Получить имя лица, подписавшего электронный документ по индексу (в UTF-16)
gohub_document_sign_time_by_index	Получить метку времени электронно-цифровой подписи по индексу
gohub_document_sign_time_by_index_w	Получить метку времени электронно-цифровой подписи по индексу (в UTF-16)
gohub_document_signature_count	Получить количество подписей в документе
gohub_document_sign_status_by_index	Получить статус документа на момент его подписания по индексу
gohub_document_sign_error_by_index	Получить текст ошибки при разборе электронно-цифровой подписи по индексу
gohub_document_sign_error_by_index_w	Получить текст ошибки при разборе электронно-цифровой подписи по индексу (в UTF-16)

gohub_document_has_signature

Проверка, есть ли у документа электронная подпись (GohubDocument).

Объявление:

```
GohubBool gohub_document_has_signature(
    GohubDocument* document);
```

Параметры:

document

Объект документа (GohubDocument);

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_document_check_signature

Проверка соответствия документа его электронной подписи.

Объявление:

```
GohubBool gohub_document_check_signature(
    GohubDocument* document);
```

Параметры:

document

Документ;

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_document_signature_name

Запрос имени человека, подписавшего документ.

Объявление:

```
const char* gohub_document_signer_name(
    GohubDocument* document);
```

Параметры:

document

Документ;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_document_signature_name_w

Запрос имени человека, подписавшего документ, результат будет в кодировке UTF-16.

Объявление:

```
const GohubWChar* gohub_document_signer_name_w(
    GohubDocument* document);
```

Параметры:

document

Документ;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_document_signer_info

Запрос информации о подписавшем документ – возвращается форматированная строка с ключами полей и их значениями, через разделитель.

Объявление:

```
const char* gohub_document_signer_info(
        GohubDocument* document);
```

Параметры:

`document`

Документ;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_document_signer_info_w

Запрос информации о подписавшем документ – возвращается форматированная строка (в кодировке UTF-16) с ключами полей и их значениями, через разделитель.

Объявление:

```
const GohubWChar* gohub_document_signer_info_w(
        GohubDocument* document);
```

Параметры:

`document`

Документ;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_document_sign_time

Запрос информации о времени подписания документа.

Объявление:

```
const char* gohub_document_sign_time(
        GohubDocument* document);
```

Параметры:

`document`

Документ;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_document_sign_time_w

Запрос информации о времени подписания документа, возвращает значение в кодировке UTF-16.

Объявление:

```
const GohubWChar* gohub_document_sign_time_w(
    GohubDocument* document);
```

Параметры:

`document`

Документ;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_document_signature_name_by_index

Запрос имени человека, подписавшего документ по индексу подписи.

Объявление:

```
const char* gohub_document_signer_name_by_index(
    GohubDocument* document, int index);
```

Параметры:

`document`

Документ;

`index`

Индекс;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_document_signature_name_by_index_w

Запрос имени человека, подписавшего документ по индексу подписи, результат будет в кодировке UTF-16.

Объявление:

```
const GohubWChar* gohub_document_signer_name_by_index_w(
    GohubDocument* document, int index);
```

Параметры:

`document`

Документ;

`index`

Индекс;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_document_signer_info_by_index

Запрос информации о подписавшем документ по индексу подписи – возвращается форматированная строка с ключами полей и их значениями, через разделитель.

Объявление:

```
const char* gohub_document_signer_info_by_index(
    GohubDocument* document, int index);
```

Параметры:

document

Документ;

index

Индекс;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_document_signer_info_by_index_w

Запрос информации о подписавшем документ по индексу подписи – возвращается форматированная строка (в кодировке UTF-16) с ключами полей и их значениями, через разделитель.

Объявление:

```
const GohubWChar* gohub_document_signer_info_w(
    GohubDocument* document, int index);
```

Параметры:

document

Документ;

index

Индекс;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_document_sign_time_by_index

Запрос информации о времени подписания документа по индексу подписи.

Объявление:

```
const char* gohub_document_sign_time(
    GohubDocument* document, int index);
```

Параметры:

document

Документ;

index

Индекс;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_document_sign_time_by_index_w

Запрос информации о времени подписания документа по индексу подписи, возвращает значение в кодировке UTF-16.

Объявление:

```
const GohubWChar* gohub_document_sign_time_w(
      GohubDocument* document, int index);
```

Параметры:

document

Документ;

index

Индекс;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

4.4.11. Наложение электронно-цифровой подписи

gohub_open_private_key	Отрыть электронный ключ по паролю
gohub_open_private_key_w	Отрыть электронный ключ по паролю (в UTF-16)
gohub_open_private_key_by_bytes	Отрыть электронный ключ в виде массива байт по паролю
gohub_open_private_key_by_bytes_w	Отрыть электронный ключ в виде массива байт по паролю (в UTF-16)
gohub_open_private_key_from_path	Отрыть электронный ключ по пути и паролю
gohub_open_private_key_from_path_w	Отрыть электронный ключ по пути и паролю (в UTF-16)
gohub_private_key_owner_name	Получить имя владельца электронного ключа
gohub_private_key_owner_name_w	Получить имя владельца электронного ключа (в UTF-16)
gohub_private_key_owner_info	Получить подробную информацию о владельце электронного ключа
gohub_private_key_owner_info_w	Получить подробную информацию о владельце электронного ключа (в UTF-16)
gohub_sign_document	Подписать документ электронно-цифровой подписью
gohub_close_private_key	Завершить работу с электронным ключом
gohub_sign_fdu92	Подписать ФДУ-92 электронно-цифровой подписью
gohub_sign_gu46	Подписать ГУ-46 электронно-цифровой подписью
gohub_delete_old_certs_csk_uz	Запросить сертификаты АЦСК УЗ

gohub_delete_old_certs_csk_uz_w	Запросить сертификаты АЦСК УЗ (в UTF-16)

gohub_open_private_key

Открытие сессии работы электронно-цифрового ключа.

Объявление:

```
GohubBool gohub_open_private_key(
      GohubConnection* connection,
      const char* passwordToKey);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`passwordToKey`

Строка с паролем от электронного ключа;

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др.`

gohub_open_private_key_w

Открытие сессии работы электронно-цифрового ключа.

Объявление:

```
GohubBool gohub_open_private_key_w(
      GohubConnection* connection,
      const GohubWChar* passwordToKey);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`passwordToKey`

Строка с паролем от электронного ключа в кодировке UTF-16;

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др.`

gohub_open_private_key_by_bytes

Открытие сессии работы электронно-цифрового ключа, который представлен массивом байт.

Объявление:

```
GohubBool gohub_open_private_key(
      GohubConnection* connection,
      const char* passwordToKey,
      unsigned char* keyBinary,
      unsigned int length);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`passwordToKey`

Строка с паролем от электронного ключа;

`keyBinary`

Электронный ключ в виде массива байт;

`length`

Длина массива байт;

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др.`

gohub_open_private_key_by_bytes_w

Открытие сессии работы электронно-цифрового ключа, который представлен массивом байт.

Объявление:

```
GohubBool gohub_open_private_key_w(
      GohubConnection* connection,
      const GohubWChar* passwordToKey,
      unsigned char* keyBinary,
      unsigned int length);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`passwordToKey`

Строка с паролем от электронного ключа в кодировке UTF-16;

`keyBinary`

Электронный ключ в виде массива байт;

`length`

Длина массива байт;

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* `и др.`

gohub_open_private_key_from_path

Открытие сессии работы электронно-цифрового ключа, который расположен по указанному пути.

Объявление:

```
GohubBool gohub_open_private_key(
      GohubConnection* connection,
      const char* passwordToKey,
      const char* keyFileName);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`passwordToKey`

Строка с паролем от электронного ключа;

`keyFileName`

Путь к электронному ключу;

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_open_private_key_from_path_w

Открытие сессии работы электронно-цифрового ключа, который расположен по указанному пути.

Объявление:

```
GohubBool gohub_open_private_key_w(
      GohubConnection* connection,
      const GohubWChar* passwordToKey,
      const GohubWChar* keyFileName);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`passwordToKey`

Строка с паролем от электронного ключа в кодировке UTF-16;

`keyFileName`

Путь к электронному ключу в кодировке UTF-16;

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_private_key_owner_name

Возвращает имя владельца электронно-цифрового ключа (в кодировке UTF-16), рабочая сессия которого на данный момент открыта.

Объявление:

```
const char* gohub_private_key_owner_name(
      GohubConnection* connection);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_private_key_owner_name_w

Возвращает имя владельца электронно-цифрового ключа (в кодировке UTF-16), рабочая сессия которого на данный момент открыта.

Объявление:

```
const GohubWChar* gohub_private_key_owner_name_w(
      GohubConnection* connection);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_private_key_owner_info

Запрос информации о владельце электронно-цифрового ключа, рабочая сессия которого на данный момент открыта – возвращается форматированная строка с ключами полей и их значениями, через разделитель.

Объявление:

```
const char* gohub_private_key_owner_info(

      GohubConnection* connection);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_private_key_owner_info_w

Запрос информации о владельце электронно-цифрового ключа, рабочая сессия которого на данный момент открыта – возвращается форматированная строка (в кодировке UTF-16) с ключами полей и их значениями, через разделитель.

Объявление:

```
const GohubWChar* gohub_private_key_owner_info_w(

      GohubConnection* connection);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_sign_document

Подписание документа электронно-цифровым ключом, сессия которого на данный момент открыта.

Объявление:

```
GohubBool gohub_sign_document(
      GohubConnection* connection,
      GohubDocument* document);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`document`

Документ;

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_close_private_key

Закрытие рабочей сессии электронно-цифрового ключа.

Объявление:

```
GohubBool gohub_close_private_key(
      GohubConnection* connection);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_sign_fdu92

Подписание ФДУ-92 электронно-цифровым ключом, сессия которого на данный момент открыта.

Объявление:

```
GohubBool gohub_sign_fdu92(
GohubConnection* connection, GohubFdu92* fdu92);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`fdu92`

Документ;

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_sign_gu46

Подписание ГУ-46 электронно-цифровым ключом, сессия которого на данный момент открыта.

Объявление:

```
GohubBool gohub_sign_gu46(
GohubConnection* connection, GohubGu46* gu46);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`gu46`

Документ;

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_delete_old_certs_csk_uz

Запросить сертификаты АЦСК УЗ.

Объявление:

```
GohubBool gohub_delete_old_certs_csk_uz(
GohubConnection* connection, const char* passwordToKey);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`passwordToKey`

Строка с паролем от электронного ключа;

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_delete_old_certs_csk_uz_w

Запросить сертификаты АЦСК УЗ (в кодировке UTF-16).

Объявление:

```
GohubBool gohub_delete_old_certs_csk_uz_w(
GohubConnection* connection, const GohubWChar* passwordToKey);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`passwordToKey`

Строка с паролем от электронного ключа в кодировке UTF-16;

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_keys_list

Запросить список устройств на которых может быть размещен электронный ключ.

Объявление:

```
const char* gohub_keys_list(GohubConnection* connection);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_keys_list_w

Запросить список устройств на которых может быть размещен электронный ключ (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_keys_list_w(GohubConnection* connection);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_open_private_key_by_index_path

Открытие сессии работы электронно-цифрового ключа, который расположен по указанному индексу из списка gohub_keys_list

Объявление:

```
GohubBool gohub_open_private_key_by_index(GohubConnection*
connection, const char* passwordToKey, int mediaIndex , int
deviceIndex);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`passwordToKey`

Строка с паролем от электронного ключа;

`mediaIndex`

Индекс типа носителя из списка носителей.

deviceIndex

Индекс названия ключа в списке носителей.

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

gohub_open_private_key_by_index_w

Открытие сессии работы электронно-цифрового ключа, который расположен по указанному индексу из списка gohub_keys_list_w

Объявление:

```
GohubBool gohub_open_private_key_by_index_w(GohubConnection*
connection, const GohubWChar* passwordToKey, int mediaIndex , int
deviceIndex);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`passwordToKey`

Строка с паролем от электронного ключа в кодировке UTF-16;

`mediaIndex`

Индекс типа носителя из списка носителей.

deviceIndex

Индекс названия ключа в списке носителей.

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *`gohub_last_error` и др.*

4.4.12. Операции с файлами электронных ключей

Этот блок функций позволяет сделать доступным для использования файлы электронных ключей, размещенных в произвольном месте файловой системы. Монтирование файла ключа по своему эффекту аналогично подключению внешнего носителя (например, модуля флеш-памяти) с электронным ключом.

ВАЖНО! Подключение электронного ключа распространяется только на текущего пользователя.

gohub_mount_file_key	Монтировать электронный ключ из указанного файла
gohub_mount_file_key_w	Монтировать электронный ключ из указанного файла (в UTF-16)
gohub_unmount_file_key	Демонтировать электронный ключ из файла
gohub_unmount_file_key_w	Демонтировать электронный ключ из файла (в UTF-16)
gohub_query_mounted_file_keys	Запросить количество монтированных из файлов электронных ключей
gohub_mounted_file_key_id	Запросить ID электронного ключа по его порядковому номеру в списке
gohub_mounted_file_key_id_w	Запросить ID электронного ключа по его порядковому номеру в списке (в UTF-16)
gohub_mounted_file_key_dir	Запросить путь к электронному ключу по его порядковому номеру в списке
gohub_mounted_file_key_dir_w	Запросить путь к электронному ключу по его порядковому номеру в списке (в UTF-16)

gohub_mount_file_key

Монтировать электронный ключ из указанного файла.

Объявление:

```
GohubBool gohub_mount_file_key(
      const char* keyId,
      const char* path);
```

Параметры:

keyId

Идентификатор ключа;

path

путь к файлу ключа.

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_mount_file_key_w

Монтировать электронный ключ из указанного файла.

Объявление:

```
GohubBool gohub_mount_file_key_w(
      const GohubWChar* keyId,
      const GohubWChar* path);
```

Параметры:

keyId

Идентификатор ключа в кодировке UTF-16;

`path`

путь к файлу ключа в кодировке UTF-16.

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_unmount_file_key

Демонтировать электронный ключ по его ID.

Объявление:

```
GohubBool gohub_unmount_file_key(
      const char* keyId);
```

Параметры:

`keyId`

Идентификатор ключа.

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_unmount_file_key_w

Демонтировать электронный ключ по его ID.

Объявление:

```
GohubBool gohub_unmount_file_key_w(
      const GohubWChar* keyId);
```

Параметры:

`keyId`

Идентификатор ключа в кодировке UTF-16.

Результат:

В случае успеха – значение True. В случае ошибки – False. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_query_mounted_file_keys

Запросить количество монтированных из файлов электронных ключей.

Объявление:

```
int gohub_query_mounted_file_keys();
```

Результат:

В случае успеха – не отрицательное значение. В случае ошибки или пустого списка – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_mounted_file_key_id

Получить ID электронного ключа, по его порядковому номеру в списке монтированных электронных ключей.

Объявление:

```
const char* gohub_mounted_file_key_id(
      int index);
```

Параметры:

`index`

Порядковый номер.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_mounted_file_key_id_w

Получить ID электронного ключа (в кодировке UTF-16), по его порядковому номеру в списке монтированных электронных ключей.

Объявление:

```
const GohubWChar* gohub_mounted_file_key_id_w(
      int index);
```

Параметры:

`index`

Порядковый номер.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_mounted_file_key_dir

Получить путь к файлу электронного ключа по его порядковому номеру.

Объявление:

```
const char* gohub_mounted_file_key_dir(
      int index);
```

Параметры:

`index`

Порядковый номер.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_mounted_file_key_dir_w

Получить путь (в кодировке UTF-16) к файлу электронного ключа по его порядковому номеру.

Объявление:

```
const GohubWChar* gohub_mounted_file_key_dir_w(
      int index);
```

Параметры:

`index`

Порядковый номер.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

4.4.13. Работа с АС «Месплан»

Этот блок функций позволяет получить заявки за указанный месяц из АС «Месплан».

gohub_get_mp_months	Запросить перечень идентификаторов месяцев доступных в АС «Месплан» в виде строки
gohub_get_mp_months_w	Запросить перечень идентификаторов месяцев доступных в АС «Месплан» в виде строки (в UTF-16)
gohub_query_and_save_orders_for_month	Запросить и сохранить заявки за указанный месяц из АС «Месплан» в xml
gohub_query_and_save_orders_for_month_w	Запросить и сохранить заявки за указанный месяц из АС «Месплан» в xml (с параметрами в UTF-16)
gohub_query_and_save_orders_for_month_with_relogin	Запросить и сохранить заявки за указанный месяц из АС «Месплан» в xml. С возможностью указать логин и пароль.
gohub_query_and_save_orders_for_month_with_relogin_w	Запросить и сохранить заявки за указанный месяц из АС «Месплан» в xml. С возможностью указать логин и пароль (с параметрами в UTF-16)

gohub_get_mp_months

Запросить перечень идентификаторов месяцев доступных в АС «Месплан» в виде строки.

Объявление:

```
const char* gohub_get_mp_months(
      GohubConnection* connection,
      int codePage)
```

Параметры:

connection

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции gohub_connect;

codePage

Числовое обозначение страницы кодировки, в которой сохраняется файл.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_get_mp_months_w

Запросить перечень идентификаторов месяцев доступных в АС «Месплан» в виде строки (в UTF-16).

Объявление:

```
const GohubWChar* gohub_get_mp_months_w(
      GohubConnection* connection)
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_query_and_save_orders_for_month

Запросить и сохранить заявки за указанный месяц из АС «Месплан» в xml.

Объявление:

```
GohubBool gohub_query_and_save_orders_for_month(
      GohubConnection* connection,
      const char* month,
      const char* path)
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`month`

Строка с номером месяца за который необходимо запросить заявки.

`Path`

Путь к файлу в который необходимо сохранить заявки.

Результат:

В случае успеха – значение *`True`*. В случае ошибки – *`False`*. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_query_and_save_orders_for_month_w

Запросить и сохранить заявки за указанный месяц из АС «Месплан» в xml.

Объявление:

```
GohubBool gohub_query_and_save_orders_for_month_w(
      GohubConnection* connection,
      const GohubWChar* month,
      const GohubWChar* path)
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`month`

Строка с номером месяца за который необходимо запросить заявки (в кодировке UTF-16).

`Path`

Путь к файлу в который необходимо сохранить заявки (в кодировке UTF-16).

Результат:

В случае успеха – значение *`True`*. В случае ошибки – *`False`*. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_query_and_save_orders_for_month_with_relogin

Запросить и сохранить заявки из АС «Месплан» в xml за указанный месяц. С возможностью указать логин и пароль.

Объявление:

```
GohubBool gohub_query_and_save_orders_for_month(
      GohubConnection* connection,
      const char* month,
      const char* login,
      const char* password,
      const char* path)
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`month`

Строка с номером месяца за который необходимо запросить заявки.

`login`

Логин к АС «Месплан».

`password`

Пароль к АС «Месплан».

`Path`

Путь к файлу в который необходимо сохранить заявки.

Результат:

В случае успеха – значение *`True`*. В случае ошибки – *`False`*. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

gohub_query_and_save_orders_for_month_with_relogin_w

Запросить и сохранить заявки из АС «Месплан» в xml за указанный месяц. С возможностью указать логин и пароль.

Объявление:

```
GohubBool gohub_query_and_save_orders_for_month_w(
      GohubConnection* connection,
      const GohubWChar* month,
      const GohubWChar* login,
      const GohubWChar* password,
```

```
        const GohubWChar* path)
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`month`

Строка с номером месяца за который необходимо запросить заявки (в кодировке UTF-16).

`login`

Логин к АС «Месплан» (в кодировке UTF-16).

`password`

Пароль к АС «Месплан» (в кодировке UTF-16).

`Path`

Путь к файлу в который необходимо сохранить заявки (в кодировке UTF-16).

Результат:

В случае успеха – значение *`True`*. В случае ошибки – *`False`*. Информацию об ошибке можно получить с помощью функции *`gohub_last_error`* и др.

4.4.14. Работа с документами информационных услуг

Этот блок функций позволяет получить документы информационных услуг.

`gohub_query_inform_services_document`	Запросить документ
`gohub_query_next_inform_services_document`	Запросить следующий документ
`gohub_save_inform_services_document`	Сохранить документ
`gohub_save_inform_services_document_w`	Сохранить документ (с параметрами в UTF-16)
`gohub_saveXml_inform_services_document`	Сохранить документ в формате xml
`gohub_saveXml_inform_services_document_w`	Сохранить документ в формате xml (с параметрами в UTF-16)
`gohub_close_inform_services_document`	Закрыть документ
`gohub_inform_services_document_id`	Запросить ID документа
`gohub_inform_services_document_revision`	Запросить ревизию документа
`gohub_inform_services_document_filename`	Запросить имя документа
`gohub_inform_services_document_filename_w`	Запросить имя документа (в UTF-16)
`gohub_inform_services_document_comment`	Запросить комментарий документа
`gohub_inform_services_document_comment_w`	Запросить комментарий документа (в UTF-16)
`gohub_inform_services_document_created_date`	Запросить дату создания документа
`gohub_inform_services_document_created_date_w`	Запросить дату создания документа
`gohub_inform_services_document_doc_date`	Запросить дату документа
`gohub_inform_services_document_doc_date_w`	Запросить дату документа (в UTF-16)
`gohub_inform_services_document_doc_is_empty`	Запросить статус или пустой документ

`gohub_query_inform_services_document`

Запросить документ информационных услуг по идентификатору.

Объявление:

```
GohubInformServicesDoc* gohub_query_inform_services_document(
      GohubConnection* connection,
      unsigned __int64 docId)
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`docId`

Уникальный идентификатор документа, который запрашивается.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_query_next_inform_services_document

Запрос документа информационных услуг следующего по списку ревизий, начиная от ревизии переданной параметром.

Объявление:

```
GohubInformServicesDoc* gohub_query_next_inform_services_document(
      GohubConnection* connection,
      unsigned __int64 lastRevision)
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`lastRevision`

Ревизия документа, от которой начинается поиск следующего документа.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *`gohub_last_error и др.`*

gohub_save_inform_services_document

Сохранение документа информационных услуг (GohubInformServicesDoc) на диск, по указанному пути.

Объявление:

```
GohubBool gohub_save_inform_services_document(
      GohubInformServicesDoc* doc,
      const char* path)
```

Параметры:

`document`

Объект документа (GohubInformServicesDoc), который отправляется на сервер;

`path`

Путь, куда сохраняется файл.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_save_inform_services_document_w

Сохранение документа информационных услуг (GohubInformServicesDoc) на диск, по указанному пути (с параметром в UTF-16).

Объявление:

```
GohubBool gohub_save_inform_services_document_w(
      GohubInformServicesDoc* doc,
      const GohubWChar* path)
```

Параметры:

`document`

Объект документа (GohubInformServicesDoc), который отправляется на сервер;

`path`

Путь, куда сохраняется файл (в кодировке UTF-16).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_saveXml_inform_services_document

Сохранение документа информационных услуг в формате xml (GohubInformServicesDoc) на диск, по указанному пути.

Объявление:

```
GohubBool gohub_saveXml_inform_services_document(
      GohubInformServicesDoc* doc,
      const char* path)
```

Параметры:

`document`

Объект документа (GohubInformServicesDoc), который отправляется на сервер;

`path`

Путь, куда сохраняется файл.

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_saveXml_inform_services_document_w

Сохранение документа информационных услуг в формате xml (GohubInformServicesDoc) на диск, по указанному пути (с параметром в UTF-16).

Объявление:

```
GohubBool gohub_saveXml_inform_services_document_w(
      GohubInformServicesDoc* doc,
      const GohubWChar* path)
```

Параметры:

`document`

Объект документа (GohubInformServicesDoc), который отправляется на сервер;

`path`

Путь, куда сохраняется файл (в кодировке UTF-16).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_close_inform_services_document

Закрытие документа информационных услуг (GohubInformServicesDoc) на диск, по указанному пути (с параметром вUTF-16).

Объявление:

```
GohubBool gohub_close_inform_services_document(
            GohubInformServicesDoc* document)
```

Параметры:

`document`

Объект документа (GohubInformServicesDoc).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_inform_services_document_id

Запрос идентификатора документа информационных услуг (GohubInformServicesDoc).

Объявление:

```
unsigned __int64 gohub_inform_services_document_id(
            GohubInformServicesDoc* doc)
```

Параметры:

`doc`

Объект документа (GohubInformServicesDoc).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_inform_services_document_revision

Запрос ревизии документа информационных услуг (GohubInformServicesDoc).

Объявление:

```
unsigned __int64 gohub_inform_services_document_revision(
            GohubInformServicesDoc* doc)
```

Параметры:

`doc`

Объект документа (GohubInformServicesDoc).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_inform_services_document_filename

Запрос имени документа информационных услуг (GohubInformServicesDoc).

Объявление:

```
const char* gohub_inform_services_document_filename(
            GohubInformServicesDoc* document)
```

Параметры:

`document`

Объект документа (GohubInformServicesDoc).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_inform_services_document_filename_w

Запрос имени документа информационных услуг (GohubInformServicesDoc) в UTF-16.

Объявление:

```
const GohubWChar* gohub_inform_services_document_filename_w(
      GohubInformServicesDoc* document)
```

Параметры:

`document`

Объект документа (GohubInformServicesDoc).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_inform_services_document_comment

Запрос комментария документа информационных услуг (GohubInformServicesDoc).

Объявление:

```
const char* gohub_inform_services_document_comment(
            GohubInformServicesDoc* document)
```

Параметры:

`document`

Объект документа (GohubInformServicesDoc).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_inform_services_document_comment_w

Запрос комментария документа информационных услуг (GohubInformServicesDoc) в UTF-16.

Объявление:

```
const GohubWChar* gohub_inform_services_document_comment_w(
```

```
            GohubInformServicesDoc* document)
```

Параметры:

`document`

Объект документа (GohubInformServicesDoc).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_inform_services_document_created_date

Запрос даты создания документа информационных услуг (GohubInformServicesDoc).

Объявление:

```
const char* gohub_inform_services_document_created_date(
            GohubInformServicesDoc* document)
```

Параметры:

`document`

Объект документа (GohubInformServicesDoc).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_inform_services_document_created_date_w

Запрос даты создания документа информационных услуг (GohubInformServicesDoc) в UTF-16.

Объявление:

```
const GohubWChar* gohub_inform_services_document_created_date_w(
      GohubInformServicesDoc* document)
```

Параметры:

`document`

Объект документа (GohubInformServicesDoc).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_inform_services_document_doc_date

Запрос даты документа информационных услуг (GohubInformServicesDoc).

Объявление:

```
const char* gohub_inform_services_document_doc_date(
            GohubInformServicesDoc* document)
```

Параметры:

`document`

Объект документа (GohubInformServicesDoc).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_inform_services_document_doc_date_w

Запрос даты документа информационных услуг (GohubInformServicesDoc) в UTF-16.

Объявление:

```
const GohubWChar* gohub_inform_services_document_doc_date_w(
      GohubInformServicesDoc* document)
```

Параметры:

`document`

Объект документа (GohubInformServicesDoc).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_inform_services_document_doc_is_empty

Запросить статус или пустой документ информационных услуг (GohubInformServicesDoc) в UTF-16.

Объявление:

```
const bool gohub_inform_services_document_doc_is_empty(
      GohubInformServicesDoc* document)
```

Параметры:

`document`

Объект документа (GohubInformServicesDoc).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

4.4.15. Обработка ошибок

gohub_last_error_code	Получить код ошибки последней операции
gohub_last_error_title	Получить название ошибки последней операции
gohub_last_error_title_w	Получить название ошибки последней операции (в UTF-16)
gohub_last_error_text	Получить текст ошибки последней операции
gohub_last_error_text_w	Получить текст ошибки последней операции (в UTF-16)

gohub_last_error_code

Запрос кода ошибки выполнения последней операции.

Объявление:

```
GohubErrcode gohub_last_error_code();
```

gohub_last_error_title

Запрос заголовка ошибки выполнения последней операции.

Объявление:

```
const char* gohub_last_error_title();
```

gohub_last_error_title_w

Запрос заголовка ошибки выполнения последней операции (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_last_error_title_w();
```

gohub_last_error_text

Запрос текста ошибки выполнения последней операции.

Объявление:

```
const char* gohub_last_error_text();
```

gohub_last_error_text_w

Запрос текста ошибки выполнения последней операции (в кодировке UTF-16).

Объявление:

```
const GohubWChar* gohub_last_error_text_w();
```

4.4.16. Работа с перечнем информации о заказе на согласование перевозки по данным календаря планирования перевозок зерновых грузов (за последние 5 дней от текущей даты).

gohub_query_dispatch_info	Получить объект с перечнем по коду станций
gohub_query_dispatch_info_w	Получить объект с перечнем по коду станций (в UTF-16)
gohub_document_info_description	Получить текст описания по индексу
gohub_document_info_description_w	Получить текст описания по индексу (в UTF-16)
gohub_document_info_count	Получить количество значений в списке
gohub_document_info_is_empty	Получить текст загруженности вагона
gohub_document_info_is_empty_w	Получить текст загруженности вагона (в UTF-16)
gohub_document_info_wag_owner	Получить владельца вагона
gohub_document_info_wag_owner_w	Получить владельца вагона (в UTF-16)
gohub_document_info_date	Получить дату
gohub_document_info_date_w	Получить дату (в UTF-16)
gohub_document_info_number	Получить номер
gohub_document_info_number_w	Получить номер (в UTF-16)
gohub_document_info_type	Получить тип
gohub_document_info_type_w	Получить тип(в UTF-16)
gohub_close_dispatch_info	Закрыть объект

gohub_query_dispatch_info

Запрос даты документа информационных услуг (GohubDispatchInfo)

Объявление:

```
GohubDispatchInfo* gohub_query_dispatch_info(GohubConnection*
connection, const char* start_esr, const char* end_esr);
```

Параметры:

connection

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции gohub_connect;

start_esr

Код станции отправления

end_esr

Код станции назначения

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_query_dispatch_info_w

Запрос даты документа информационных услуг (GohubDispatchInfo) (в кодировке UTF-16).

Объявление:

```
GohubDispatchInfo* gohub_query_dispatch_info(GohubConnection*
connection, const char* start_esr, const char* end_esr);
```

Параметры:

`connection`

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции `gohub_connect;`

`start_esr`

`Код станции отправления`(в кодировке UTF-16).

`end_esr`

`Код станции назначения`(в кодировке UTF-16).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_document_info_description

Запрос текста описания (GohubDispatchInfo)

Объявление:

```
const char*  gohub_document_info_description(GohubDispatchInfo*
document, int index);
```

Параметры:

`document`

Объект (GohubDispatchInfo), текст данных которого запрашивается;

`Index`

`Индекс элемента которого запрашивается данные;`

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_document_info_description_w

Запрос текста описания (GohubDispatchInfo) (в кодировке UTF-16).

Объявление:

```
const GohubWChar*
gohub_document_info_description_w(GohubDispatchInfo* document, int
index);
```

Параметры:

`document`

Объект (GohubDispatchInfo), текст данных которого запрашивается;

`Index`

Индекс элемента которого запрашивается данные(в кодировке UTF-16).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_document_info_count

Запрос количества элементов в списке

Объявление:

```
int gohub_document_info_count(GohubDispatchInfo* document);
```

Параметры:

`document`

Объект (GohubDispatchInfo), текст данных которого запрашивается;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_document_info_is_empty

Запрос текста загруженности вагона (GohubDispatchInfo)

Объявление:

```
const char*  gohub_document_info_is_empty(GohubDispatchInfo*
document, int index);
```

Параметры:

`document`

Объект (GohubDispatchInfo), текст данных которого запрашивается;

`Index`

Индекс элемента которого запрашивается данные;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_document_info_is_empty_w

Запрос текста загруженности вагона (GohubDispatchInfo) (в кодировке UTF-16).

Объявление:

```
const GohubWChar*  gohub_document_info_is_empty_w(GohubDispatchInfo* document, int index);
```

Параметры:

document

Объект (GohubDispatchInfo), текст данных которого запрашивается;

Index

Индекс элемента которого запрашивается данные(в кодировке UTF-16).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_document_info_wag_owner

Запрос текста владельца вагона (GohubDispatchInfo)

Объявление:

```
const char*  gohub_document_info_wag_owner(GohubDispatchInfo* document, int index);
```

Параметры:

document

Объект (GohubDispatchInfo), текст данных которого запрашивается;

Index

Индекс элемента которого запрашивается данные;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_document_info__wag_owner_w

Запрос текста владельца вагона (GohubDispatchInfo) (в кодировке UTF-16).

Объявление:

```
const GohubWChar*  gohub_document_info_wag_owner_w(GohubDispatchInfo* document, int index);
```

Параметры:

document

Объект (GohubDispatchInfo), текст данных которого запрашивается;

Index

Индекс элемента которого запрашивается данные(в кодировке UTF-16).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_document_info_date

Запрос даты (GohubDispatchInfo)

Объявление:

const char* gohub_document_info_date(GohubDispatchInfo* document, int index);

Параметры:

document

Объект (GohubDispatchInfo), текст данных которого запрашивается;

Index

Индекс элемента которого запрашивается данные;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_document_info_date_w

Запрос даты (GohubDispatchInfo) (в кодировке UTF-16).

Объявление:

const GohubWChar* gohub_document_ info_date_w(GohubDispatchInfo* document, int index);

Параметры:

document

Объект (GohubDispatchInfo), текст данных которого запрашивается;

Index

Индекс элемента которого запрашивается данные(в кодировке UTF-16).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error и др.*

gohub_document_info_number

Запрос номера (GohubDispatchInfo)

Объявление:

const char* gohub_document_info_number(GohubDispatchInfo* document, int index);

Параметры:

document

Объект (GohubDispatchInfo), текст данных которого запрашивается;

Index

Индекс элемента которого запрашивается данные;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_document_info_number_w

Запрос номера (GohubDispatchInfo) (в кодировке UTF-16).

Объявление:

const GohubWChar* gohub_document_ info_number_w(GohubDispatchInfo* document, int index);

Параметры:

document

Объект (GohubDispatchInfo), текст данных которого запрашивается;

Index

Индекс элемента которого запрашивается данные(в кодировке UTF-16).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_document_info_type

Запрос текста тип перевозки (GohubDispatchInfo)

Объявление:

const char* gohub_document_info_type(GohubDispatchInfo* document, int index);

Параметры:

document

Объект (GohubDispatchInfo), текст данных которого запрашивается;

Index

Индекс элемента которого запрашивается данные;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_document_info_type_w

Запрос текста тип перевозки (GohubDispatchInfo) (в кодировке UTF-16).

Объявление:

const GohubWChar* gohub_document_ info_type_w(GohubDispatchInfo* document, int index);

Параметры:

document

Объект (GohubDispatchInfo), текст данных которого запрашивается;

Index

Индекс элемента которого запрашивается данные(в кодировке UTF-16).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_close_dispatch_info

Закрыть объект(GohubDispatchInfo) .

Объявление:

GohubBool gohub_close_dispatch_info(GohubDispatchInfo* document);

Параметры:

document

Объект (GohubDispatchInfo), текст данных которого запрашивается;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

4.4.17. Работа с перечнем информации о № заказа на Согласования сокращения сроком доставки.

gohub_query_pstd_info	Получить объект с перечнем по коду станций и вида роботы станции
gohub_query_pstd_info_w	Получить объект с перечнем по коду станций и вида роботы станции (в UTF-16)
gohub_pstd_info_arrivaldate	Получить дату прибытия по индексу(в UTF-16)
gohub_pstd_info_arrivaldate_w	Получить дату прибытия по индексу (в UTF-16)
gohub_pstd_info_count	Получить количество значений в списке
gohub_pstd_info_departuredate	Получить дату отправления по индексу (в UTF-16)
gohub_pstd_info_departuredate_w	Получить дату отправления по индексу (в UTF-16)
gohub_pstd_info_reqdate	Получить дату заказа по индексу (в UTF-16)
gohub_pstd_info_reqdate_w	Получить дату заказа по индексу (в UTF-16)
gohub_pstd_info_number	Получить номер по индексу
gohub_pstd_info_number_w	Получить номер по индексу (в UTF-16)
gohub_close_pstd_info	Закрыть объект

gohub_query_pstd_info

Запрос перечня по коду станций и виду роботы станции (GohubPSTDInfo)

Объявление:

```
GohubPSTDInfo* gohub_query_pstd_info(GohubConnection* connection, const char* work_kind, const char* departure_esr, const char* arrival_esr);
```

Параметры:

connection

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции gohub_connect;

departure_esr

Код станции отправления

arrival_esr

Код станции назначения

work_kind

Код вида роботы станции

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_query_pstd_info_w

Запрос перечня по коду станций и виду роботы станции (GohubPSTDInfo)

(в кодировке UTF-16).

Объявление:

```
GohubPSTDInfo* gohub_query_pstd_info_w(GohubConnection* connection, const GohubWChar* work_kind, const GohubWChar* departure_esr, const GohubWChar* arrival_esr);
```

Параметры:

connection

Указатель соединения с *Модулем Согласования*, полученный ранее при помощи функции gohub_connect;

departure_esr

Код станции отправления (в кодировке UTF-16).

arrival_esr

Код станции назначения(в кодировке UTF-16).

work_kind

Код вида роботы станции(в кодировке UTF-16).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_pstd_info_count

Запрос количества элементов в списке

Объявление:

```
int gohub_pstd_info_count(GohubPSTDInfo * document);
```

Параметры:

document

Объект (GohubPSTDInfo)

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_pstd_info_arrivaldate

Запрос даты прибытия (GohubPSTDInfo)

Объявление:

const char* gohub_pstd_info_arrivaldate(GohubPSTDInfo* document, int index);

Параметры:

document

Объект (GohubPSTDInfo), текст данных которого запрашивается;

Index

Индекс элемента которого запрашивается данные;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_pstd_info_arrivaldate_w

Запрос даты прибытия (GohubPSTDInfo) (в кодировке UTF-16).

Объявление:

const GohubWChar* gohub_pstd_info_arrivaldate_w(GohubPSTDInfo* document, int index);

Параметры:

document

Объект (GohubPSTDInfo), текст данных которого запрашивается;

Index

Индекс элемента которого запрашивается данные(в кодировке UTF-16).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_pstd_info_departuredate

Запрос даты отправления (GohubPSTDInfo)

Объявление:

const char* gohub_pstd_info_departuredate(GohubPSTDInfo* document, int index);

Параметры:

document

Объект (GohubPSTDInfo), текст данных которого запрашивается;

Index

Индекс элемента которого запрашивается данные;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_pstd_info_departuredate_w

Запрос даты отправления (GohubPSTDInfo) (в кодировке UTF-16).

Объявление:

const GohubWChar* gohub_pstd_info_departuredate_w(GohubPSTDInfo* document, int index);

Параметры:

document

Объект (GohubPSTDInfo), текст данных которого запрашивается;

Index

Индекс элемента которого запрашивается данные(в кодировке UTF-16).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_pstd_info_number

Запрос номера (GohubPSTDInfo)

Объявление:

const char* gohub_pstd_info_number(GohubPSTDInfo * document, int index);

Параметры:

document

Объект (GohubPSTDInfo), текст данных которого запрашивается;

Index

Индекс элемента которого запрашивается данные;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_document_info_number_w

Запрос номера (GohubPSTDInfo) (в кодировке UTF-16).

Объявление:

const GohubWChar* gohub_pstd_ info_number_w(GohubDispatchInfo* document, int index);

Параметры:

document

Объект (GohubPSTDInfo), текст данных которого запрашивается;

Index

Индекс элемента которого запрашивается данные(в кодировке UTF-16).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_pstd_info_reqdate

Запрос даты заказа (GohubPSTDInfo)

Объявление:

const char* gohub_pstd_info_reqdate(GohubPSTDInfo* document, int index);

Параметры:

document

Объект (GohubPSTDInfo), текст данных которого запрашивается;

Index

Индекс элемента которого запрашивается данные;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_pstd_info_reqdate_w

Запрос даты заказа (GohubPSTDInfo) (в кодировке UTF-16).

Объявление:

const GohubWChar* gohub_pstd_info_reqdate_w(GohubPSTDInfo* document, int index);

Параметры:

document

Объект (GohubPSTDInfo), текст данных которого запрашивается;

Index

Индекс элемента которого запрашивается данные(в кодировке UTF-16).

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

gohub_close_pstd_info

Закрыть объект(GohubPSTDInfo) .

Объявление:

GohubBool gohub_close_pstd_info(GohubPSTDInfo* document);

Параметры:

`document`

Объект (GohubPSTDInfo), текст данных которого запрашивается;

Результат:

В случае успеха – значение отличное от нуля. В случае ошибки – 0. Информацию об ошибке можно получить с помощью функции *gohub_last_error* и др.

4.5. Коды ошибок

Коды ошибок в *Библиотеке* представлены перечислением (*enum*) `GohubErrcode`, содержащим набор констант, семантически соответствующих возможным типам ошибок, возникающих при работе с *Библиотекой*.

Сводная таблица кодов ошибок

`gohub_success`	Операция выполнена успешно
`gohub_server_connection_could_not_be_established`	Ошибка при открытии соединения с *Модулем Согласования*
`gohub_server_inaccessible`	Служба *Модуля Согласования* недоступна
`gohub_document_creation_failed`	Ошибка открытия файла
`gohub_document_query_failed`	Ошибка при запросе документа из АС «Клиент УЗ» по идентификатору
`gohub_next_document_query_failed`	Ошибка при запросе документа из АС «Клиент УЗ» по номеру ревизии
`gohub_document_sending_failed`	Ошибка при отправке документа в АС «Клиент УЗ»
`gohub_document_saving_failed`	Ошибка при сохранении документа в файл
`gohub_document_loading_failed`	Ошибка при загрузке документа из файла
`gohub_invalid_code_page`	Недействительная кодовая страница
`gohub_private_key_could_not_be_opened`	Ошибка при открытии электронного ключа
`gohub_private_key_path_could_not_be_opened`	Ошибка при открытии электронного ключа по указанному пути
`gohub_private_key_bytes_could_not_be_opened`	Ошибка при открытии электронного ключа массивом байт
`gohub_private_key_is_inaccessible`	Электронный ключ недоступен. Возможно, устройство отсоединили от компьютера
`gohub_document_could_not_be_signed`	Ошибка при наложении электронно-цифровой подписи
`gohub_document_signature_verification_failed`	Электронно-цифровая подпись недействительна, возможно, данные были повреждены
`gohub_document_has_not_signature`	Документ не содержит электронно-цифровую подпись

gohub_client_is_obsolete	Версия клиента устарела, обновите
gohub_server_is_obsolete	Версия сервера устарела, обновите
gohub_document_reclamation_failed	Отзыв документа не удался
gohub_document_deletion_failed	Удалить документ не удалось
gohub_attachment_creation_failed	Создать сопроводительный документ не удалось
gohub_attachment_sending_failed	Отправить сопроводительный документ не удалось
gohub_attachment_query_failed	Запросить сопроводительный документ не удалось
gohub_attachment_deletion_failed	Удалить сопроводительный документ не удалось
gohub_attaching_to_document_failed	Прикрепить сопроводительный документ к перевозному документу не удалось
gohub_detaching_from_document_failed	Открепить сопроводительный документ от перевозного документа не удалось
gohub_mount_of_file_key_failed	Монтирование электронного ключа из файла не удалось.
gohub_unmount_of_file_key_failed	Демонтирование электронного ключа из файла не удалось.
gohub_enumerating_of_file_keys_failed	Получить список монтированных из файлов электронных ключей не удалось.
gohub_mounted_file_key_inaccessible	Монтированный из файла электронный ключ не доступен.
gohub_edata_creating_failed	Ошибка при создании ЭД предварительного информирования
gohub_edata_sending_failed	Ошибка при отправке ЭД предварительного информирования в АС «Клиент УЗ»
gohub_edata_updating_failed	Ошибка при обновлении ЭД предварительного информирования в АС «Клиент УЗ»
gohub_edata_query_failed	Ошибка при запросе ЭД предварительного информирования из АС «Клиент УЗ» по идентификатору
gohub_next_edata_query_failed	Ошибка при запросе ЭД предварительного информирования из АС «Клиент УЗ» по ревизии
gohub_pipackage_query_failed	Ошибка при запросе пакета ПИ из АС «Клиент УЗ» по идентификатору
Gohub_next_pipackage_query_failed	Ошибка при запросе пакета ПИ из АС «Клиент УЗ» по ревизии
gohub_mp_months_query_failed	Ошибка при запросе перечня идентификаторов месяцев из АС «Месплан»
gohub_orders_of_months_query_failed	Ошибка при запросе заявок из АС «Месплан»
gohub_inform_services_doc_saving_failed	Ошибка при сохранении документа информационных услуг

gohub_inform_services_doc_query_failed	Ошибка при запросе документа информационных услуг
gohub_query_changes_inform_services_failed	Ошибка при запросе изменений по документам информационных услуг
gohub_query_next_inform_services_document_failed	Ошибка при запросе следующего документа информационных услуг
gohub_programm_error = 0x1000	Неизвестная ошибка в программе
gohub_invalid_operation	Программа пользователя выполнила недопустимую операцию

Более детальное описание программного интерфейса библиотеки можно получить из заголовочного файла gohub.client.errors.h (Приложение Б).

4.6. Примеры использования

а) Вывод на экран информации об ошибке

Приведем для начала пример получения информации об ошибке. Определим функцию `print_error`, которая выводит на экран подробную информацию об ошибке, включая информацию о вложенных ошибках, если таковые имеются. Эта функция пригодится нам в следующих примерах.

```
#include <stdio.h>
#include "gohub.client.h"

void print_last_error()
{
  printf("Error [%04x] -- %s -- %s\n",
    gohub_last_error_code(),
    gohub_last_error_title(),
    gohub_last_error_text());
}
```

б) Загрузка документа из файла и передача в АС «Клиент УЗ»

А теперь проиллюстрируем загрузку документа из файла и передачу его в АС «Клиент УЗ». При этом используем электронный ключ для наложения электронно-цифровой подписи. Для вывода информации о возможных ошибках используем функцию `print_error`, определенную ранее в п.4.5.а.

```
#include "gohub.client.h"
#include <stdio.h>
#include <string.h>

void print_last_error();

void load_and_send_document(
  const char* host,           // IP адрес или имя компьютера Модуля Согласования
  int port,                   // номер TCP порта для подключения
  const char* path,           // путь к файлу документа
  const char* password) // Пароль к ключу. Если не задан, документ не будет
подписан
{
  GohubConnection* connection;
  GohubDocument* document;
  char document_id[256];

  // Создать подключение к Модулю Согласования
  printf("Connectiong to %s:%d\n", host, port);
  connection = gohub_connect(host, port);
```

```
  if(connection)
  { // Подключение создано успешно
    puts("Connection opened");

    // Загрузить документ из файла
    document = gohub_load_document(path);

    if(document)
    { // Документ загружен успешно
      printf("Document is loaded: size=%d bytes\n",
gohub_document_size(document));
      if(password)
      { //Подписать документ
        puts("Signing document . . .");
        if(gohub_open_private_key(connection, password))
        {
          printf("Key owner name: %s\n",
gohub_private_key_owner_name(connection));
          printf("Private key info: %s\n",
gohub_private_key_owner_info(connection));
          if(gohub_sign_document(connection, document))
            puts("Document signed");
          else
          {
            print_last_error();
            return;
          }
        }
        else
        { //Не удалось получить доступ к закрытому ключу
          print_last_error();
          return;
        }
      }
      // Передать документ в АС "Клиент УЗ" через Модуль Согласования
      if(gohub_send_document(connection, document))
      { // Документ передан успешно
        printf("Document is sended: id=%s revision=%d\n",
          gohub_document_id(document),
          gohub_document_revision(document));
        // Проверка наличия предупреждений в документе после отправки
        const char* str = gohub_document_warning(doc);
            if (str != NULL)
                printf("Warning in document: %s\n", str);
        // Сохраняем идентификатор документа
        // для дальнейшего использования
        strcpy(document_id, gohub_document_id(document));
      }
      else
      { // Ошибка при передаче документа
        print_last_error();
      }
      // Завершить работу с документом
      gohub_close_document(document);
            // Попробуем запросить этот же документ по идентификатору
            document = gohub_query_document(connection, document_id);
            if(document)
            { // Запрос документа выполнен успешно
              // Выводим документ на эран в кодировке windows-1251
              puts(gohub_document_text(document));
              // Завершить работу с документом
              gohub_close_document(document);
            }
            else
```

```
            { // Ошибка при получении документа
              print_last_error();
            }
    }
    else
    { // Ошибка при загрузке документа из файла
      print_last_error();
    }
    // Закрыть подключение к Модуля Согласования
    gohub_disconnect(connection);
  }
  else
  { // Ошибка при подключении к Модулю Согласования
    print_last_error();
  }
}
```

в) Запрос документов из АС «Клиент УЗ»

В этом примере показана техника последовательного запроса документов их АС «Клиент УЗ» по номерам ревизий. Кроме того показано изменение кодовой страницы документа и сохранение запрошенных документов в файлы. Кроме того, выполняется проверка электронно-цифровой подписи. Для вывода информации об ошибках по-прежнему используется функция `print_error`, определенную ранее в п.4.5.а

```
#include "gohub.client.h"
#include <stdio.h>
#include <stdlib.h>

void print_last_error();

int query_and_save_documents(
  const char* host,                 // IP адрес или имя компьютера Модуля
Согласования
  int port,                         // номер TCP порта для подключения
  int startRevision,          // номер ревизии с которой получать перевозочные
документы
  int maxCount,                     // Максимальное количество документов которое
будет запрашиваться
  const char* targetFolder)   // папка куда сохранять полученные документы
{
  int lastRevision = startRevision;
  GohubConnection* connection;
  GohubDocument* document;
  int count;

  // Создать подключение к Модулю Согласования
  printf("Connectiong to %s:%d", host, port);
  connection = gohub_connect(host, port);

  if(connection)
  { // Подключение создано успешно
    puts("Connection opened");

    for (count = 0; count < maxCount; ++count)
    { // Запрос следующего документа по ревизии
      document = gohub_query_next_document(connection, lastRevision);

      if(document)
      { // Запрос документа выполнен успешно
        printf("Document received: id=%s revision=%d\n",
          gohub_document_id(document),
          gohub_document_revision(document));
```

```
        // Запомнить ревизию последнего документа
        lastRevision = gohub_document_revision(document);

        // Вывести документ на экран
        wprintf(L"%s\n", gohub_document_text_w(document));
        { // Сохранить документ в другой кодировке.
          char path[_MAX_PATH];
          sprintf(path, "%s\\%s.xml", targetFolder, gohub_document_id(document));
          if(gohub_save_document(document, path, 866))
          { // Документ сохранен успешно
            printf("Document saved to file: path='%s'\n", path);
          }
          else
          { // Ошибка при сохранении документа
            print_last_error();
            gohub_close_document(document);
                  continue;
          }
        }

        if(gohub_document_has_signature(document))
        { //Проверка ЭЦП
          if(gohub_document_check_signature(document))
          {
            wprintf(L"%s\n", gohub_document_signer_name_w(document));
            wprintf(L"%s\n", gohub_document_signer_info_w(document));
             wprintf(L"%s\n", gohub_document_sign_time_w(document));
          }
          else
          {
            print_last_error();
          }
        }
        // Завершение работы с документом
        gohub_close_document(document);
      }
      else
      { // Не удалось полчить следующий по ревизии документ.
        // Это либо ошибка, либо новых документов больше нет пока.
        if(gohub_last_error_code() != gohub_success)
          print_last_error();
        break;
      }
    }
    // Закрыть подключение к Модулю Согласования
    gohub_disconnect(connection);
    puts("Connection closed");
  }
  else
  { // Ошибка при подключении к Модулю Согласования
    print_last_error();
  }
  // Возврат вызывающей функции номера последней обработанной ревизии
  return lastRevision;
}
```

г) Другие примеры использования

В инсталляционном пакете Клиента Модуля Согласования представлены также другие примеры использования, с которыми более детально можно ознакомиться, заглянув непосредственно в программный код (папка Клиента Модуля Согласовния\samples\ gohub.client.test.c). Здесь мы кратко опишем другие функции, представленные в примерах:

- Показана техника последовательного запроса документов ФДУ-92 из АС «Клиент УЗ» по номерам ревизий

```
int query_and_save_fdu92s(
  const char* host,                 // IP адрес или имя компьютера Модуля
Согласования
  int port,                         // номер TCP порта для подключения
  int startRevision,          // номер ревизии с которой получать перевозочные
документы
  int maxCount,                     // Максимальное количество документов которое
будет запрашиваться
  const char* targetFolder);// папка куда сохранять полученные документы
```

- Показана техника последовательного запроса документов ЭД предварительного информирования из АС «Клиент УЗ» по номерам ревизий

```
__int64 query_and_save_edatas(
  const char* host,                 // IP адрес или имя компьютера Модуля
Согласования
  int port,                         // номер TCP порта для подключения
  __int64 startRevision,      // номер ревизии с которой получать ЭД (электронные
данные) предварительного информирования
  int maxCount,                     // Максимальное количество документов ЭД
которое будет запрашиваться
  const char* targetFolder);// папка куда сохранять полученные документы ЭД
```

- Показана техника последовательного запроса пакетов предварительного информирования из АС «Клиент УЗ» по номерам ревизий

```
__int64 query_and_save_pi_packages(
  const char* host,                 // IP адрес или имя компьютера Модуля
Согласования
  int port,                         // номер TCP порта для подключения
  __int64 startRevision,      // номер ревизии с которой получать пакеты ПИ
(предварительного информирования)
  int maxCount,                     // Максимальное количество пакеты ПИ которое
будет запрашиваться
  const char* targetFolder);// папка куда сохранять полученные пакеты ПИ
```

- Показана техника добавления документа ЭД в пакет предварительного информирования

```
void add_edata_to_pi_package(
  const char* host,                 // IP адрес или имя компьютера Модуля
Согласования
  int port,                         // номер TCP порта для подключения
  const char* piPackageId,    // ID пакета ПИ (предварительного информирования) в
который будут добавляться ЭД (электронные данные)
  const char* edataPath,      // путь к xml-файлу с ЭД
  int edataType,              // код типа ЭД: 190 счет-фактура, 320 упаковочный
лист
  char *newEdataId);                // сюда будет получен ID созданного документа
ЭД
```

- Показана техника обновления документа ЭД содержащегося на сервере АС «Клиент УЗ»

```
__int64 update_edata(
  const char* host,                 // IP адрес или имя компьютера Модуля
Согласования
  int port,                         // номер TCP порта для подключения
  const char* edataId,        // ID документа ЭД который будет меняться
```

```
  const char* edataPath);     // путь к xml-файлу с ЭД
```

- Показана техника работы с перечнем информации о заказе на согласование перевозки по данным календаря планирования перевозок зерновых грузов (за последние 5 дней от текущей даты).

```
try
{
      printf(" Test Query Dispatch Info(C++)\n");
      printf(" Please enter ESR number of start station: ");
      char start_esr[101];
      scanf("%s", start_esr);
      printf(" Please enter ESR number of end station: ");
      char end_esr[101];
      scanf("%s", end_esr);
      GohubDispatchInfo* str = gohub_query_dispatch_info(connection,
      start_esr, end_esr);
      check_errors();
      printf("\nlist loaded.\n");
      for(int i = 0; i < gohub_document_info_count(str); i++)
      {
            printf("%s\n", gohub_document_info_description(str, i));
            printf("%s\n", gohub_document_info_number(str, i));
            printf("%s\n", gohub_document_info_date(str, i));
            printf("%s\n", gohub_document_info_type(str, i));
            printf("%s\n", gohub_document_info_wag_owner(str, i));
            printf("%s\n", gohub_document_info_is_empty(str, i));
      }
      gohub_close_dispatch_info(str);
      printf("----------------\n");
}
catch (...)
{
      printf("Exception");
}
```

- Показана техника получения перечня подписей из ПД

```
GohubDocument* document = gohub_query_document(connection, id):
for (int i = 0; i < gohub_document_signature_count(document); i++)
{
      printf("\n %d) Document received: id=%d\n %s\n", i + 1, id,
      gohub_document_signer_info_by_index(document, i));
      printf("%s\n", gohub_document_signer_name_by_index(document, i));
      printf("%s\n", gohub_document_sign_time_by_index(document, i));
      printf("%d\n", gohub_document_sign_status_by_index(document, i));
      printf("%s\n", gohub_document_sign_error_by_index(document, i));
}
gohub_close_document(document);
```

- Показана техника получения перечня доступных носителей и открытие электронного ключа за индексом из списка

```
try
      {
            printf(" CryptoMedia Name :%s", gohub_keys_list(connection));
            check_errors();
            printf(" Please enter Media index: ");
            int mediaIndex;
            scanf("%d", &mediaIndex);
            printf(" Please enter Device index: ");
```

```
            int deviceIndex;
            scanf("%d", &deviceIndex);
            printf(" Please enter password: ");
            char password[101];
            scanf("%s", password);
if(gohub_open_private_key_by_index(connection, password, mediaIndex,
deviceIndex))
        {
          printf("Key owner name: %s\n",
gohub_private_key_owner_name(connection));
          printf("Private key info: %s\n",
gohub_private_key_owner_info(connection));
            }

            printf("%s\n", gohub_last_error_text());
      }
      catch (...)
      {
            printf("Exception");
      }
```

5. .NET библиотека - TMSoft.Gohub.Client.Net.dll

Программный интерфейс *Библиотеки* представлен набором типов, список которых приведен ниже. Все типы библиотеки определены в пространстве имен *TMSoft.Gohub.Client*.

5.1. Список типов

GohubConnection	Класс соединений с Сервером модуля согласования
GohubDocument	Класс перевозочных документов
GohubAttachment	Класс сопроводительных документов
GohubEData	Класс сопроводительных документов
GohubInformServicesDoc	Класс документов информационных услуг
GohubSigner	Класс информации о владельцах электронных ключей и авторах электронно-цифровых подписей
GohubDocumentFilter	Класс информации о состоянии фильтра запроса документов
GohubException	Класс исключений
GohubErrCode	Перечисление кодов ошибок
DispatchInfo	Класс № Заказа для ПД
PSTDInfo	Класс № Заказа Согласования сокращения сроком доставки
CryptoMedia	Класс списка типов носителей электронных ключей
CryptoDevices	Класс названия ключа в списке носителей

5.2. Состояния перевозочного документа

Коды перевозочного документа в *Библиотеке* представлены перечислением (*enum*) DocumentStatus, содержащим набор констант.

Сводная таблица кодов состояний перевозочного документа

Unknown = 0	Статус неизвестен
Draft = 1	Черновик
Sending = 2	Документ передается товарному кассиру
Registered = 3	Документ передан товарному кассиру
Reclaiming = 4	Документ отзывается от товарного кассира
Accepted = 5	Груз принят к перевозке
Delivered = 6	Груз прибыл
Recieved = 7	Груз получен получателем
Uncredited = 8	Документ раскредитован товарным кассиром
RecDraft = 9	Груз получен получателем и редактируется
RecSending = 10	Груз получен получателем и передан товарному кассиру
RecReclaiming = 11	Груз получен получателем и отзывается от товарного кассира
Canceled = 12	Документ испорчен товарным кассиром

5.3. GohubConnection

Класс GohubConnection служит для представления соединений с Сервером модуля согласования и содержит следующие члены:

GohubConnection (конструктор)	Создать соединение с Сервером модуля согласования. Для закрытия соединения необходимо использовать метод Dispose.
DocumentFilter (свойство)	Экземпляр класса GohubDocumentFilter, позволяющий устанавливать фильтры длязапроса документов.
SignerInfo (свойство)	Информация о владельце электронного, открытого при помощи метода OpenPrivateKey. В случае, если устройство электронного ключа было отключено, возбуждается исключение GohubException с кодом ошибки GohubErrCode.gohub_private_key_is_inaccessible. В случае, если электронный ключ не был открыт при помощи метода OpenPrivateKey, возбуждается исключение GohubException с кодом ошибки GohubErrCode.gohub_invalid_operation
ClosePrivateKey (метод)	Закрыть электронный ключ, открытый ранее с помощью метода OpenPrivateKey. Электронный ключ автоматически закрывается также при закрытии соединения. с Сервером модуля согласования.
Dispose (метод)	Закрыть соединение с Сервером модуля согласования. Автоматически также закрывается электронный ключ, если он был открыт.
OpenPrivateKey (метод)	Открыть электронный ключ. Если электронный ключ был открыт уже ранее, он автоматически закрывается. Электронный ключ автоматически закрывается также при закрытии соединения с Сервером модуля согласования. Для закрытия электронного ключа без разрыва соединения с Сервером модуля согласования можно использовать метод ClosePrivateKey. Электронный ключ нужен для подписания электронных документов. Для проверки электронно-цифровой подписи электронный ключ не используется и его можно не открывать. Возможны следующие варианты открытия электронного ключа: 1) По паролю. Ключ должен быть примонтирован. 2) По указанному пути и паролю. 3) Ключ представлен массивом байт. Необходимо указать пароль и длину массива байт.
QueryDocument (метод)	Запросить документ по уникальному идентификатору.
QueryDocuments (метод)	Получить упорядоченную последовательность документов, с номерами ревизии больше заданного
QueryDocuments2 (метод)	Получить упорядоченную последовательность документов, с номерами ревизии больше заданного
QueryDocumentPrintableForm (метод)	Запросить печатную форму документа по его ID. Возвращает массив байт.
QueryAndSaveDocumentPrintableForm (метод)	Запросить печатную форму документа по его ID. Результат запроса сохраняется в файл по пути указанному в параметре path.
SendDocument (метод)	Отправить документ в АС «Клиент УЗ»
SignDocument (метод)	Подписать документ электронно-цифровой подписью
ReclaimDocument (метод)	Отзыв документа с сервера СГР
DeleteDocument (метод)	Удаление документа на АС «Клиент УЗ» по его ID
SendAttachment (метод)	Отправить сопроводительный документ в АС «Клиент УЗ»
QueryAttachment (метод)	Запросить сопроводительный документ из системы АС «Клиент УЗ»
QueryAttachmentWithUserData	Запросить сопроводительный документ с электронными

(метод)	данными пользователя из системы АС «Клиент УЗ»
DeleteAttachment (метод)	Удалить сопроводительный документ из АС «Клиент УЗ»
QueryEData (метод)	Запросить электронные данные ПИ из системы АС «Клиент УЗ»
QueryEDatas (метод)	Запросить электронные данные ПИ из системы АС «Клиент УЗ» по ревизии
SendEData (метод)	Отправить электронные данные ПИ в АС «Клиент УЗ»
AddEDataToPiPackage (метод)	Добавить электронные данные ПИ в АС «Клиент УЗ»
UpdateEData (метод)	Обновить имеющиеся электронные данные ПИ в АС «Клиент УЗ»
QueryPiPackage (метод)	Запросить пакет ПИ из системы АС «Клиент УЗ»
QueryPiPackages (метод)	Запросить пакеты ПИ из системы АС «Клиент УЗ» по ревизии
SendReceivedDocument (метод)	Отправить документ по прибытию
QueryMPMonths (метод)	Запросить перечень идентификаторов месяцев из АС «Месплан» в виде списка
QueryMPMonthsString (метод)	Запросить перечень идентификаторов месяцев из АС «Месплан» в виде строки (номера месяцев отделены пробелом)
QueryOrdersForMonth (метод)	Запросить заявки по номеру месяца из АС «Месплан». Возможен запрос с указанием логина и пароля.
QueryAndSaveOrdersForMonth (метод)	Запросить и сохранить заявки по номеру месяца в xml. Возможен запрос с указанием логина и пароля.
QueryEDataForAttachment (метод)	Запросить электронные данные ПИ из системы АС «Клиент УЗ» по идентификатору сопроводительного документа
QueryAndSaveFdu92PrintableForm (метод)	Запросить печатную форму ФДУ-92 по его ID. Результат запроса сохраняется в файл по пути указанному в параметре path.
QueryFdu92PrintableForm (метод)	Запросить печатную форму ФДУ-92 по его ID. Возвращает массив байт.
QueryAndSaveGu46PrintableForm (метод)	Запросить печатную форму ГУ-46 по его ID. Результат запроса сохраняется в файл по пути указанному в параметре path.
QueryGu46PrintableForm (метод)	Запросить печатную форму ГУ-46 по его ID. Возвращает массив байт.
QueryAndSaveGu45PrintableForm (метод)	Запросить печатную форму ГУ-45 по его ID. Результат запроса сохраняется в файл по пути указанному в параметре path.
QueryGu45PrintableForm (метод)	Запросить печатную форму ГУ-45 по его ID. Возвращает массив байт.
QueryFdu92 (метод)	Запросить ФДУ-92 по уникальному идентификатору.
QueryFdu92s (метод)	Получить упорядоченную последовательность ФДУ-92, с номерами ревизии больше заданного
QueryFdu92_ByNumber (метод)	Запросить ФДУ-92 по номеру станции и номеру накопительной карточки.
SignFdu92 (метод)	Подписать накопительную карточку электронно-цифровой подписью ФДУ-92
SendFdu92 (метод)	Отправить ФДУ-92 в АС «Клиент УЗ»
QueryGu46 (метод)	Запросить ГУ-46 по уникальному идентификатору.
QueryGu46s (метод)	Получить упорядоченную последовательность ГУ-46, с номерами ревизии больше заданного
SignGu46 (метод)	Подписать накопительную карточку электронно-цифровой подписью ГУ-46

SendGu46 (метод)	Отправить ГУ-46 АС «Клиент УЗ»
QueryGu45 (метод)	Запросить ГУ-45 по уникальному идентификатору.
QueryGu45s (метод)	Получить упорядоченную последовательность ГУ-45, с номерами ревизии больше заданного
QueryInformServicesDoc (метод)	Запросить документ информационных услуг по уникальному идентификатору.
QueryChangesInformServices (метод)	Получить упорядоченную последовательность документов информационных услуг, с номерами ревизии больше заданного
QueryDispatchInfo (метод)	Получить перечень информации о заказе на согласование перевозки по данным календаря планирования перевозок зерновых грузов (за последние 5 дней от текущей даты).
QueryGu45_ByNumber (метод)	Запросить ГУ-45 за номером станции, номером памятки и датой формирования
QueryGu46_ByNumber (метод)	Запросить ГУ-46 за номером станции и номером ведомости
GetKeysList (свойство)	Запросить список носителей и ключей для открытия электронного ключа по индексу

5.4. GohubDocument

Класс GohubDocument служит для представления электронных перевозочных документов и содержит следующие члены:

HasSignature (свойство)	Признак наличия у документа электронно-цифровой подписи
Id (свойство)	Уникальный идентификатор документа
Revision (свойство)	Номер ревизии документа
Status (свойство)	Статус документа
SignerInfo (свойство)	Информация о лице, подписавшем документ электронно-цифровой подписью. Если проверка электронно-цифровой подписи не подтверждает достоверность данных, результатом будет null. Если документ не содержит электронно-цифровой подписи, возбуждается исключение GohubException с кодом ошибки GohubErrCode.gohub_document_has_not_signature.
TimeStamp (свойство)	Метка времени, полученная при подписании документа Электронно-цифровой подписью. Если проверка электронно-цифровой подписи не подтверждает достоверность данных, возбуждается исключение GohubException с кодом ошибки GohubErrCode.gohub_document_signature_verification_failed Если документ не содержит электронно-цифровой подписи, возбуждается исключение GohubException с кодом ошибки GohubErrCode.gohub_document_has_not_signature.
Attachments (свойство)	Перечисление ID сопроводительных документов, присоединенных к этому документу.
MeasureEquipNum (свойство get; set;)	Сведения вагоноизмерительной техники.
BusinessUnitNum (свойство get; set;)	Номер филиала ЧАО УЗ.
SetVerifiedEmptyWeightForWagon (метод)	Установить уточненный вес тары вагона *int wagonIndex* – индекс вагона уточненный вес которого необходимо установить. Индексация вагонов начинается с нуля.

	int weight – уточненный вес вагона.
GetVerifiedEmptyWeightForWagon (метод)	Получить уточненный вес тары вагона *int wagonIndex* – индекс вагона уточненный вес которого необходимо получить. Индексация вагонов начинается с нуля.
ForeignNotAccept (свойство get;)	Получить отметку возвращения непринятых пограничными станциями иностранных железных дорог вагонов на территорию Украины
WarrantType (свойство get; set;)	Тип оснований для получения груза (0 - доверенность, 1 - приказ)
Deserialize (методы)	Десериализовать документ из массива байт, потока ввода, TextReader или XmlReader. В случае десериализации из массива байт или потока ввода Xml-документ, должен начинаться с xml-заголовка с указанием кодировки xml-документа. Фактическая кодировка символов xml-документа должна соответствовать продекларированной в заголовке. В случае отсутствия заголовка xml-документа, по умолчанию считается, что xml-документ представлен в кодировке utf-8. В случае десериализации документа с помощью TextReader или XmlReader заголовок не является обязательным и игнорируется как избыточная информация.
FromXml (методы)	Создать документ из XML-документа (XmlDocument) или его элемента (XmlElement)
FromXmlText (метод)	Создать документ из текста XML-документа
GetXmlText (метод)	Получить XML-текст документа
GetDataXmlText (метод)	Получить XML-текст электронных данных документа в заданной версии ЭПД *int epdVersion* – версия ЭПД в которой получить электронный данные документа (принимаемые значения 10, 11, 12, 13, 14, что соответствует версии ЭПД 1.0, 1.1, 1.2, 1.3, 1.4)
Load (метод)	Загрузить документ из файла. Xml-документ, содержащийся в файле должен начинаться с xml-заголовка с указанием кодировки xml-документа. Фактическая кодировка символов xml-документа должна соответствовать продекларированной в заголовке. В случае отсутствия заголовка xml-документа, по умолчанию считается, что xml-документ представлен в кодировке utf-8.
Save (метод)	Сохранить документ в файл в заданной кодировке
SaveData (метод)	Сохранить электронные данные документа в файл в заданной кодировке и в заданной версии ЭПД *int epdVersion* – версия ЭПД в которой получить электронный данные документа (принимаемые значения 10, 11, 12, 13, 14, что соответствует версии ЭПД 1.0, 1.1, 1.2, 1.3, 1.4)
Serialize (методы)	Сериализовать документ в массив байт, поток вывода, TextWriter или XmlWriter
SerializeData (методы)	Сериализовать электронные данные документа в массив байт, поток вывода, TextWriter или XmlWriter в заданной версии ЭПД *int epdVersion* – версия ЭПД в которой получить электронный данные документа (принимаемые значения 10, 11, 12, 13, 14, что соответствует версии ЭПД 1.0, 1.1, 1.2, 1.3,

	1.4)
ToXml (метод)	Преобразовать электронный перевозочный документ в XML-документ (XmlDocument)
DataToXml (метод)	Преобразовать электронные данные документа в XML-документ (XmlDocument) в заданной версии ЭПД *int epdVersion* – версия ЭПД в которой получить электронный данные документа (принимаемые значения 10, 11, 12, 13, 14, что соответствует версии ЭПД 1.0, 1.1, 1.2, 1.3, 1.4)
GetOTPR (метод)	Сохранение актуального текста перевозочного документа в формате xml в файл.
GetOTPRString (метод)	Получить актуальный текст перевозочного документа в кодировке utf-8.
Warning (свойство get;)	Предупреждение при отправке документа
SignerInfos (свойство get;)	Список информации о лице, что подписала документ электронно-цифровой подписью, которую можно получить по индексу. Если проверка электронно-цифровой подписи подтверждает достоверность данных, результатом будет null и в поле Error будет текст ошибки. Если документ не содержит электронно-цифровой подписи, генерируется исключение GohubException с кодом ошибки GohubErrCode.gohub_document_has_not_signature.

5.5. GohubAttachment

Класс GohubAttachment служит для представления информации о сопроводительных документах и содержит следующие члены:

Id (свойство)	Уникальный идентификатор сопроводительного документа
Description (свойство)	Текстовое описание сопроводительного документа, формирующееся из названия типа документа, его регистрационного номера и даты.
Name (свойство)	Название сопроводительного документа.
TypeCode (свойство)	Код типа сопроводительного документа для бланков ГУ и ЦИМ.
SmgsTypeCode (свойство)	Код типа сопроводительного документа для бланков СМГС и ЦИМ/СМГС согласно информационного руководства СМГС.
RegistrationNumber (свойство)	Регистрационный номер сопроводительного документа.
RegistrationDate (свойство)	Дата регистрации сопроводительного документа.
ValidFrom (свойство)	Дата начала действия сопроводительного документа.
ValidTo (свойство)	Дата окончания действия сопроводительного документа.
Load (метод)	Создает объект GohubAttachment по параметрам: *string typeCode* – код типа документа для бланков ГУ и ЦИМ; *string smgsTypeCode* – код типа документа для бланков СМГС и ЦИМ/СМГС; *string name* – название документа; *string registrationNumber* – регистрационный номер; *System.DateTime registrationDate* – дата регистрации; *System.DateTime validFrom* – дата, с которой сопроводительный

	документ действителен; *System.DateTime validTo* – дата, по которую сопроводительный документ действителен; *string path* – путь к отсканированному тексту документа в pdf формате.
`Load` `(метод)`	Создает объект `GohubAttachment` по параметрам: *string typeCode* – код типа документа для бланков ГУ и ЦИМ; *string smgsTypeCode* – код типа документа для бланков СМГС и ЦИМ/СМГС; *string name* – название документа; *string registrationNumber* – регистрационный номер; *System.DateTime registrationDate* – дата регистрации; *System.DateTime validFrom* – дата, с которой сопроводительный документ действителен; *System.DateTime validTo* – дата, по которую сопроводительный документ действителен; *string path* – путь к отсканированному тексту документа в pdf формате. *string pathUserData* – путь к электронным данным сопроводительного документа в xml формате.
`Save` `(метод)`	Сохраняет отсканированный текст сопроводительного документа в pdf формате. *string path* – путь к отсканированному тексту документа в pdf формате.
`SaveUserData` `(метод)`	Сохраняет пользовательские данные сопроводительного документа в xml формате. *string path* – путь к электронным данным сопроводительного документа в xml формате.

5.6. GohubEData

Класс `GohubEData` служит для представления информации об электронных данных ПИ и содержит следующие члены:

`Id` `(свойство)`	Уникальный идентификатор электронных данных ПИ
`Revision` `(свойство)`	Номер ревизии электронных данных ПИ.
`RevisionDate` `(свойство)`	Дата ревизии электронных данных ПИ.
`Version` `(свойство)`	Версия электронных данных ПИ.
`DocType` `(свойство)`	Код типа электронных данных ПИ: 190 счет-фактура, 320 упаковочный лист.
`Status` `(свойство)`	Код статуса электронных данных ПИ.
`AttachmentId` `(свойство)`	IDсопроводительного документа электронных данных ПИ.
`Data` `(свойство)`	Массив данных содержащийся в электронных данных ПИ.
`Load` `(метод)`	Создает объект `GohubEData` по параметрам: *ulong attachmentSmgsTypeCode* – код сопроводительного документа: 325 Счет-проформа (Инвойс), 380 (Инвойс) счет-фактура, 935 Счет-фактура; *string xmlPath* – путь к xml-файлу содержащий электронные данные ПИ;

	string name – название сопроводительного документа; *string registrationNumber* – регистрационный номер; *System.DateTime registrationDate* – дата регистрации; *System.DateTime validFrom* – дата, с которой сопроводительный документ действителен; *System.DateTime validTo* – дата, по которую сопроводительный документ действителен; *string pdfPath* – путь к отсканированному тексту сопроводительного документа в pdf формате.
`Load` (метод)	Создает объект `GohubEData` по параметрам: *ulong attachmentSmgsTypeCode* – код сопроводительного документа: 325 Счет-проформа (Инвойс), 380 (Инвойс) счет-фактура, 935 Счет-фактура; *string xmlPath* – путь к xml-файлу содержащий электронные данные
`Save` (метод)	Сохраняет текст электронных данных ПИ в xml формате. *string path* – путь к тексту электронных данных ПИ в xml формате.
`LoadData` (метод)	Загрузить данные из файла в имеющиеся электронные данные ПИ. *string path* – путь к тексту электронных данных ПИ в xml формате.

5.7. GohubPiPackage

Класс `GohubPiPackage` служит для представления информации пакете ПИ и содержит следующие члены:

`Id` (свойство)	Уникальный идентификатор пакета ПИ
`Revision` (свойство)	Номер ревизии пакета ПИ.
`RevisionDate` (свойство)	Дата ревизии пакета ПИ.
`ConsignmentId` (свойство)	ID перевозочного документа на который ссылается пакет ПИ.
`Status` (свойство)	Код статуса пакета ПИ.
`PiPackageToEDataList` (свойство)	Список `GohubPiPackageToEData` в пакете ПИ.
`Data` (свойство)	Массив данных содержащийся в полную информацию о пакете ПИ.
`Save` (метод)	Сохраняет текст данные пакета ПИ в xml формате. *string path* – путь к тексту данных пакета ПИ в xml формате.

5.8. GohubPiPackageToEData

Класс `GohubPiPackageToEData` служит для представления информации об электронных данных содержащиеся в пакете ПИ и содержит следующие члены:

`Id` (свойство)	ID объекта GohubPiPackageToEData
`EDataId` (свойство)	ID электронных данных в объекте GohubPiPackageToEData
`EDataVersion` (свойство)	Версия электронных данных в объекте GohubPiPackageToEData
`PiPackageId` (свойство)	ID пакета ПИ в объекте GohubPiPackageToEData
`Status` (свойство)	Статус объекта GohubPiPackageToEData

Note (свойство)	Примечание объекта GohubPiPackageToEData

5.9. GohubFdu92

Класс GohubFdu92 служит для представления информации о накопительных карточках ФДУ-92 и содержит следующие члены:

Id (свойство)	Уникальный идентификатор накопительной карточки ФДУ-92
Revision (свойство)	Номер ревизии накопительной карточки ФДУ-92.
Status (свойство)	Статус накопительной карточки ФДУ-92.
HasSignature (свойство)	Свойство проверяющее, подписана ли накопительная карточка ФДУ-92 электронно-цифровой подписью.
SignerInfo (свойство)	Информация о лице, подписавшем накопительную карточку ФДУ-92 электронно-цифровой подписью.
TimeStamp (свойство)	Метка времени, полученная при подписании накопительной карточки ФДУ-92 Электронно-цифровой подписью.
Load (метод)	Загрузить накопительную карточку ФДУ-92 из файла Xml.
Save (метод)	Сохраняет текст накопительной карточки ФДУ-92 в Xml формате.
GetXmlText (свойство)	Получить ФДУ-92 в виде строки
ToXml (свойство)	Преобразовать ФДУ-92 в Xml-документ

5.10. GohubGu46

Класс GohubGu46 служит для представления информации о ведомости пользования вагонами/контейнерами ГУ-46 и содержит следующие члены:

Id (свойство)	Уникальный идентификатор ведомости ГУ-46
Revision (свойство)	Номер ревизии ведомости ГУ-46
Status (свойство)	Статус ведомости ГУ-46
HasSignature (свойство)	Свойство проверяющее, подписана ли ведомость ГУ-46 электронно-цифровой подписью.
SignerInfo (свойство)	Информация о лице, подписавшем ведомость ГУ-46 электронно-цифровой подписью.
TimeStamp (свойство)	Метка времени, полученная при подписании ведомости ГУ-46 Электронно-цифровой подписью.
Load (метод)	Загрузить ведомость ГУ-46 из файла Xml
Save (метод)	Сохраняет текст ведомости ГУ-46 в Xml формате
GetXmlText (свойство)	Получить ГУ-46 в виде строки
ToXml (свойство)	Преобразовать ГУ-46 в Xml-документ

5.11. GohubGu45

Класс GohubGu45 служит для представления информации о памятке подаче/уборке вагонов и выдаче/приёме контейнеров ГУ-45 и содержит следующие члены:

Id (свойство)	Уникальный идентификатор памятки ГУ-45
Revision (свойство)	Номер ревизии памятки ГУ-45
Status (свойство)	Статус памятки ГУ-45
HasSignature (свойство)	Свойство проверяющее, подписана ли памятка ГУ-45электронно-цифровой подписью.
SignerInfo (свойство)	Информация о лице, подписавшем памятку ГУ-45 электронно-цифровой подписью.
TimeStamp (свойство)	Метка времени, полученная при подписании памятки ГУ-45 Электронно-цифровой подписью.
Save (метод)	Сохраняет текст памятки ГУ-45 в Xml формате
GetXmlText (свойство)	Памятка про подачу / уборку вагонов в виде строки
ToXml (свойство)	Преобразовать ГУ-45 в Xml-документ

5.12. GohubDocumentFilter

Класс GohubDocumentFilter служит для управления фильтрами для запросов документов с сервера и содержит следующие члены:

WagonNumber (свойство)	Фильтрация по номеру вагона
DocumentNumber (свойство)	Фильтрация по номеру документа
DocumentStatus (свойство)	Фильтрация по статусу документа
DepartureClientCode (свойство)	Фильтрация по коду отправителя
DeparturePayerCode (свойство)	Фильтрация по коду плательщика по отправлению
DepartureStationCode (свойство)	Фильтрация по коду станции отправления
ArrivalClientCode (свойство)	Фильтрация по коду получателя
ArrivalPayerCode (свойство)	Фильтрация по коду плательщика по прибытию
ArrivalStationCode (свойство)	Фильтрация по коду станции назначения
Clear (метод)	Очистить все фильтры.

5.13. GohubSigner

Класс GohubSigner служит для представления информации о владельцах электронных ключей и авторах электронно-цифровых подписей и содержит следующие члены:

Address (свойство)	Адрес
Department	Отдел

(свойство)	
Dns (свойство)	DNS
DrfoCode (свойство)	Код ДРФО
Edrpou_code (свойство)	Код ЕДРПОУ
Email (свойство)	Электронный адрес
Establishment (свойство)	Учреждение
FullName (свойство)	Полное имя (Фамилия, Имя, Отчество)
Issuer (свойство)	Организация, выдавшая сертификат подлинности
IssuerSummary (свойство)	Полная информация об организации, выдавшей сертификат подлинности
Locality (свойство)	Город
Name (свойство)	Имя
Phone (свойство)	Телефон
Region (свойство)	Область
SerialNumber (свойство)	Серийный номер сертификата
StaffPost (свойство)	Почтовый индекс
Summary (свойство)	Суммарная информация
ToString (свойство)	Получить строковое представление информации о владельце электронного ключа
Status (свойство)	Получить статус документа на момент наложения электронно-цифровой подписи при получении перечня подписей Значение статуса соответствует описанию статуса документа в ЭПД
Error (свойство)	Получить текст ошибки по индексу подписи при получении перечня подписей

5.14. GohubClient

Класс GohubClient служит для подключения файлов электронных ключей и содержит следующие члены:

MountFileKey (метод)	Монтировать файл электронного ключа, указав путь к директории и указав идентификатор для ключа
UnmountFileKey (метод)	Демонтировать файл электронного ключа, по его идентификатору
GetMountedKeys (метод)	Получить массив идентификаторов монтировать файлов электронных ключей
GetMountedKeyDir (метод)	Получить путь к директории в которой находится файл электронного ключа по его идентификатору

5.15. GohubException

Класс `GohubException` служит для информирования об ошибках времени выполнения с использованием механизма исключений платформы .NET Framework и содержит следующие члены:

`ErrCode` (свойство)	Код ошибки. Возможные значения см. в п.5.6 GohubErrCode

5.16. GohubErrCode

Перечисление `GohubErrCode` представляет набор кодов ошибок и содержит следующие члены:

`gohub_success = 0`	Операция выполнена успешно
`gohub_server_connection_could_not_be_established = 1`	Невозможно установить соединение с Сервером модуля согласования
`gohub_server_inaccessible = 2`	Сервер модуля согласования недоступен
`gohub_document_creation_failed = 3`	Невозможно создать документ
`gohub_document_query_failed = 4`	Запрос документа по идентификатору не удался
`gohub_next_document_query_failed = 5`	Запрос документа по ревизии не удался
`gohub_document_sending_failed = 6`	Отправка документа в АС «Клиент УЗ» не удалась
`gohub_document_saving_failed = 7`	Сохранение документа не удалось
`gohub_document_loading_failed = 8`	Загрузка документа не удалась
`gohub_private_key_could_not_be_opened = 10`	Не удалось открыть электронный ключ
`gohub_private_key_is_inaccessible = 11`	Электронный ключ недоступен
`gohub_document_could_not_be_signed = 12`	Не удалось подписать документ
`gohub_document_signature_verification_failed = 13`	Проверка электронно-цифровой подписи не подтвердила достоверность данных
`gohub_document_has_not_signature = 14`	Недопустимая операция. Документ не содержит электронно-цифровой подписи
`gohub_client_is_obsolete = 15`	Версия клиента устарела, обновите
`gohub_server_is_obsolete = 16`	Версия сервера устарела, обновите
`gohub_document_reclamation_failed = 17`	Отозвать документ не удалось
`gohub_document_deletion_failed = 18`	Удалить документ не удалось
`gohub_attachment_creation_failed = 19`	Создать сопроводительный документ не удалось
`gohub_attachment_sending_failed = 20`	Отправить сопроводительный документ не удалось
`gohub_attachment_query_failed = 21`	Запросить сопроводительный документ не удалось
`gohub_attachment_deletion_failed = 22`	Удалить сопроводительный документ не удалось
`gohub_attaching_to_document_failed = 23`	Прикрепить сопроводительный документ к перевозному документу не удалось
`gohub_detaching_from_document_failed = 24`	Открепить сопроводительный документ от перевозного документа не удалось
`gohub_mount_of_file_key_failed = 25`	Монтирование электронного ключа из файла не удалось.
`gohub_unmount_of_file_key_failed = 26`	Демонтирование электронного ключа из файла не удалось.

gohub_enumerating_of_file_keys_failed = 27	Получить список монтированных из файлов электронных ключей не удалось.
gohub_mounted_file_key_inaccessible = 28	Монтированный из файла электронный ключ не доступен.
gohub_edata_creation_failed = 29	Невозможно создать ЭД ПИ
gohub_edata_sending_failed = 30	Оправить ЭД ПИ не удалось
gohub_edata_updating_failed = 31	Обновить ЭД ПИ не удалось
gohub_edata_query_failed = 32	Запрос ЭД ПИ по идентификатору не удался
gohub_next_edata_query_failed = 33	Запрос ЭД ПИ по ревизии не удался
gohub_add_edata_to_pipackage_failed = 34	Добавить ЭД в пакет ПИ не удалось
gohub_pipackage_query_failed = 35	Запрос пакета ПИ по идентификатору не удался
gohub_next_pipackage_query_failed = 36	Запрос пакета ПИ по ревизии не удался
gohub_mp_months_query_failed = 37	Ошибка при запросе перечня идентификаторов месяцев из АС «Месплан»
gohub_orders_of_months_query_failed = 38	Ошибка при запросе заявок из АС «Месплан»
gohub_programm_error = 4096	Неизвестная ошибка в программе
gohub_invalid_operation = 4097	Клиентское приложение выполнило недопустимую операцию
gohub_inform_services_doc_saving_failed = 4098	Сохранение документа информационных услуг не удалось
gohub_inform_services_doc_query_failed = 4099	Запрос документа информационных услуг по идентификатору не удался
gohub_query_changes_inform_services_failed = 4100	Запрос идентификаторов документов информационных услуг с ревизиями выше заданной не удался
gohub_query_next_inform_services_document_failed = 4101	Запрос следующего согласно ревизии документа информационных услуг не удался
gohub_gu45_get_xml_text_failed	Получить ГУ-45 в виде строки не удалось
gohub_gu45_to_xml_failed	Получить ГУ-45 в виде XmlDocument не удалось
gohub_gu46_get_xml_text_failed	Получить ГУ-46 в виде строки не удалось
gohub_gu46_to_xml_failed	Получить ГУ-46 в виде XmlDocument не удалось
gohub_fdu92_get_xml_text_failed	Получить ФДУ-92 в виде строки не удалось
gohub_fdu92_to_xml_failed	Получить ФДУ-92 в виде XmlDocument не удалось

5.17. GohubInformServicesDoc

Класс GohubInformServicesDoc служит для представления документов информационных услуг и содержит следующие члены:

Id (свойство get;)	Уникальный идентификатор документа
Revision (свойство get;)	Номер ревизии документа
Comment (свойство get;)	Комментарий документа
CreatedDate	Дата создания документа

(свойство get;)	
DocDate (свойство get;)	Дата документа
FileBody (свойство get;)	Тело документа в виде массива байт
FileName (свойство get;)	Наименование документа
Save (метод)	Сохранить документ в файл
SaveXml (метод)	Сохранить документ в файл формате xml
IsEmpty (свойство get;)	Значение или пустой документ

5.18. GohubDispatchInfo

Класс GohubDispatchInfo служит для представления перечня информации о заказе на согласование перевозки по данным календаря планирования перевозок зерновых грузов (за последние 5 дней от текущей даты):

Date (поле)	Дата
Description (поле)	Описание
Number (поле)	Номер
Type (поле)	Тип
WagOwner (поле)	Владелец
IsEmpty (поле)	Состояние вагона Пустой или Загружен

5.19 GohubPSTDInfo

Класс GohubPSTDInfo служит для представления перечня информации о № заказе Согласования сокращения сроком доставки:

ArrivaDate (поле)	Дата прибытия
DepartureDate (поле)	Дата отправления
Number (поле)	Номер
ReqDate (поле)	Дата заказа

5.20 Примеры использования

а) Загрузка документа из файла и передача в АС «Клиент УЗ»

Проиллюстрируем загрузку документа из файла и передачу его в АС «Клиент УЗ». При этом используем электронный ключ для наложения электронно-цифровой подписи.

```
static string LoadAndSendDocument(
```

```
            GohubConnection connection, // соединение с сервером Модуля
      Согласования
            string path,            // путь к файлу документа
            string password)    //Пароль к ключу. Если не задан, документ не
будет подписан
        {
            try
            {
                // Загрузить документ из файла
                GohubDocument document = GohubDocument.Load(path);
                connection.OpenPrivateKey(password);
                Console.WriteLine("Private key info: {0}",
connection.SignerInfo);
                try
                {
                    connection.SignDocument(document);
                }
                catch
                {
                    throw;
                }
                finally
                {
                    connection.ClosePrivateKey();
                }

                connection.SendDocument(document);
                // Документ передан успешно

                Console.WriteLine("Document is sended: id={0} revision={1}",
document.Id, document.Revision);
                // Проверка предупреждений после отправки документа
                string warning = document.Warning;
                if (!string.IsNullOrEmpty(warning))
                    Console.WriteLine("Warning in document: {0}", warning);

                // Попробуем запросить этот же документ по идентификатору
                    document = connection.QueryDocument(document.Id);
                Console.WriteLine(document.GetXmlText());
                return document.Id;
            }
            catch (Exception e)
            {
                Console.WriteLine(e.ToString());
            }
            return null;
        }
```

б) Запрос документов из АС «Клиент УЗ»

В этом примере показана техника последовательного запроса документов их АС «Клиент УЗ» по номерам ревизий.

```
        static int QueryAndSaveDocuments(
            GohubConnection connection, // соединение с сервером Модуля
Согласования
            int startRevision,  //номер ревизии с которой получать перевозочные
документы
            int maxCount,       //Максимальное количество документов которое
будет запрашиваться
            string targetFolder)//папка куда сохранять полученные документы
        {
            int lastRevision = startRevision;
            int i = 0;
            var encoding = Encoding.GetEncoding(1251);
            foreach (var document in connection.QueryDocuments(lastRevision))
```

```
            {
                // Запрос документа выполнен успешно
                Console.WriteLine("Document received: id={0} revision={1}",
document.Id, document.Revision);
                // Вывести документ на экран
                Console.WriteLine(document.GetXmlText());
                try
                {
                    string path = string.Format("{0}\\{1}.xml", targetFolder,
document.Id);
                    document.Save(path, encoding);
                    Console.WriteLine("Document saved to file: path='{0}'",
path);
                }
                catch (Exception e)
                { // Ошибка при сохранении документа
                    Console.WriteLine(e.ToString());
                }

                // Проверка подписи
                if (document.HasSignature)
                {
                    Console.WriteLine("SignerInfo={0}\nTimeStamp={1}",
document.SignerInfo, document.TimeStamp);
                }

                // Запомнить ревизию последнего документа
                lastRevision = document.Revision;

                //увеличиваем счетчик полученных документов
                i++;
                if (i == maxCount)
                    break;
            }
            // Возврат вызывающей функции номера последней обработанной ревизии
            return lastRevision;
        }
```

в) Другие примеры использования

В инсталляционном пакете Клиента Модуля Согласования представлены также другие примеры использования, с которыми более детально можно ознакомиться, заглянув непосредственно в программный код (папка Клиента Модуля Согласовния\samples\ gohub.client.Test.cs). Здесь мы кратко опишем другие функции, представленные в примерах:

- Показана техника последовательного запроса документов ФДУ-92 из АС «Клиент УЗ» по номерам ревизий

```
        static ulong QueryAndSaveFdu92s(
            GohubConnection connection, // соединение с сервером Модуля
Согласования
            ulong startRevision,  //номер ревизии с которой получать перевозочные
документы
            int maxCount,       //Максимальное количество документов которое
будет запрашиваться
            string targetFolder)//папка куда сохранять полученные документы
```

- Показана техника последовательного запроса документов ЭД предварительного информирования из АС «Клиент УЗ» по номерам ревизий

```
        static ulong QueryAndSavePiPackages(
```

```
            GohubConnection connection, // соединение с сервером Модуля
Согласования
            ulong startRevision,  //номер ревизии с которой получать перевозочные
документы
            int maxCount,       //Максимальное количество документов которое
будет запрашиваться
            string targetFolder)//папка куда сохранять полученные документы
```

- Показана техника последовательного запроса пакетов предварительного информирования из АС «Клиент УЗ» по номерам ревизий

```
        static ulong QueryAndSavePiPackages(
            GohubConnection connection, // соединение с сервером Модуля
Согласования
            ulong startRevision,  //номер ревизии с которой получать перевозочные
документы
            int maxCount,       //Максимальное количество документов которое
будет запрашиваться
            string targetFolder)//папка куда сохранять полученные документы
```

- Показана техника добавления документа ЭД в пакет предварительного информирования

```
        static string AddEDataToPiPackage(
            GohubConnection connection, // соединение с сервером Модуля
Согласования
            string piPackageId, // ID пакета ПИ (предварительного информирования)
в который будут добавляться ЭД (электронные данные)
            string edataPath,   // путь к xml-файлу с ЭД
            uint edataType)     // код ЭД: 190 счет-фактура, 320 упаковочный лист
```

- Показана техника обновления документа ЭД содержащегося на сервере АС «Клиент УЗ»

```
        static ulong UpdateEData(
            GohubConnection connection, // соединение с сервером Модуля
Согласования
            string edataId,     // ID документа ЭД который будет меняться
            string edataPath)   // путь к xml-файлу с ЭД
```

- Показана техника Получение перечня информации о заказе на согласование перевозки по данным календаря планирования перевозок зерновых грузов (за последние 5 дней от текущей даты)

```
try
{
        Console.WriteLine("Please enter ESR code station from:");
        var from_esr = Console.ReadLine();
        Console.WriteLine("Please enter ESR code station to:");
        var to_esr = Console.ReadLine();
        var list = connection.QueryDispatchInfo(from_esr, to_esr);
        if (list != null && list.Count > 0)
        {
            Console.WriteLine("list loaded.\n");
            foreach (var item in list)
            {
                Console.Write(item.Description + "\n");
                Console.Write(item.Number + "\n");
                Console.Write(item.Date + "\n");
                Console.Write(item.Type + "\n");
                Console.Write(item.WagOwner + "\n");
                Console.Write(item.IsEmpty + "\n");
```

```
                }
            }
            else
                Console.WriteLine("list is empty");
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
```

- Показана техника получения списка с информацией про подпись в ПД

```
try
 {
     Console.WriteLine("Please enter document Id");
     string id = Console.ReadLine();

     Console.WriteLine("Test is starting...");
     GohubDocument doc = connection.QueryDocument(id);

     foreach (var item in doc.SignatureInfos)
     {
            if (item.Error == null)
Console.WriteLine(item.Summary + "\n" + item.Status.ToString());
            else
                  Console.WriteLine(item.Error);
      }
  }
  catch (Exception ex)
  {
      Console.WriteLine(ex.Message);
  }
```

- Показана техника получения списка и открытие электронного ключа по индексу

```
try
 {
            var list = connection.GetKeysList;
             if (list != null && list.Count > 0)
             {
                 Console.WriteLine("list loaded.\n");
                 foreach (var item in list)
                 {
                     Console.Write("NameMedia :" + item.name + "\t");
                     Console.Write("IndexMedia :" + item.index + "\t\n");
                     foreach (var i in item.devices)
                     {
                         Console.Write("\tNameDevice :" + i.name + "");
                         Console.Write("\tIndexDevice :" + i.index + "\n");
                     }
                 }
             }
             else
                 Console.WriteLine("list is empty");
             Console.WriteLine("Enter Media index");
             int mediaIndex = int.Parse(Console.ReadLine());
             Console.WriteLine("Enter Device index");
             int deviceIndex = int.Parse(Console.ReadLine());
             Console.WriteLine("Enter password");
             string password = Console.ReadLine();
```

```
            connection.OpenPrivateKey(password, mediaIndex, deviceIndex);
            Console.WriteLine(connection.SignerInfo);
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
```

6. COM/OLE библиотека

6.1.Состояния перевозочного документа

Коды перевозочного документа в *Библиотеке* представлены перечислением (*enum*) `GohubDocumentStatus`, содержащим набор констант.

Сводная таблица кодов состояний перевозочного документа

`gohub_document_status_unknown = 0`	Статус неизвестен
`gohub_document_status_draft = 1`	Черновик
`gohub_document_status_sending = 2`	Документ передается товарному кассиру
`gohub_document_status_registered = 3`	Документ передан товарному кассиру
`gohub_document_status_reclaiming = 4`	Документ отзывается от товарного кассира
`gohub_document_status_accepted = 5`	Груз принят к перевозке
`gohub_document_status_delivered = 6`	Груз прибыл
`gohub_document_status_recieved = 7`	Груз получен получателем
`gohub_document_status_uncredited = 8`	Документ раскредитован товарным кассиром
`gohub_document_status_recieved_draft = 9`	Груз получен получателем и редактируется
`gohub_document_status_recieved_sending = 10`	Груз получен получателем и передан товарному кассиру
`gohub_document_status_recieved_reclaiming = 11`	Груз получен получателем и отзывается от товарного кассира
`gohub_document_status_canceled = 12`	Документ испорчен товарным кассиром
`gohub_document_status_locked = 13`	Документ заблокирован

Более детальное описание программного интерфейса библиотеки можно получить из заголовочного файла gohub.client.errors.h (Приложение Б).

6.2.Идентификаторы интерфейсов

Обращаться к функциям библиотеки можно с использованием технологии COM/OLE. Имя COM-компонента (ProgID): *"TMSoft.GohubClient"*, Идентификатор (CLSID): `CF908D67-DAF6-43B0-9621-1DD417CFF3D7`. Компонент предоставляет следующие интерфейсы:

Имя	Идентификатор
IGohubClient	ABDA6C07-5320-4F28-B995-FADE037D0A82
IGohubDocument	0D5D6225-8E07-46E2-8B8D-9C0966481994
IGohubAttachment	044603D2-574E-463C-87EC-DCD98C30F319
IGohubEData	685B21FA-43A7-4ACC-9A57-AB7AC332942C
IGohubPiPackage	62C21013-07D6-4d9d-83C4-9FA6E770B2EA
IGohubPiPackageToEData	88AAFC89-0426-4551-BD1C-F57F63D0C335
IGohubConnection	E9967B9D-1141-47BA-A5C4-573FB02DB396
IGohubSignerInfo	B0E1F579-5836-4E97-9340-58B092418947
IGohubError	F1077417-21D9-4871-84A2-9F89525E7214
IGohubInformServicesDocument	0af49642-b12d-4fdb-b363-a7497cc78462
IGohubDispatchInfo	0669AB3B-8E1B-4161-8A2E-244D5A62029A
IGohubFdu92	EB03BE7D-48B7-4AFD-8A94-A5012D844A17
IGohubGu46	78DCD121-84B7-4D41-B15C-F9F7677A3519
IGohubGu45	C2F92D3C-295A-4453-B4CF-B33D2C8966E3
IGohubPSTDInfo	B8FFED27-656B-48C0-AD6F-8EBE09F8F8E8

Детальное описание приведено в приложении В

6.3.Интерфейс IGohubClient

Его идентификатор: ABDA6C07-5320-4F28-B995-FADE037D0A82

Детальное описание приведено в приложении В

Методы:

Название	Параметры	Возвращаемое значение
GetLastError		IgohubError
Connect	host как String, port как Long	IGohubConnection
CreateDocument	xmlText как String	IGohubDocument
LoadDocument	path как String	IGohubDocument
LoadAttachment	typeCode как String, name как String, regNumber как String, regDate как String, validFrom как String, validTo как String, path как String	IGohubAttachment
LoadAttachmentWithUserData	typeCode как String, name как String, regNumber как String, regDate как String, validFrom как String, validTo как String, path как String, pathUserData как String	IGohubAttachment
LoadSmgsAttachment	smgsTypeCode как String, name как String, regNumber как String, regDate как String, validFrom как String, validTo как String, path как String	IGohubAttachment
LoadSmgsAttachmentWithUserData	smgsTypeCode как String, name как String, regNumber как String, regDate как String, validFrom как String, validTo как String, path как String, pathUserData как String	IGohubAttachment
MountFileKey	keyId как String, keyDir как String	Boolean
UnmountFileKey	keyId как String	Boolean
QueryMountedKeys		Long
GetMountedKeyId	index как Long	String
GetMountedKeyDir	index как Long	String
LoadEData	codeType как UInt, xmlPath как String, name как String, regNumber как String, regDate как String, validFrom как String, validTo как String, pdfPath как String	IGohubEData
LoadEDataSimple	codeType как UInt, xmlPath как String	IGohubEData

LoadFdu92	ident как String, path как String	IGohubFdu92
LoadGu46	ident как String, path как String	IGohubGu46

6.4.Интерфейс IGohubDocument

Его идентификатор: 0D5D6225-8E07-46E2-8B8D-9C0966481994

Детальное описание приведено в приложении В

Свойства:

Название	Возвращаемое значение
Id	String
Text	String
Revision	Long
Status	Long
HasSignature	Boolean
Signer	IGohubSignerInfo
SignTime	String
AttachmentsCount	Long
MeasureEquipNum	String [get, put]
BusinessUnitNum	String [get, put]
ForeignNotAccept	Boolean
WarrantType	LONG[get, put]
OTPRString	String [get]
Warning	String [get]
SignerCount	LONG [get]

Методы:

Название	Параметры	Возвращаемое значение
Close		
Save	Path как String, codePage как Long	Boolean
SaveData	Path как String, codePage как Long, epdVersion как Long	Boolean
DataText	epdVersion как Long	String
Sign	connection как IGohubConnection	Boolean
Send	connection как IGohubConnection	Boolean
GetAttachmentIdByIndex	index как Long	String
SendReceived	connection как IgohubConnection docId как String	Boolean
set_VerifiedEmptyWeightForWagon, SetVerifiedEmptyWeightForWagon	wagonIndex как LONG weight как LONG	
get_VerifiedEmptyWeightForWagon, GetVerifiedEmptyWeightForWagon	wagonIndex как LONG	LONG
GetOTPR	Path как String	Boolean
SignerByIndex	index как Long	String
SignTimeByIndex	index как Long	String
SignStatusByIndex	index как Long	String
SignErrorByIndex	index как Long	String

6.5.Интерфейс IGohubAttachment

Его идентификатор: 044603D2-574E-463C-87EC-DCD98C30F319

Детальное описание приведено в приложении В

Свойства:

Название	Возвращаемое значение
Id	String
TypeCode	String
SmgsTypeCode	String
Name	String
Description	String
RegNumber	String
RegDate	String
ValidFrom	String
ValidTo	String

Методы:

Название	Параметры	Возвращаемое значение
Close		
Save	Path как String	Boolean
Send	connection как IGohubConnection	Boolean
SaveUserData	Path как String	Boolean

6.6.Интерфейс IGohubEData

Его идентификатор: 685B21FA-43A7-4ACC-9A57-AB7AC332942C

Детальное описание приведено в приложении В

Свойства:

Название	Возвращаемое значение
Id	String
Revision	UInt64
RevisionDate	String
Version	String
DocType	UInt
Status	Int
AttachmentId	String

Методы:

Название	Параметры	Возвращаемое значение
Close		
Save	Path как String	Boolean
LoadData	Path как String	Boolean
Send	connection как IGohubConnection	Boolean
Update	connection как IGohubConnection	Boolean

6.7.Интерфейс IGohubPiPackage

Его идентификатор: 62C21013-07D6-4d9d-83C4-9FA6E770B2EA

Детальное описание приведено в приложении В

Свойства:

Название	Возвращаемое значение
Id	String
Revision	UInt64
RevisionDate	String
ConsignmentId	String
Status	Int
PiPackageToEDataCount	Int

Методы:

Название	Параметры	Возвращаемое значение
Close		
Save	Path как String	Boolean
PiPackageToEData	Index как Int	IGohubPiPackageToEData

6.8.Интерфейс IGohubPiPackageToEData

Его идентификатор: 88AAFC89-0426-4551-BD1C-F57F63D0C335

Детальное описание приведено в приложении В

Свойства:

Название	Возвращаемое значение
Id	String
EDataId	String
PiPackageId	String
Note	String
Status	Int
EDataVersion	String

6.9.Интерфейс IGohubConnection

Его идентификатор: E9967B9D-1141-47BA-A5C4-573FB02DB396

Детальное описание приведено в приложении В

Свойства:

Название	Возвращаемое значение
Signer	IGohubSignerInfo

Методы:

Название	Параметры	Возвращаемое значение
Close		
QueryDocument	docId как String	IGohubDocument

QueryNextDocument	revision как Long	IGohubDocument
QueryNextDocument2	revision как Long	IGohubDocument
OpenPrivateKey	password как String	Boolean
OpenPrivateKeyFromPath	password как String, path как String	Boolean
ReclaimDocument	docId как String	Boolean
DeleteDocument	docId как String	Boolean
QueryAttachment	attachmentId как String	IgohubAttachment
QueryAttachmentWithUserData	attachmentId как String	IgohubAttachment
DeleteAttachment	attachmentId как String	Boolean
FilterByDocumentStatus	newVal как Long	Свойства – возвращает либо устанавливает параметр (в зависимости от примения)
FilterByDocumentNumber	newVal как String	
FilterByWagonNumber	newVal как String	
FilterByDepartureClientCode	newVal как String	
FilterByDeparturePayerCode	newVal как String	
FilterByDepartureStationCode	newVal как String	
FilterByArrivalClientCode	newVal как String	
FilterByArrivalPayerCode	newVal как String	
FilterByArrivalStationCode	newVal как String	
ClearAllFilters		Boolean
QueryAndSaveDocumentPrintableForm, SaveDocumentPrintableForm	docId как String, path как String	Boolean
QueryEData	eDataId как String	IGohubEData
QueryNextEData	revision как Long	IGohubEData
QueryPiPackage	piPackageId как String	IGohubPiPackage
QueryNextPiPackage	revision как Long	IGohubPiPackage
AddEDataToPiPackage	eData как IGohubEData piPackageId как String	IGohubPiPackage
GetMPMonths		String
QueryAndSaveOrdersForMonth	month как String, path как String	Boolean
QueryAndSaveOrdersForMonthWithRelogin, SaveOrdersForMonthWithRelogin	month как String, login как String, password как String,	Boolean

	path как String	
QueryEDataForAttachment	attachmentId как String	IGohubEData
QueryInfServsDoc	docId как Long	IGohubInformServicesDocument
QueryNextInfServsDoc	revision как Long	IGohubInformServicesDocument
QueryFdu92ByNumber	registration_esr как String, registration_num как String	IGohubFdu92
QueryGu45ByNumber	registration_esr как String, registration_num как String, registration_date как String	IGohubGu45
QueryGu46ByNumber	registration_esr как String, registration_num как String	IGohubGu46
QueryAndSaveFdu92PrintableForm	docId як String, path як String	Boolean
QueryAndSaveGu45PrintableForm	docId як String, path як String	Boolean
QueryAndSaveGu46PrintableForm	docId як String, path як String	Boolean

6.10. Интерфейс IGohubSignerInfo

Его идентификатор: B0E1F579-5836-4E97-9340-58B092418947

Детальное описание приведено в приложении В

Свойства:

Название	Возвращаемое значение
Name	String
Subject	String

6.11. Интерфейс IGohubError

Его идентификатор: F1077417-21D9-4871-84A2-9F89525E7214

Детальное описание приведено в приложении В

Свойства:

Название	Возвращаемое значение
Code	Long
Title	String
Text	String

6.12. Интерфейс IGohubFdu92

Его идентификатор: EB03BE7D-48B7-4AFD-8A94-A5012D844A17

Детальное описание приведено в приложении В

Свойства:

Название	Возвращаемое значение
Id	String
Revision	Long
Status	Long
Text	String
HasSignature	Boolean
Signer	IGohubSignerInfo
SignTime	String

Методы:

Название	Параметры	Возвращаемое значение
Close		
Save	Path как String	Boolean
Send	connection как IGohubConnection	Boolean
Sign	connection как IGohubConnection	Boolean

6.13. Интерфейс IGohubGu46

Его идентификатор: 78DCD121-84B7-4D41-B15C-F9F7677A3519

Детальное описание приведено в приложении В

Свойства:

Название	Возвращаемое значение
Id	String
Revision	Long
Status	Long
Text	String
HasSignature	Boolean
Signer	IGohubSignerInfo
SignTime	String

Методы:

Название	Параметры	Возвращаемое значение
Close		
Save	Path как String	Boolean

Send	connection как IGohubConnection	Boolean
Sign	connection как IGohubConnection	Boolean

6.14. Интерфейс IGohubGu45

Его идентификатор: 78DCD121-84B7-4D41-B15C-F9F7677A3519

Детальное описание приведено в приложении В

Свойства:

Название	Возвращаемое значение
Id	String
Revision	Long
Status	Long
Text	String
HasSignature	Boolean
Signer	IGohubSignerInfo
SignTime	String

Методы:

Название	Параметры	Возвращаемое значение
Close		
Save	Path как String	Boolean

6.15. Интерфейс IGohubInformServicesDocument

Его идентификатор: 0af49642-b12d-4fdb-b363-a7497cc78462

Детальное описание приведено в приложении В

Свойства:

Название	Возвращаемое значение
Id	UInt64
Revision	UInt64
FileName	String
Comment	String
CreatedDate	String
DocDate	String

Методы:

Название	Параметры	Возвращаемое значение
Close		
Save	Path как String	Boolean
SaveXml	Path как String	Boolean

6.16. Интерфейс IGohubDispatchInfo

Его идентификатор: 0669AB3B-8E1B-4161-8A2E-244D5A62029A

Детальное описание приведено в приложении В

Свойства:

Название	Параметры	Возвращаемое значение
Close		
WagOwnerByIndex	index як Long	String
TypeByIndex	index як Long	String
DateByIndex	index як Long	String
DescriptionByIndex	index як Long	String
NumberByIndex	index як Long	String
IsEmptyByIndex	index як Long	String
Count		UInt64

6.17 Интерфейс IPSTDInfo

Его идентификатор: B8FFED27-656B-48C0-AD6F-8EBE09F8F8E8

Детальное описание приведено в приложении В

Свойства:

Название	Параметры	Возвращаемое значение
Close		
ArrivelDateByIndex	index як Long	String
DepartureByIndex	index як Long	String
NumberByIndex	index як Long	String
ReqDateByIndex	index як Long	String
Count		UInt64

6.18 Примеры использования

Все приводимые примеры использования Модуля Согласования для COM/OLE библиотеки приведены на языки Visual Basic.

а) Вывод на экран информации об ошибке

Приведем пример получения информации об ошибке. Определим функцию PrintError, которая выводит на экран подробную информацию об ошибке, включая информацию о вложенных ошибках, если таковые имеются. Эта функция пригодится нам в следующих примерах.

```
Sub PrintError(ByVal client As Object)
    Dim Err As Object = client.GetLastError()
    Console.WriteLine(Err.Code)
    Console.WriteLine(Err.Title)
    Console.WriteLine(Err.Text)
    Marshal.ReleaseComObject(Err)
End Sub
```

б) Загрузка документа из файла и передача в АС «Клиент УЗ»

Проиллюстрируем загрузку документа из файла и передачу его в АС «Клиент УЗ». При этом используем электронный ключ для наложения электронно-цифровой подписи. Для вывода информации о возможных ошибках используем функцию PrintError, определенную ранее в п.6.15.а.

```
    Sub LoadAndSendDocument(ByRef client As Object, ByRef connection As Object, _
                            ByVal path As String, _
                            ByVal password As String)
        ' Загрузить документ из файла
        Dim document As Object = client.LoadDocument(path)
        If (document Is Nothing) Then
            ' Ошибка при загрузке документа из файла
            PrintError(client)
            Return
        End If
        ' Документ загружен успешно
        Console.WriteLine("Document is loaded")

        ' Подписать документ
        If Not connection.OpenPrivateKey(password) Then
            ' Не удалось получить доступ к закрытому ключу
            PrintError(client)
            CloseAndRelease(document)
            Return
        End If
        Console.WriteLine("Signing document . . .")
        Console.WriteLine("Private key info: name={0}, Subject={1}",
connection.Signer.Name, connection.Signer.Subject)

        If Not document.Sign(connection) Then
            ' Не удалось получить доступ к закрытому ключу
            PrintError(client)
            CloseAndRelease(document)
            Return
        End If

        ' Передать документ в АС "Клиент УЗ" через Модуль Согласования
        If Not document.Send(connection) Then
            ' Ошибка при передаче документа
            PrintError(client)
            CloseAndRelease(document)
            Return
        End If

        ' Документ передан успешно
        Console.WriteLine("Document is sended: id={0} revision={1}", document.Id,
document.Revision)

        'Проверка и вывод предупреждений после отправки документа
        Dim warning As String
        warning = documnet.Warning
        If Not String.IsNullOrEmpty(warning) Then
            Console.WriteLine("Warning: {0}", warning)
        End If

        ' Сохраняем идентификатор документа для дальнейшего использования
        Dim id As String = document.Id
        CloseAndRelease(document)

        ' Попробуем запросить этот же документ по идентификатору
        document = connection.QueryDocument(id)
        If (document Is Nothing) Then
            ' Ошибка при получении документа
            PrintError(client)
            Return
        End If

        Console.WriteLine("Document is queried.")
        Console.WriteLine(document.Text)
```

```
        CloseAndRelease(document)
    End Sub
```

в) Запрос документов из АС «Клиент УЗ»

В этом примере показана техника последовательного запроса документов их АС «Клиент УЗ» по номерам ревизий. Кроме того показано изменение кодовой страницы документа и сохранение запрошенных документов в файлы. Кроме того, выполняется проверка электронно-цифровой подписи. Для вывода информации об ошибках по-прежнему используется функция `PrintError`, определенную ранее в п.6.15.а

```
    Function QueryAndSaveDocuments(ByRef client As Object, ByRef connection As
Object, _
                                   ByVal startRevision As Integer, _
                                   ByVal maxCount As Integer, _
                                   ByVal targetFolder As String _
                                   ) As Integer
        Dim lastRevision As Integer = startRevision
        Dim document As Object
        For count As Integer = 0 To maxCount
            ' Запрос следующего документа по ревизии
            document = connection.QueryNextDocument(lastRevision)
            If (document Is Nothing) Then
                PrintError(client)
                Exit For
            End If
            ' Запрос документа выполнен успешно
            ' Вывести документ на экран
            Console.WriteLine("Document queried: Id={0}, Rev={1}", document.Id,
document.Revision)

            Dim path As String = targetFolder + "\" + document.Id + ".xml"
            If (document.Save(path, 866)) Then
                Console.WriteLine("Document saved to file: path={0}", path)
            Else
                'Ошибка при сохранении документа
                PrintError(client)
            End If

            ' Проверка подписи
            If (document.HasSignature) Then
                Console.WriteLine("Signer: Name={0}, Subject={1}. TimeStamp={2}",
document.Signer.Name, document.Signer.Subject, document.SignTime)
            End If
            ' Запомнить ревизию последнего документа
            lastRevision = document.Revision
            CloseAndRelease(document)
        Next
        Return lastRevision
    End Function
```

г) Другие примеры использования

В инсталляционном пакете Клиента Модуля Согласования представлены также другие примеры использования, с которыми более детально можно ознакомиться, заглянув непосредственно в программный код (папка Клиента Модуля Согласовния\samples\ Gohub.Client.COM.Test.VB). Здесь мы кратко опишем другие функции, представленные в примерах:

- Показана техника последовательного запроса документов ФДУ-92 из АС «Клиент УЗ» по номерам ревизий

```
    Function QueryAndSaveFdu92s(ByRef client As Object, ByRef connection As
Object, _
                                ByVal startRevision As Integer, _
                                ByVal maxCount As Integer, _
                                ByVal targetFolder As String _
                                ) As Integer
```

- Показана техника последовательного запроса документов ЭД предварительного информирования из АС «Клиент УЗ» по номерам ревизий

```
    Function QueryAndSaveEDatas(ByRef client As Object, ByRef connection As
Object, _
                                ByVal startRevision As ULong, _
                                ByVal maxCount As Integer, _
                                ByVal targetFolder As String _
                                ) As ULong
```

- Показана техника последовательного запроса пакетов предварительного информирования из АС «Клиент УЗ» по номерам ревизий

```
    Function QueryAndSavePiPackages(ByRef client As Object, ByRef connection As
Object, _
                                    ByVal startRevision As ULong, _
                                    ByVal maxCount As Integer, _
                                    ByVal targetFolder As String _
                                    ) As ULong
```

- Показана техника добавления документа ЭД в пакет предварительного информирования

```
    Function AddEDataToPiPackage(ByRef client As Object, ByRef connection As
Object, _
                                 ByVal piPackageId As String, _
                                 ByVal edataPath As String, _
                                 ByVal edataType As Integer _
                                 ) As String
```

- Показана техника обновления документа ЭД содержащегося на сервере АС «Клиент УЗ»

```
    Function UpdateEData(ByRef client As Object, ByRef connection As Object, _
                         ByVal edataId As String, _
                         ByVal edataPath As String _
                         ) As ULong
```

- Показана техника Получение перечня информации о заказе на согласование перевозки по данным календаря планирования перевозок зерновых грузов (за последние 5 дней от текущей даты)

```
        Try
              Console.Write(" Please enter ESR number of start station: ")
              Dim start_esr = Console.ReadLine()
              Console.Write(" Please enter ESR number of end station: ")
              Dim end_ser = Console.ReadLine()
              Dim doc = connection.QueryDispatchInfo(start_esr, end_ser)
              PrintError(client.GetLastError())
              If doc IsNot Nothing Then
                  Dim count = doc.Count()
                  If count > 0 Then
```

```
                Dim i = 0
                While i < count
                    Console.WriteLine(doc.DescriptionByIndex(i))
                    Console.WriteLine(doc.NumberByIndex(i))
                    Console.WriteLine(doc.DateByIndex(i))
                    Console.WriteLine(doc.TypeByIndex(i))
                    Console.WriteLine(doc.WagOwnerByIndex(i))
                    Console.WriteLine(doc.IsEmptyByIndex(i))
                    i += 1
                End While
            End If
            doc.Close()
        End If
    Catch ex As Exception
        Console.WriteLine(ex.Message)
        Console.ReadKey()
```

- Показана техника получения списка информации про подписи в ПД

```
Try
Dim id As Object
Dim sign As Object
Dim count As Integer
Dim time As Object
Dim status As Integer
Dim err As Object
Dim id = Console.ReadLine()
Dim doc = connection.QueryDocument(id)
PrintError(client.GetLastError())
If (doc Is Nothing) Then
    Exit Sub
End If
id = doc.Id
count = doc.SignerCount
sign = doc.SignerByIndex(0)
time = doc.SignTimeByIndex(0)
status = doc.SignStatusByIndex(0)
err = doc.SignErrorByIndex(0)
If (sign Is Nothing) Then
      Console.WriteLine(err.ToString())
Exit Sub
End If
Console.WriteLine("Document queried: Id={0} SignName={1}  Count={2}
SignTime={3}  SignStatus{4}", id, sign.Name, count, time, status)
doc.Close()
Catch ex As Exception
      Console.WriteLine(ex.Message)
      Console.ReadKey()
```

- Показана техника получения списка носителей и ключей, открытие электронного ключа по индексу

```
Try
      Dim keysList = connection.get_KeysList()
      PrintError(client.GetLastError())
      If keysList IsNot Nothing Then
          Console.WriteLine(keysList)
      End If
      Dim mediaIndex As Integer
      Dim deviceIndex As Integer
      Dim password As String
      Console.WriteLine("Enter Media index")
```

```
            mediaIndex = Console.ReadLine()
            Console.WriteLine("Enter device index")
            deviceIndex = Console.ReadLine()
            Console.WriteLine("Enter password")
            password = Console.ReadLine()
            connection.OpenPrivateKeyByIndex(password, mediaIndex, deviceIndex)
            Console.WriteLine("Private key info: name={0}, Subject={1}",
connection.Signer.Name, connection.Signer.Subject)
        Catch ex As Exception
            Console.WriteLine(ex.Message)
            Console.ReadKey()
        End Try
```

ПРИЛОЖЕНИЕ А. Заголовочный файл ***gohub.client.h***

```
#ifndef GOHUB_CLIENT_H_
#define GOHUB_CLIENT_H_

#include "gohub.client.errors.h"

#ifdef __cplusplus
extern "C" {
#endif

////////////////////
// Основные типы //
////////////////////
typedef int GohubBool;
typedef unsigned short GohubWChar;
typedef struct GohubConnection GohubConnection;
typedef struct GohubDocument GohubDocument;
typedef struct GohubAttachment GohubAttachment;
typedef struct GohubEData GohubEData;
typedef struct GohubPiPackageToEData GohubPiPackageToEData;
typedef struct GohubPiPackage GohubPiPackage;
typedef struct GohubFdu92 GohubFdu92;
typedef struct GohubGu46 GohubGu46;
typedef struct GohubGu45 GohubGu45;
typedef struct GohubGu27 GohubGu27;
typedef struct GohubInformServicesDoc GohubInformServicesDoc;
typedef struct GohubDispatchInfo GohubDispatchInfo;
typedef struct GohubPSTDInfo GohubPSTDInfo;
////////////////////////////////////
// Статусы перевозочного документа //
////////////////////////////////////
typedef enum GohubDocumentStatus
{
  gohub_document_status_unknown,
  gohub_document_status_draft,
  gohub_document_status_sending,
  gohub_document_status_registered,
  gohub_document_status_reclaiming,
  gohub_document_status_accepted,
  gohub_document_status_delivered,
  gohub_document_status_recieved,
  gohub_document_status_uncredited,
  gohub_document_status_recieved_draft,
  gohub_document_status_recieved_sending,
  gohub_document_status_recieved_reclaiming,
  gohub_document_status_canceled,
  gohub_document_status_locked,
} GohubDocumentStatus;
////////////////////////////////////
// Статусы перевозочного документа //
////////////////////////////////////
typedef enum GohubDocumentStatus
{
  gohub_document_status_unknown,
  gohub_document_status_draft,
  gohub_document_status_sending,
  gohub_document_status_registered,
  gohub_document_status_reclaiming,
  gohub_document_status_accepted,
  gohub_document_status_delivered,
  gohub_document_status_recieved,
  gohub_document_status_uncredited,
  gohub_document_status_recieved_draft,
  gohub_document_status_recieved_sending,
```

```
  gohub_document_status_recieved_reclaiming,
  gohub_document_status_canceled,
  gohub_document_status_locked,
} GohubDocumentStatus;

////////////////////
// Статусы ГУ-27 //
////////////////////
typedef enum GohubGu27Status
{
  gohub_gu27_status_unknown,
  gohub_gu27_status_draft,
  gohub_gu27_status_sending,
  gohub_gu27_status_registered,
  gohub_gu27_status_reclaiming,
  gohub_gu27_status_accepted,
  gohub_gu27_status_delivered,
  gohub_gu27_status_recieved,
  gohub_gu27_status_uncredited,
  gohub_gu27_status_recieved_draft,
  gohub_gu27_status_recieved_sending,
  gohub_gu27_status_recieved_reclaiming,
  gohub_gu27_status_canceled,
  gohub_gu27_status_locked,
} GohubGu27Status;

///////////////////////////////////
// Работа с кодовыми страницами //
///////////////////////////////////
int gohub_codepage();
int gohub_set_codepage(int codepage);
int gohub_encoding_codepage(const char* encodingName);
int gohub_encoding_codepage_w(const GohubWChar* encodingName);

/////////////////////////////////////////
// Подключение к Модулю Согласования //
/////////////////////////////////////////
GohubConnection* gohub_connect(const char* host, int port);
GohubConnection* gohub_connect_w(const GohubWChar* host, int port);
GohubBool gohub_disconnect(GohubConnection* connection);

//////////////////////////
// Работа с документами //
//////////////////////////
GohubDocument* gohub_load_document(const char* path);
GohubDocument* gohub_load_document_w(const GohubWChar* path);
GohubDocument* gohub_create_document(const char* content);
GohubDocument* gohub_create_document_w(const GohubWChar* content);
GohubDocument* gohub_query_document(GohubConnection* connection, const char*
documentId);
GohubDocument* gohub_query_document_w(GohubConnection* connection, const
GohubWChar* documentId);
GohubDocument* gohub_query_next_document(GohubConnection* connection, int
lastRevision);
GohubDocument* gohub_query_next_document2(GohubConnection* connection, int
lastRevision);
const char* gohub_document_id(GohubDocument* document);
const GohubWChar* gohub_document_id_w(GohubDocument* document);
int gohub_document_revision(GohubDocument* document);
const char* gohub_document_text(GohubDocument* document);
const GohubWChar* gohub_document_text_w(GohubDocument* document);
const char* gohub_document_data_text(GohubDocument* document, int epdVersion);
const GohubWChar* gohub_document_data_text_w(GohubDocument* document, int
epdVersion);
int gohub_document_size(GohubDocument* document);
```

```
const char* gohub_document_measure_equip_num(GohubDocument* document);
const GohubWChar* gohub_document_measure_equip_num_w(GohubDocument* document);
const char* gohub_document_business_unit_num(GohubDocument* document);
const GohubWChar* gohub_document_business_unit_num_w(GohubDocument* document);
int gohub_document_get_verified_empty_weight_for_wagon(GohubDocument* document,
int wagonIndex);
bool gohub_document_get_foreign_not_accept(GohubDocument* document);
int gohub_document_warrant_type(GohubDocument* document);
GohubBool gohub_document_set_measure_equip_num(GohubDocument* document, const
char* val);
GohubBool gohub_document_set_measure_equip_num_w(GohubDocument* document, const
GohubWChar* val);
GohubBool gohub_document_set_business_unit_num(GohubDocument* document, const
char* val);
GohubBool gohub_document_set_business_unit_num_w(GohubDocument* document, const
GohubWChar* val);
GohubBool gohub_document_set_warrant_type(GohubDocument* document, int val);
GohubBool gohub_document_set_verified_empty_weight_for_wagon(GohubDocument*
document, int wagonIndex, int val);
GohubBool gohub_send_document(GohubConnection* connection, GohubDocument*
document);
GohubBool gohub_send_received_document(GohubConnection* connection,
GohubDocument* document, const char* documentId);
GohubBool gohub_send_received_document_w(GohubConnection* connection,
GohubDocument* document, const GohubWChar* documentId);
GohubBool gohub_save_document(GohubDocument* document, const char* path, int
codePage);
GohubBool gohub_save_document_w(GohubDocument* document, const GohubWChar* path,
int codePage);
GohubBool gohub_save_document_data(GohubDocument* document, const char* path, int
codePage, int epdVersion);
GohubBool gohub_save_document_data_w(GohubDocument* document, const GohubWChar*
path, int codePage, int epdVersion);
GohubBool gohub_close_document(GohubDocument* document);
GohubBool gohub_reclaim_document(GohubConnection* connection, const char*
documentId);
GohubBool gohub_reclaim_document_w(GohubConnection* connection, const GohubWChar*
documentId);
GohubBool gohub_delete_document(GohubConnection* connection, const char*
documentId);
GohubBool gohub_delete_document_w(GohubConnection* connection, const GohubWChar*
documentId);
GohubDocumentStatus gohub_document_status(GohubDocument* document);
GohubBool gohub_document_get_otpr(GohubDocument* document, const char* path);
GohubBool gohub_document_get_otpr_w(GohubDocument* document, const GohubWChar*
path);
const char* gohub_document_get_otpr_string(GohubDocument* document);
const GohubWChar* gohub_document_get_otpr_string_w(GohubDocument* document);
const char* gohub_document_warning(GohubDocument* document);
const GohubWChar* gohub_document_warning_w(GohubDocument* document);


//////////////////////////////////////////
// Работа с фильтрами запроса документов //
//////////////////////////////////////////
// Наложение фильтров влияет на результат работы функции
gohub_query_next_document
GohubBool gohub_clear_all_filters(GohubConnection* connection);
GohubBool gohub_set_filter_by_document_status(GohubConnection* connection, int
documentStatusCode);
GohubBool gohub_set_filter_by_document_number(GohubConnection* connection, const
char* documentNumber);
GohubBool gohub_set_filter_by_document_number_w(GohubConnection* connection,
const GohubWChar* documentNumber);
```

```
GohubBool gohub_set_filter_by_wagon_number(GohubConnection* connection, const
char* wagonNumber);
GohubBool gohub_set_filter_by_wagon_number_w(GohubConnection* connection, const
GohubWChar* wagonNumber);
GohubBool gohub_set_filter_by_departure_client(GohubConnection* connection, const
char* clientCode);
GohubBool gohub_set_filter_by_departure_client_w(GohubConnection* connection,
const GohubWChar* clientCode);
GohubBool gohub_set_filter_by_departure_payer(GohubConnection* connection, const
char* payerCode);
GohubBool gohub_set_filter_by_departure_payer_w(GohubConnection* connection,
const GohubWChar* payerCode);
GohubBool gohub_set_filter_by_departure_station(GohubConnection* connection,
const char* stationCode);
GohubBool gohub_set_filter_by_departure_station_w(GohubConnection* connection,
const GohubWChar* stationCode);
GohubBool gohub_set_filter_by_arrival_client(GohubConnection* connection, const
char* clientCode);
GohubBool gohub_set_filter_by_arrival_client_w(GohubConnection* connection, const
GohubWChar* clientCode);
GohubBool gohub_set_filter_by_arrival_payer(GohubConnection* connection, const
char* payerCode);
GohubBool gohub_set_filter_by_arrival_payer_w(GohubConnection* connection, const
GohubWChar* payerCode);
GohubBool gohub_set_filter_by_arrival_station(GohubConnection* connection, const
char* stationCode);
GohubBool gohub_set_filter_by_arrival_station_w(GohubConnection* connection,
const GohubWChar* stationCode);

GohubDocumentStatus gohub_get_filter_by_document_status(GohubConnection*
connection);
const char* gohub_get_filter_by_document_number(GohubConnection* connection);
const GohubWChar* gohub_get_filter_by_document_number_w(GohubConnection*
connection);
const char* gohub_get_filter_by_wagon_number(GohubConnection* connection);
const GohubWChar* gohub_get_filter_by_wagon_number_w(GohubConnection*
connection);
const char* gohub_get_filter_by_departure_client(GohubConnection* connection);
const GohubWChar* gohub_get_filter_by_departure_client_w(GohubConnection*
connection);
const char* gohub_get_filter_by_departure_payer(GohubConnection* connection);
const GohubWChar* gohub_get_filter_by_departure_payer_w(GohubConnection*
connection);
const char* gohub_get_filter_by_departure_station(GohubConnection* connection);
const GohubWChar* gohub_get_filter_by_departure_station_w(GohubConnection*
connection);
const char* gohub_get_filter_by_arrival_client(GohubConnection* connection);
const GohubWChar* gohub_get_filter_by_arrival_client_w(GohubConnection*
connection);
const char* gohub_get_filter_by_arrival_payer(GohubConnection* connection);
const GohubWChar* gohub_get_filter_by_arrival_payer_w(GohubConnection*
connection);
const char* gohub_get_filter_by_arrival_station(GohubConnection* connection);
const GohubWChar* gohub_get_filter_by_arrival_station_w(GohubConnection*
connection);

////////////////////////////////////
// Запрос печатных форм документов //
////////////////////////////////////
GohubBool gohub_query_and_save_document_printable_form(
  GohubConnection* connection,
  const char* documentId,
  const char* path);

GohubBool gohub_query_and_save_document_printable_form_w(
```

```
  GohubConnection* connection,
  const GohubWChar* documentId,
  const GohubWChar* path);

GohubBool gohub_query_and_save_fdu92_printable_form(
  GohubConnection* connection,
  const char* fdu92Id,
  const char* path);

GohubBool gohub_query_and_save_fdu92_printable_form_w(
  GohubConnection* connection,
  const GohubWChar* fdu92Id,
  const GohubWChar* path);

GohubBool gohub_query_and_save_gu46_printable_form(
  GohubConnection* connection,
  const char* gu46Id,
  const char* path);

GohubBool gohub_query_and_save_gu46_printable_form_w(
  GohubConnection* connection,
  const GohubWChar* gu46Id,
  const GohubWChar* path);

GohubBool gohub_query_and_save_gu45_printable_form(
  GohubConnection* connection,
  const char* gu45Id,
  const char* path);

GohubBool gohub_query_and_save_gu45_printable_form_w(
  GohubConnection* connection,
  const GohubWChar* gu45Id,
  const GohubWChar* path);

GohubBool gohub_query_and_save_gu27_printable_form(
  GohubConnection* connection,
  const char* gu27Id,
  const char* path);

GohubBool gohub_query_and_save_gu27_printable_form_w(
  GohubConnection* connection,
  const GohubWChar* gu27Id,
  const GohubWChar* path);

//////////////////////////////////////////
// Работа с сопроводительными документами //
//////////////////////////////////////////
GohubAttachment* gohub_load_attachment(
  const char* typeCode,
  const char* name,
  const char* regNumber,
  const char* regDate,
  const char* validFrom,
  const char* validTo,
  const char* path);
GohubAttachment* gohub_load_attachment_w(
  const GohubWChar* typeCode,
  const GohubWChar* name,
  const GohubWChar* regNumber,
  const GohubWChar* regDate,
  const GohubWChar* validFrom,
  const GohubWChar* validTo,
  const GohubWChar* path);
GohubAttachment* gohub_load_smgs_attachment(
  const char* smgsTypeCode,
```

```
  const char* name,
  const char* regNumber,
  const char* regDate,
  const char* validFrom,
  const char* validTo,
  const char* path);
GohubAttachment* gohub_load_smgs_attachment_w(
  const GohubWChar* smgsTypeCode,
  const GohubWChar* name,
  const GohubWChar* regNumber,
  const GohubWChar* regDate,
  const GohubWChar* validFrom,
  const GohubWChar* validTo,
  const GohubWChar* path);
GohubAttachment* gohub_load_attachment_with_user_data(
  const char* typeCode,
  const char* name,
  const char* regNumber,
  const char* regDate,
  const char* validFrom,
  const char* validTo,
  const char* pathPdf,
  const char* pathUserData);
GohubAttachment* gohub_load_attachment_with_user_data_w(
  const GohubWChar* typeCode,
  const GohubWChar* name,
  const GohubWChar* regNumber,
  const GohubWChar* regDate,
  const GohubWChar* validFrom,
  const GohubWChar* validTo,
  const GohubWChar* pathPdf,
  const GohubWChar* pathUserData);
GohubAttachment* gohub_load_smgs_attachment_with_user_data(
  const char* smgsTypeCode,
  const char* name,
  const char* regNumber,
  const char* regDate,
  const char* validFrom,
  const char* validTo,
  const char* pathPdf,
  const char* pathUserData);
GohubAttachment* gohub_load_smgs_attachment_with_user_data_w(
  const GohubWChar* smgsTypeCode,
  const GohubWChar* name,
  const GohubWChar* regNumber,
  const GohubWChar* regDate,
  const GohubWChar* validFrom,
  const GohubWChar* validTo,
  const GohubWChar* pathPdf,
  const GohubWChar* pathUserData);
GohubBool gohub_send_attachment(GohubConnection* connection, GohubAttachment*
attachment);
GohubAttachment* gohub_query_attachment(GohubConnection* connection, const char*
attachmentId);
GohubAttachment* gohub_query_attachment_w(GohubConnection* connection, const
GohubWChar* attachmentId);
GohubAttachment* gohub_query_attachment_with_user_data(GohubConnection*
connection, const char* attachmentId);
GohubAttachment* gohub_query_attachment_with_user_data_w(GohubConnection*
connection, const GohubWChar* attachmentId);
GohubBool gohub_save_attachment(GohubAttachment* attachment, const char* path);
GohubBool gohub_save_attachment_w(GohubAttachment* attachment, const GohubWChar*
path);
GohubBool gohub_save_attachment_with_user_data(GohubAttachment* attachment, const
char* path);
```

```
GohubBool gohub_save_attachment_with_user_data_w(GohubAttachment* attachment,
const GohubWChar* path);
GohubBool gohub_delete_attachment(GohubConnection* connection, const char*
attachmentId);
GohubBool gohub_delete_attachment_w(GohubConnection* connection, const
GohubWChar* attachmentId);
GohubBool gohub_close_attachment(GohubAttachment* attachment);
const char* gohub_attachment_id(GohubAttachment* attachment);
const GohubWChar* gohub_attachment_id_w(GohubAttachment* attachment);
const char* gohub_attachment_description(GohubAttachment* attachment);
const GohubWChar* gohub_attachment_description_w(GohubAttachment* attachment);
const char* gohub_attachment_type_code(GohubAttachment* attachment);
const GohubWChar* gohub_attachment_type_code_w(GohubAttachment* attachment);
const char* gohub_attachment_smgs_type_code(GohubAttachment* attachment);
const GohubWChar* gohub_attachment_smgs_type_code_w(GohubAttachment* attachment);
const char* gohub_attachment_name(GohubAttachment* attachment);
const GohubWChar* gohub_attachment_name_w(GohubAttachment* attachment);
const char* gohub_attachment_reg_number(GohubAttachment* attachment);
const GohubWChar* gohub_attachment_reg_number_w(GohubAttachment* attachment);
const char* gohub_attachment_reg_date(GohubAttachment* attachment);
const GohubWChar* gohub_attachment_reg_date_w(GohubAttachment* attachment);
const char* gohub_attachment_valid_from(GohubAttachment* attachment);
const GohubWChar* gohub_attachment_valid_from_w(GohubAttachment* attachment);
const char* gohub_attachment_valid_to(GohubAttachment* attachment);
const GohubWChar* gohub_attachment_valid_to_w(GohubAttachment* attachment);
int gohub_attachment_count(GohubDocument* document);
const char* gohub_attachment_id_by_index(GohubDocument* document, int
attachmentIndex);
const GohubWChar* gohub_attachment_id_by_index_w(GohubDocument* document, int
attachmentIndex);

////////////////////////////////////////////////////////////////
// Работа с электронными данными предварительного информирования //
////////////////////////////////////////////////////////////////
GohubEData* gohub_load_edata(
  unsigned int attachmentSmgsTypeCode,
  const char* xmlPath,
  const char* name,
  const char* regNumber,
  const char* regDate,
  const char* validFrom,
  const char* validTo,
  const char* pdfPath);
GohubEData* gohub_load_edata_w(
  unsigned int attachmentSmgsTypeCode,
  const GohubWChar* xmlPath,
  const GohubWChar* name,
  const GohubWChar* regNumber,
  const GohubWChar* regDate,
  const GohubWChar* validFrom,
  const GohubWChar* validTo,
  const GohubWChar* pdfPath);
GohubEData* gohub_load_edata_simple(
  unsigned int attachmentSmgsTypeCode,
  const char* xmlPath);
GohubEData* gohub_load_edata_simple_w(
  unsigned int attachmentSmgsTypeCode,
  const GohubWChar* xmlPath);
GohubBool gohub_edata_load_data(GohubEData* edata, const char* xmlPath);
GohubBool gohub_edata_load_data_w(GohubEData* edata, const GohubWChar* xmlPath);
GohubBool gohub_update_edata(GohubConnection* connection, GohubEData* eData);
GohubBool gohub_send_edata(GohubConnection* connection, GohubEData* eData);
GohubEData* gohub_query_edata(GohubConnection* connection, const char* eDataId);
GohubEData* gohub_query_edata_w(GohubConnection* connection, const GohubWChar*
eDataId);
```

```
GohubEData* gohub_query_edata_for_attachment(GohubConnection* connection, const
char* attachmentId);
GohubEData* gohub_query_edata_for_attachment_w(GohubConnection* connection, const
GohubWChar* attachmentId);
GohubEData* gohub_query_next_edata(GohubConnection* connection, unsigned __int64
lastRevision);
GohubBool gohub_save_edata(GohubEData* eData, const char* path);
GohubBool gohub_save_edata_w(GohubEData* eData, const GohubWChar* path);
GohubBool gohub_close_edata(GohubEData* eData);
const char* gohub_edata_id(GohubEData* eData);
const GohubWChar* gohub_edata_id_w(GohubEData* eData);
unsigned __int64 gohub_edata_revision(GohubEData* eData);
const char* gohub_edata_revision_date(GohubEData* eData);
const GohubWChar* gohub_edata_revision_date_w(GohubEData* eData);
unsigned int gohub_edata_doc_type(GohubEData* eData);
int gohub_edata_status(GohubEData* eData);
const char* gohub_edata_version(GohubEData* eData);
const GohubWChar* gohub_edata_version_w(GohubEData* eData);
const char* gohub_edata_attachment_id(GohubEData* eData);
const GohubWChar* gohub_edata_attachment_id_w(GohubEData* eData);

//////////////////////////////////////////////////////
// Работа с пакетами предварительного информирования //
//////////////////////////////////////////////////////
GohubPiPackage* gohub_query_pi_package(GohubConnection* connection, const char*
piPackageId);
GohubPiPackage* gohub_query_pi_package_w(GohubConnection* connection, const
GohubWChar* piPackageId);
GohubPiPackage* gohub_query_next_pi_package(GohubConnection* connection, unsigned
__int64 lastRevision);
GohubBool gohub_save_pi_package(GohubPiPackage* piPackage, const char* path);
GohubBool gohub_save_pi_package_w(GohubPiPackage* piPackage, const GohubWChar*
path);
GohubBool gohub_close_pi_package(GohubPiPackage* piPackage);
const char* gohub_pi_package_id(GohubPiPackage* piPackage);
const GohubWChar* gohub_pi_package_id_w(GohubPiPackage* piPackage);
unsigned __int64 gohub_pi_package_revision(GohubPiPackage* piPackage);
const char* gohub_pi_package_revision_date(GohubPiPackage* piPackage);
const GohubWChar* gohub_pi_package_revision_date_w(GohubPiPackage* piPackage);
const char* gohub_pi_package_consignment_id(GohubPiPackage* piPackage);
const GohubWChar* gohub_pi_package_consignment_id_w(GohubPiPackage* piPackage);
int gohub_pi_package_status(GohubPiPackage* piPackage);
int gohub_pi_package_pipacktoed_count(GohubPiPackage* piPackage);
GohubPiPackageToEData* gohub_pi_package_pipacktoed(GohubPiPackage* piPackage, int
index);

const char* gohub_pipacktoed_id(GohubPiPackageToEData* piPackageToEData);
const GohubWChar* gohub_pipacktoed_id_w(GohubPiPackageToEData* piPackageToEData);
const char* gohub_pipacktoed_edata_id(GohubPiPackageToEData* piPackageToEData);
const GohubWChar* gohub_pipacktoed_edata_id_w(GohubPiPackageToEData*
piPackageToEData);
const char* gohub_pipacktoed_pi_package_id(GohubPiPackageToEData*
piPackageToEData);
const GohubWChar* gohub_pipacktoed_pi_package_id_w(GohubPiPackageToEData*
piPackageToEData);
const char* gohub_pipacktoed_note(GohubPiPackageToEData* piPackageToEData);
const GohubWChar* gohub_pipacktoed_note_w(GohubPiPackageToEData*
piPackageToEData);
int gohub_pipacktoed_status(GohubPiPackageToEData* piPackageToEData);
const char* gohub_pipacktoed_edata_version(GohubPiPackageToEData*
piPackageToEData);
const GohubWChar* gohub_pipacktoed_edata_version_w(GohubPiPackageToEData*
piPackageToEData);
```

```
GohubPiPackage* gohub_add_edata_to_pi_package(GohubConnection* connection,
GohubEData* edata, const char* piPackageId);
GohubPiPackage* gohub_add_edata_to_pi_package_w(GohubConnection* connection,
GohubEData* edata, const GohubWChar* piPackageId);

////////////////////////////////
// Статусы электронных ФДУ-92 //
////////////////////////////////
typedef enum GohubFdu92Status
{
      gohub_fdu92_status_unknown = -1,
      gohub_fdu92_status_approving = 0,
      gohub_fdu92_status_approving_modified = 1,
      gohub_fdu92_status_confirmed = 2,
      gohub_fdu92_status_canceled = 3,
      gohub_fdu92_status_agreed_noted_sending = 4,
      gohub_fdu92_status_agreed = 5,
      gohub_fdu92_status_agreed_noted = 6,
      gohub_fdu92_status_expired = 7,
      gohub_fdu92_status_agreed_sending = 8,
      gohub_fdu92_status_confirmed_paper = 9,
      gohub_fdu92_status_rejecting = 10,
      gohub_fdu92_status_rejected = 11,
      gohub_fdu92_status_paper = 60,
} GohubFdu92Status;

////////////////////////////////
// Работа с электронными ФДУ-92 //
////////////////////////////////
GohubFdu92* gohub_load_fdu92(const char* id, const char* path);
GohubFdu92* gohub_load_fdu92_w(const GohubWChar* id, const GohubWChar* path);
GohubFdu92* gohub_create_fdu92(const char* id, const char* content);
GohubFdu92* gohub_create_fdu92_w(const char* id, const GohubWChar* content);
GohubFdu92* gohub_query_fdu92(GohubConnection* connection, const char* fdu92Id);
GohubFdu92* gohub_query_fdu92_w(GohubConnection* connection, const GohubWChar*
fdu92Id);
GohubFdu92* gohub_query_fdu92_by_number(GohubConnection* connection, const char*
registration_esr, const char* registration_num);
GohubFdu92* gohub_query_fdu92_by_number_w(GohubConnection* connection, const
GohubWChar* registration_esr, const GohubWChar* registration_num);
GohubFdu92* gohub_query_next_fdu92(GohubConnection* connection, int
lastRevision);
GohubBool gohub_save_fdu92(GohubFdu92* fdu92, const char* path, int codePage);
GohubBool gohub_save_fdu92_w(GohubFdu92* fdu92, const GohubWChar* path, int
codePage);
GohubBool gohub_close_fdu92(GohubFdu92* fdu92);
GohubBool gohub_send_fdu92(GohubConnection* connection, GohubFdu92* document);
GohubBool gohub_reject_fdu92(GohubConnection* connection, const char* id);
GohubBool gohub_reject_fdu92_w(GohubConnection* connection, const GohubWChar*
id);

const char* gohub_fdu92_id(GohubFdu92* fdu92);
const GohubWChar* gohub_fdu92_id_w(GohubFdu92* fdu92);
int gohub_fdu92_revision(GohubFdu92* fdu92);
GohubFdu92Status gohub_fdu92_status(GohubFdu92* fdu92);
const char* gohub_fdu92_text(GohubFdu92* fdu92);
const GohubWChar* gohub_fdu92_text_w(GohubFdu92* fdu92);
int gohub_fdu92_size(GohubFdu92* fdu92);
const char* gohub_fdu92_signer_info(GohubFdu92* fdu92);
const GohubWChar* gohub_fdu92_signer_info_w(GohubFdu92* fdu92);
const char* gohub_fdu92_sign_time(GohubFdu92* fdu92);
const GohubWChar* gohub_fdu92_sign_time_w(GohubFdu92* fdu92);
GohubBool gohub_fdu92_has_signature(GohubFdu92* fdu92);
const GohubWChar* gohub_fdu92_signer_name_w(GohubFdu92* fdu92);
```

```
////////////////////////////////
// Статусы электронных ГУ-46 //
////////////////////////////////
typedef enum GohubGu46Status
{
	gohub_gu46_status_unknown = -1,
	gohub_gu46_status_approving = 0,
	gohub_gu46_status_approving_modified = 1,
	gohub_gu46_status_confirmed = 2,
	gohub_gu46_status_canceled = 3,
	gohub_gu46_status_agreed_noted_sending = 4,
	gohub_gu46_status_agreed = 5,
	gohub_gu46_status_agreed_noted = 6,
	gohub_gu46_status_expired = 7,
	gohub_gu46_status_agreed_sending = 8,
	gohub_gu46_status_confirmed_paper = 9,
	gohub_gu46_status_rejecting = 10,
	gohub_gu46_status_rejected = 11,
	gohub_gu46_status_paper = 60,
} GohubGu46Status;
////////////////////////////////
// Работа с электронными ГУ-46 //
////////////////////////////////
GohubGu46* gohub_load_gu46(const char* id, const char* path);
GohubGu46* gohub_load_gu46_w(const GohubWChar* id, const GohubWChar* path);
GohubGu46* gohub_create_gu46(const char* id, const char* content);
GohubGu46* gohub_create_gu46_w(const GohubWChar* id, const GohubWChar* content);
GohubGu46* gohub_query_gu46(GohubConnection* connection, const char* gu46Id);
GohubGu46* gohub_query_gu46_w(GohubConnection* connection, const GohubWChar*
gu46Id);
GohubGu46* gohub_query_next_gu46(GohubConnection* connection, int lastRevision);
GohubBool gohub_save_gu46(GohubGu46* gu46, const char* path, int codePage);
GohubBool gohub_save_gu46_w(GohubGu46* gu46, const GohubWChar* path, int
codePage);
GohubBool gohub_close_gu46(GohubGu46* gu46);
GohubBool gohub_send_gu46(GohubConnection* connection, GohubGu46* document);
GohubBool gohub_reject_gu46(GohubConnection* connection, const char* id);
GohubBool gohub_reject_gu46_w(GohubConnection* connection, const GohubWChar* id);
GohubGu46* gohub_query_gu46_by_number(GohubConnection* connection, const char*
registration_esr, const char* registration_num);
GohubGu46* gohub_query_gu46_by_number_w(GohubConnection* connection, const
GohubWChar* registration_esr, const GohubWChar* registration_num);

const char* gohub_gu46_id(GohubGu46* gu46);
const GohubWChar* gohub_gu46_id_w(GohubGu46* gu46);
int gohub_gu46_revision(GohubGu46* gu46);
GohubGu46Status gohub_gu46_status(GohubGu46* gu46);
const char* gohub_gu46_text(GohubGu46* gu46);
const GohubWChar* gohub_gu46_text_w(GohubGu46* gu46);
int gohub_gu46_size(GohubGu46* gu46);
const char* gohub_gu46_signer_info(GohubGu46* gu46);
const GohubWChar* gohub_gu46_signer_info_w(GohubGu46* gu46);
const char* gohub_gu46_sign_time(GohubGu46* gu46);
const GohubWChar* gohub_gu46_sign_time_w(GohubGu46* gu46);
GohubBool gohub_gu46_has_signature(GohubGu46* gu46);
const GohubWChar* gohub_gu46_signer_name_w(GohubGu46* gu46);

////////////////////////////////
// Статусы электронных ГУ-45 //
////////////////////////////////
typedef enum GohubGu45Status
{
	gohub_gu45_status_unknown = -1,
	gohub_gu45_status_confirmed = 2,
	gohub_gu45_status_canceled = 3,
```

```
      gohub_gu45_status_confirmed_paper = 9,
      gohub_gu45_status_paper = 60,
} GohubGu45Status;
////////////////////////////////////
// Работа с электронными ГУ-45 //
////////////////////////////////////
GohubGu45* gohub_query_gu45(GohubConnection* connection, const char* gu45Id);
GohubGu45* gohub_query_gu45_w(GohubConnection* connection, const GohubWChar*
gu45Id);
GohubGu45* gohub_query_next_gu45(GohubConnection* connection, int lastRevision);
GohubBool gohub_save_gu45(GohubGu45* gu45, const char* path, int codePage);
GohubBool gohub_save_gu45_w(GohubGu45* gu45, const GohubWChar* path, int
codePage);
GohubBool gohub_close_gu45(GohubGu45* gu45);
GohubGu45* gohub_query_gu45_by_number(GohubConnection* connection, const char*
registration_esr, const char* registration_num, const char* registration_date);
GohubGu45* gohub_query_gu45_by_number_w(GohubConnection* connection, const
GohubWChar* registration_esr, const GohubWChar* registration_num, const
GohubWChar* registration_date);

const char* gohub_gu45_id(GohubGu45* gu45);
const GohubWChar* gohub_gu45_id_w(GohubGu45* gu45);
int gohub_gu45_revision(GohubGu45* gu45);
GohubGu45Status gohub_gu45_status(GohubGu45* gu45);
const char* gohub_gu45_text(GohubGu45* gu45);
const GohubWChar* gohub_gu45_text_w(GohubGu45* gu45);
int gohub_gu45_size(GohubGu45* gu45);
const char* gohub_gu45_signer_info(GohubGu45* gu45);
const GohubWChar* gohub_gu45_signer_info_w(GohubGu45* gu45);
const char* gohub_gu45_sign_time(GohubGu45* gu45);
const GohubWChar* gohub_gu45_sign_time_w(GohubGu45* gu45);
GohubBool gohub_gu45_has_signature(GohubGu45* gu45);
const GohubWChar* gohub_gu45_signer_name_w(GohubGu45* gu45);

////////////////////////////////////////
// Проверка электронно-цифровой подписи //
////////////////////////////////////////
GohubBool gohub_document_has_signature(GohubDocument* document);
GohubBool gohub_document_check_signature(GohubDocument* document);
const char* gohub_document_signer_name(GohubDocument* document);
const GohubWChar* gohub_document_signer_name_w(GohubDocument* document);
const char* gohub_document_signer_info(GohubDocument* document);
const GohubWChar* gohub_document_signer_info_w(GohubDocument* document);
const char* gohub_document_signer_info_by_index(GohubDocument* document, int
index);
const GohubWChar* gohub_document_signer_info_by_index_w(GohubDocument* document,
int index);
const char* gohub_document_signer_name_by_index(GohubDocument* document, int
index);
const GohubWChar* gohub_document_signer_name_by_index_w(GohubDocument* document,
int index);
const char* gohub_document_sign_time_by_index(GohubDocument* document, int
index);
const GohubWChar* gohub_document_sign_time_by_index_w(GohubDocument* document,
int index);
const char* gohub_document_sign_time(GohubDocument* document);
const GohubWChar* gohub_document_sign_time_w(GohubDocument* document);
const int gohub_document_signature_count(GohubDocument* document);
UzRwcDocStatus gohub_document_sign_status_by_index(GohubDocument* document, int
index);
const GohubWChar* gohub_document_sign_error_by_index_w(GohubDocument* document,
int index);
const char* gohub_document_sign_error_by_index(GohubDocument* document, int
index);
```

```
//////////////////////////////////////////
// Наложение электронно-цифровой подписи //
//////////////////////////////////////////
GohubBool gohub_open_private_key(GohubConnection* connection, const char*
passwordToKey);
GohubBool gohub_open_private_key_w(GohubConnection* connection, const GohubWChar*
passwordToKey);
GohubBool gohub_open_private_key_from_path(GohubConnection* connection, const
char* passwordToKey, const char* keyFileName);
GohubBool gohub_open_private_key_from_path_w(GohubConnection* connection, const
GohubWChar* passwordToKey, const GohubWChar* keyFileName);
GohubBool gohub_open_private_key_by_bytes(GohubConnection* connection, const
char* passwordToKey, unsigned char* keyBinary, unsigned int length);
GohubBool gohub_open_private_key_by_bytes_w(GohubConnection* connection, const
GohubWChar* passwordToKey, unsigned char* keyBinary, unsigned int length);
const char* gohub_private_key_owner_name(GohubConnection* connection);
const GohubWChar* gohub_private_key_owner_name_w(GohubConnection* connection);
const char* gohub_private_key_owner_info(GohubConnection* connection);
const GohubWChar* gohub_private_key_owner_info_w(GohubConnection* connection);
GohubBool gohub_sign_document(GohubConnection* connection, GohubDocument*
document);
GohubBool gohub_sign_fdu92(GohubConnection* connection, GohubFdu92* fdu92);
GohubBool gohub_sign_gu46(GohubConnection* connection, GohubGu46* gu46);
GohubBool gohub_close_private_key(GohubConnection* connection);
GohubBool gohub_delete_old_certs_csk_uz(GohubConnection* connection, const char*
passwordToKey);
GohubBool gohub_delete_old_certs_csk_uz_w(GohubConnection* connection, const
GohubWChar* passwordToKey);

//////////////////////////////////////////
// Операции с файлами электронных ключей //
//////////////////////////////////////////
GohubBool gohub_mount_file_key(const char* keyId, const char* path);
GohubBool gohub_mount_file_key_w(const GohubWChar* keyId, const GohubWChar*
path);
GohubBool gohub_unmount_file_key(const char* keyId);
GohubBool gohub_unmount_file_key_w(const GohubWChar* keyId);
int gohub_query_mounted_file_keys();
const char* gohub_mounted_file_key_id(int index);
const GohubWChar* gohub_mounted_file_key_id_w(int index);
const char* gohub_mounted_file_key_dir(int index);
const GohubWChar* gohub_mounted_file_key_dir_w(int index);

//////////////////////
// Обработка ошибок //
//////////////////////
GohubErrcode gohub_last_error_code();
const char* gohub_last_error_title();
const GohubWChar* gohub_last_error_title_w();
const char* gohub_last_error_text();
const GohubWChar* gohub_last_error_text_w();

//////////////////
// Работа с Гу27 //
//////////////////
GohubGu27* gohub_load_gu27(const char* path);
GohubGu27* gohub_load_gu27_w(const GohubWChar* path);
GohubGu27* gohub_create_gu27(const char* content);
GohubGu27* gohub_create_gu27_w(const GohubWChar* content);
GohubGu27* gohub_query_gu27(GohubConnection* connection, const char* gu27Id);
GohubGu27* gohub_query_gu27_w(GohubConnection* connection, const GohubWChar*
gu27Id);
GohubGu27* gohub_query_next_gu27(GohubConnection* connection, int lastRevision);
const char* gohub_gu27_id(GohubGu27* gu27);
const GohubWChar* gohub_gu27_id_w(GohubGu27* gu27);
```

```
int gohub_gu27_revision(GohubGu27* gu27);
const char* gohub_gu27_text(GohubGu27* gu27);
const GohubWChar* gohub_gu27_text_w(GohubGu27* gu27);
const char* gohub_gu27_data_text(GohubGu27* gu27, int gu27Version);
const GohubWChar* gohub_gu27_data_text_w(GohubGu27* gu27, int gu27Version);
int gohub_gu27_size(GohubGu27* gu27);
GohubBool gohub_send_gu27(GohubConnection* connection, GohubGu27* gu27);
GohubBool gohub_send_received_gu27(GohubConnection* connection, GohubGu27* gu27,
const char* gu27Id);
GohubBool gohub_send_received_gu27_w(GohubConnection* connection, GohubGu27*
gu27, const GohubWChar* gu27Id);
GohubBool gohub_save_gu27(GohubGu27* gu27, const char* path, int codePage);
GohubBool gohub_save_gu27_w(GohubGu27* gu27, const GohubWChar* path, int
codePage);
GohubBool gohub_save_gu27_data(GohubGu27* gu27, const char* path, int codePage,
int gu27Version);
GohubBool gohub_save_gu27_data_w(GohubGu27* gu27, const GohubWChar* path, int
codePage, int gu27Version);
GohubBool gohub_close_gu27(GohubGu27* gu27);
GohubBool gohub_reclaim_gu27(GohubConnection* connection, const char* gu27Id);
GohubBool gohub_reclaim_gu27_w(GohubConnection* connection, const GohubWChar*
gu27Id);
GohubBool gohub_delete_gu27(GohubConnection* connection, const char* gu27Id);
GohubBool gohub_delete_gu27_w(GohubConnection* connection, const GohubWChar*
gu27Id);
GohubGu27Status gohub_gu27_status(GohubGu27* gu27);

GohubBool gohub_sign_gu27(GohubConnection* connection, GohubGu27* gu27);

////////////////////////////////////////////
// Проверка электронно-цифровой подписи GU27 //
////////////////////////////////////////////
GohubBool gohub_gu27_has_signature(GohubGu27* gu27);
GohubBool gohub_gu27_check_signature(GohubGu27* gu27);
const char* gohub_gu27_signer_name(GohubGu27* gu27);
const GohubWChar* gohub_gu27_signer_name_w(GohubGu27* gu27);
const char* gohub_gu27_signer_info(GohubGu27* gu27);
const GohubWChar* gohub_gu27_signer_info_w(GohubGu27* gu27);
const char* gohub_Gu27_sign_time(GohubGu27* gu27);
const GohubWChar* gohub_gu27_sign_time_w(GohubGu27* gu27);

//////////////////////////
// Работа с АС "Месплан" //
//////////////////////////
const char* gohub_get_mp_months(GohubConnection* connection, int codePage);
const GohubWChar* gohub_get_mp_months_w(GohubConnection* connection);
GohubBool gohub_query_and_save_orders_for_month(GohubConnection* connection,
const char* month, const char* path);
GohubBool gohub_query_and_save_orders_for_month_w(GohubConnection* connection,
const GohubWChar* month, const GohubWChar* path);
GohubBool gohub_query_and_save_orders_for_month_with_relogin(GohubConnection*
connection, const char* month, const char* login, const char* password, const
char* path);
GohubBool gohub_query_and_save_orders_for_month_with_relogin_w(GohubConnection*
connection, const GohubWChar* month, const GohubWChar* login, const GohubWChar*
password, const GohubWChar* path);

////////////////////////////////////////////
// Работа с документами информационных услуг //
////////////////////////////////////////////
GohubInformServicesDoc* gohub_query_inform_services_document(GohubConnection*
connection, unsigned __int64 docId);
GohubInformServicesDoc*
gohub_query_next_inform_services_document(GohubConnection* connection, unsigned
__int64 lastRevision);
```

```
GohubBool gohub_save_inform_services_document(GohubInformServicesDoc* document,
const char* path);
GohubBool gohub_save_inform_services_document_w(GohubInformServicesDoc* document,
const GohubWChar* path);
GohubBool gohub_saveXml_inform_services_document(GohubInformServicesDoc*
document, const char* path);
GohubBool gohub_saveXml_inform_services_document_w(GohubInformServicesDoc*
document, const GohubWChar* path);
GohubBool gohub_close_inform_services_document(GohubInformServicesDoc* document);
unsigned __int64 gohub_inform_services_document_id(GohubInformServicesDoc*
document);
unsigned __int64 gohub_inform_services_document_revision(GohubInformServicesDoc*
document);
const char* gohub_inform_services_document_filename(GohubInformServicesDoc*
document);
const GohubWChar*
gohub_inform_services_document_filename_w(GohubInformServicesDoc* document);
const char* gohub_inform_services_document_comment(GohubInformServicesDoc*
document);
const GohubWChar*
gohub_inform_services_document_comment_w(GohubInformServicesDoc* document);
const char* gohub_inform_services_document_created_date(GohubInformServicesDoc*
document);
const GohubWChar*
gohub_inform_services_document_created_date_w(GohubInformServicesDoc* document);
const char* gohub_inform_services_document_doc_date(GohubInformServicesDoc*
document);
const GohubWChar*
gohub_inform_services_document_doc_date_w(GohubInformServicesDoc* document);
gohub_inform_services_document_doc_is_empty(GohubInformServicesDoc* document);

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
// Работа с перечнем о заказе на согласование перевозки по данным календаря
планирования перевозок зерновых грузов (за последние 5 дней от текущей даты)//
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
GohubDispatchInfo* gohub_query_dispatch_info(GohubConnection* connection, const
char* start_esr, const char* end_esr);
GohubDispatchInfo* gohub_query_dispatch_info_w(GohubConnection* connection, const
GohubWChar* start_esr, const GohubWChar* end_esr);
const char*  gohub_document_info_description(GohubDispatchInfo* document, int
index);
int gohub_document_info_count(GohubDispatchInfo* document);
GohubBool gohub_close_dispatch_info(GohubDispatchInfo* document);
const char*  gohub_document_info_number(GohubDispatchInfo* document, int index);
const char*  gohub_document_info_type(GohubDispatchInfo* document, int index);
const char*  gohub_document_info_date(GohubDispatchInfo* document, int index);
const char*  gohub_document_info_wag_owner(GohubDispatchInfo* document, int
index);
const char*  gohub_document_info_is_empty(GohubDispatchInfo* document, int
index);
const GohubWChar*  gohub_document_info_number_w(GohubDispatchInfo* document, int
index);
const GohubWChar*  gohub_document_info_type_w(GohubDispatchInfo* document, int
index);
const GohubWChar*  gohub_document_info_date_w(GohubDispatchInfo* document, int
index);
const GohubWChar*  gohub_document_info_wag_owner_w(GohubDispatchInfo* document,
int index);
const GohubWChar*  gohub_document_info_is_empty_w(GohubDispatchInfo* document,
int index);
const GohubWChar*  gohub_document_info_description_w(GohubDispatchInfo* document,
int index);
```

```
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
// Работа з перечнем про Номер Заказа Согласования сокращения сроком доставки///
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
GohubBool gohub_close_pstd_info(GohubPSTDInfo* document);
GohubPSTDInfo* gohub_query_pstd_info_w(GohubConnection* connection, const
GohubWChar* work_kind, const GohubWChar* departure_esr, const GohubWChar*
arrival_esr);
GohubPSTDInfo* gohub_query_pstd_info(GohubConnection* connection, const char*
work_kind, const char* departure_esr, const char* arrival_esr);
const char*  gohub_pstd_info_number(GohubPSTDInfo* document, int index);
const char*  gohub_pstd_info_arrivaldate(GohubPSTDInfo* document, int index);
const char*  gohub_pstd_info_departuredate(GohubPSTDInfo* document, int index);
const char*  gohub_pstd_info_reqdate(GohubPSTDInfo* document, int index);
const GohubWChar*  gohub_pstd_info_number_w(GohubPSTDInfo* document, int index);
const GohubWChar*  gohub_pstd_info_arrivaldate_w(GohubPSTDInfo* document, int
index);
const GohubWChar*  gohub_pstd_info_departuredate_w(GohubPSTDInfo* document, int
index);
const GohubWChar*  gohub_pstd_info_reqdate_w(GohubPSTDInfo* document, int index);
int gohub_pstd_info_count(GohubPSTDInfo* document);


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

#endif //GOHUB_CLIENT_H_
```

ПРИЛОЖЕНИЕ Б. Заголовочный файл ***gohub.client.errors.h***

```
#ifndef GOHUB_CLIENT_ERRORS_H_
#define GOHUB_CLIENT_ERRORS_H_

#ifdef __cplusplus
extern "C" {
#endif

//////////////////
// Коды ошибок //
//////////////////

typedef enum GohubErrcode
{
  // Operation is success
  gohub_success,

  // Gohub server connection could not be established
  gohub_server_connection_could_not_be_established,

  // Gohub server inaccessible
  gohub_server_inaccessible,

  // Document creation failed
  gohub_document_creation_failed,

  // Document query failed
  gohub_document_query_failed,

  // Next document query failed
  gohub_next_document_query_failed,

  // Document sending failed
  gohub_document_sending_failed,

  // Document saving failed
  gohub_document_saving_failed,

  // Document loading failed
  gohub_document_loading_failed,

  // Invalid code page
  gohub_invalid_code_page,

  // Private key could not be opened
  gohub_private_key_could_not_be_opened,

  // Private key path could not be opened
  gohub_private_key_path_could_not_be_opened,

  // Private key bytes could not be opened
  gohub_private_key_bytes_could_not_be_opened,

  // Private key is inaccessible
  gohub_private_key_is_inaccessible,

  // Document could not be signed
  gohub_document_could_not_be_signed,

  // Document signature verification failed
  gohub_document_signature_verification_failed,

  // Document has not signature
  gohub_document_has_not_signature,
```

```
// Gohub Client is obsolete. Upgrade your Gohub Client version
gohub_client_is_obsolete,

// Gohub Serveris obsolete. Upgrade your Gohub Server version
gohub_server_is_obsolete,

//Document reclamation failed
gohub_document_reclamation_failed,

//Document deletion failed
gohub_document_deletion_failed,

// Attachment creation failed
gohub_attachment_creation_failed,

// Attachment sending failed
gohub_attachment_sending_failed,

// Attachment query failed
gohub_attachment_query_failed,

// Attachment deletion failed
gohub_attachment_deletion_failed,

// Mount of file key failed
gohub_mount_of_file_key_failed,

// Unmount of file key failed
gohub_unmount_of_file_key_failed,

// Enumerating of file keys failed
gohub_enumerating_of_file_keys_failed,

// Mounted file key inaccessible
gohub_mounted_file_key_inaccessible,

// EData creation failed
gohub_edata_creation_failed,

// EData sending failed
gohub_edata_sending_failed,

// EData updating failed
gohub_edata_updating_failed,

// EData query failed
gohub_edata_query_failed,

// Next EData query failed
gohub_next_edata_query_failed,

// PiPackage query failed
gohub_pipackage_query_failed,

// Next PiPackage query failed
gohub_next_pipackage_query_failed,

// MP months query failed
gohub_mp_months_query_failed,

// Orders of month query failed
gohub_orders_of_months_query_failed,

// Internal program error
```

```
  gohub_programm_error = 0x1000,

  // Client application performed invalid operation
  gohub_invalid_operation,

  // "Inform services doc saving failed"
  gohub_inform_services_doc_saving_failed,

  // "Inform services doc query failed"
  gohub_inform_services_doc_query_failed,

  // "Query changes inform services failed")]
  gohub_query_changes_inform_services_failed,

  // "Query next inform services document")]
  gohub_query_next_inform_services_document_failed,

} GohubErrcode;

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

#endif //GOHUB_CLIENT_ERRORS_H_
```

ПРИЛОЖЕНИЕ В. Файл описания COM-интерфейсов GohubClientCOM.idl

```
import "oaidl.idl";
import "ocidl.idl";

interface IGohubClient;
interface IGohubDocument;
interface IGohubAttachment;
interface IGohubEData;
interface IGohubPiPackage;
interface IGohubPiPackageToEData;
interface IGohubConnection;
interface IGohubSignerInfo;
interface IGohubError;
interface IGohubFdu92;
interface IGohubGu46;
interface IGohubGu45;
interface IGohubGu27;
interface IGohubInformServicesDocument;
interface IGohubPSTDInfo;

[
	object,
	uuid(B0E1F579-5836-4E97-9340-58B092418947),
	dual,
	nonextensible,
	helpstring("IGohubSignerInfo Interface"),
	pointer_default(unique)
]
interface IGohubSignerInfo : IDispatch{
  [propget, id(1), helpstring("property Name")] HRESULT Name([out, retval] BSTR*
pVal);
  [propget, id(2), helpstring("property Subject")] HRESULT Subject([out, retval]
BSTR* pVal);
};
[
	object,
	uuid(0D5D6225-8E07-46E2-8B8D-9C0966481994),
	dual,
	nonextensible,
	helpstring("IGohubDocument Interface"),
	pointer_default(unique)
]
interface IGohubDocument : IDispatch{
  [propget, id(1), helpstring("property Id")] HRESULT Id([out, retval] BSTR*
pVal);
  [propget, id(2), helpstring("property Text")] HRESULT Text([out, retval] BSTR*
pVal);
  [propget, id(3), helpstring("property Revision")] HRESULT Revision([out,
retval] LONG* pVal);
  [propget, id(4), helpstring("property HasSignature")] HRESULT
HasSignature([out, retval] VARIANT_BOOL* pVal);
  [propget, id(5), helpstring("property Signer")] HRESULT Signer([out, retval]
IGohubSignerInfo** pVal);
  [propget, id(6), helpstring("property SignTime")] HRESULT SignTime([out,
retval] BSTR* pVal);
  [id(7), helpstring("method Save")] HRESULT Save([in] BSTR path, [in] LONG
codePage, [out,retval] VARIANT_BOOL* result);
  [id(8), helpstring("method Sign")] HRESULT Sign([in] IGohubConnection*
connection, [out,retval] VARIANT_BOOL* result);
  [id(9), helpstring("method Send")] HRESULT Send([in] IGohubConnection*
connection, [out,retval] VARIANT_BOOL* result);
  [id(10), helpstring("method Close")] HRESULT Close(void);
  [propget, id(11), helpstring("property AttachmentsCount")] HRESULT
AttachmentsCount([out, retval] LONG* pVal);
```

```
  [id(12), helpstring("method GetAttachmentIdByIndex")] HRESULT
GetAttachmentIdByIndex([in] LONG index, [out,retval] BSTR* result);
  [propget, id(15), helpstring("property Status")] HRESULT Status([out, retval]
LONG* pVal);
  [id(16), helpstring("method SendReceived")] HRESULT SendReceived([in]
IGohubConnection* connection, [in] BSTR docId, [out,retval] VARIANT_BOOL*
result);
  [id(17), helpstring("method SaveData")] HRESULT SaveData([in] BSTR path, [in]
LONG codePage, [in] LONG epdVersion, [out,retval] VARIANT_BOOL* result);
  [id(18), helpstring("method DataText")] HRESULT DataText([in] LONG epdVersion,
[out, retval] BSTR* pVal);
  [propget, id(19), helpstring("property MeasureEquipNum")] HRESULT
MeasureEquipNum([out, retval] BSTR* pVal);
  [propput, id(19), helpstring("property MeasureEquipNum")] HRESULT
MeasureEquipNum([in] BSTR pVal);
  [id(20), helpstring("method set_VerifiedEmptyWeightForWagon")] HRESULT
set_VerifiedEmptyWeightForWagon([in] LONG wagonIndex, [in] LONG weight);
  [id(21), helpstring("method get_VerifiedEmptyWeightForWagon")] HRESULT
get_VerifiedEmptyWeightForWagon([in] LONG wagonIndex, [out,retval] LONG* weight);
  [propget, id(22), helpstring("property BusinessUnitNum")] HRESULT
BusinessUnitNum([out, retval] BSTR* pVal);
  [propput, id(22), helpstring("property BusinessUnitNum")] HRESULT
BusinessUnitNum([in] BSTR pVal);
  [propget, id(23), helpstring("property ForeignNotAccept")] HRESULT
ForeignNotAccept([out, retval] VARIANT_BOOL* pVal);
  [propget, id(24), helpstring("property WarrantType")] HRESULT WarrantType([out,
retval] LONG* pVal);
  [propput, id(24), helpstring("property WarrantType")] HRESULT WarrantType([in]
LONG newVal);
  [id(25), helpstring("method SetVerifiedEmptyWeightForWagon")] HRESULT
SetVerifiedEmptyWeightForWagon([in] LONG wagonIndex, [in] LONG weight);
  [id(26), helpstring("method GetVerifiedEmptyWeightForWagon")] HRESULT
GetVerifiedEmptyWeightForWagon([in] LONG wagonIndex, [out,retval] LONG* weight);
  [id(27), helpstring("method GetOTPR")] HRESULT GetOTPR([in] BSTR path,
[out,retval] VARIANT_BOOL* result);
  [propget, id(28), helpstring("property OTPRString")] HRESULT OTPRString([out,
retval] BSTR* pVal);
  [propget, id(29), helpstring("property Warning")] HRESULT Warning([out, retval]
BSTR* pVal);
  [id(30), helpstring("property SignerByIndex")] HRESULT SignerByIndex([in] LONG
index, [out, retval] IGohubSignerInfo** pVal);
  [propget, id(31), helpstring("property SignerCount")] HRESULT SignerCount([out,
retval] LONG* length);
  [id(32), helpstring("property SignTimeByIndex")] HRESULT SignTimeByIndex([in]
LONG index, [out, retval] BSTR* pVal);
  [id(33), helpstring("property SignStatusByIndex")] HRESULT
SignStatusByIndex([in] LONG index, [out, retval] LONG* pVal);
  [id(34), helpstring("property SignErrorByIndex")] HRESULT SignErrorByIndex([in]
LONG index, [out, retval] BSTR* pVal);
};
[
      object,
      uuid(E9967B9D-1141-47BA-A5C4-573FB02DB396),
      dual,
      nonextensible,
      helpstring("IGohubConnection Interface"),
      pointer_default(unique)
]
interface IGohubConnection : IDispatch{
  [propget, id(1), helpstring("property Signer")] HRESULT Signer([out, retval]
IGohubSignerInfo** pVal);
  [id(2), helpstring("method Close")] HRESULT Close(void);
  [id(3), helpstring("method QueryDocument")] HRESULT QueryDocument([in] BSTR
docId, [out,retval] IGohubDocument** document);
```

```
  [id(4), helpstring("method QueryNextDocument")] HRESULT QueryNextDocument([in]
LONG revision, [out,retval] IGohubDocument** document);
  [id(5), helpstring("method OpenPrivateKey")] HRESULT OpenPrivateKey([in] BSTR
password, [out,retval] VARIANT_BOOL* result);
  [id(6), helpstring("method ReclaimDocument")] HRESULT ReclaimDocument([in] BSTR
docId, [out,retval] VARIANT_BOOL* result);
  [id(7), helpstring("method DeleteDocument")] HRESULT DeleteDocument([in] BSTR
docId, [out,retval] VARIANT_BOOL* result);
  [id(8), helpstring("method QueryAttachment")] HRESULT QueryAttachment([in] BSTR
attachmentId, [out,retval] IGohubAttachment** attachment);
  [id(9), helpstring("method DeleteAttachment")] HRESULT DeleteAttachment([in]
BSTR attachmentId, [out,retval] VARIANT_BOOL* result);
  [propget, id(10), helpstring("property FilterByDocumentStatus")] HRESULT
FilterByDocumentStatus([out, retval] LONG* pVal);
  [propput, id(10), helpstring("property FilterByDocumentStatus")] HRESULT
FilterByDocumentStatus([in] LONG newVal);
  [propget, id(11), helpstring("property FilterByDocumentNumber")] HRESULT
FilterByDocumentNumber([out, retval] BSTR* pVal);
  [propput, id(11), helpstring("property FilterByDocumentNumber")] HRESULT
FilterByDocumentNumber([in] BSTR newVal);
  [propget, id(12), helpstring("property FilterByWagonNumber")] HRESULT
FilterByWagonNumber([out, retval] BSTR* pVal);
  [propput, id(12), helpstring("property FilterByWagonNumber")] HRESULT
FilterByWagonNumber([in] BSTR newVal);
  [propget, id(13), helpstring("property FilterByDepartureClientCode")] HRESULT
FilterByDepartureClientCode([out, retval] BSTR* pVal);
  [propput, id(13), helpstring("property FilterByDepartureClientCode")] HRESULT
FilterByDepartureClientCode([in] BSTR newVal);
  [propget, id(14), helpstring("property FilterByDeparturePayerCode")] HRESULT
FilterByDeparturePayerCode([out, retval] BSTR* pVal);
  [propput, id(14), helpstring("property FilterByDeparturePayerCode")] HRESULT
FilterByDeparturePayerCode([in] BSTR newVal);
  [propget, id(15), helpstring("property FilterByDepartureStationCode")] HRESULT
FilterByDepartureStationCode([out, retval] BSTR* pVal);
  [propput, id(15), helpstring("property FilterByDepartureStationCode")] HRESULT
FilterByDepartureStationCode([in] BSTR newVal);
  [propget, id(16), helpstring("property FilterByArrivalClientCode")] HRESULT
FilterByArrivalClientCode([out, retval] BSTR* pVal);
  [propput, id(16), helpstring("property FilterByArrivalClientCode")] HRESULT
FilterByArrivalClientCode([in] BSTR newVal);
  [propget, id(17), helpstring("property FilterByArrivalPayerCode")] HRESULT
FilterByArrivalPayerCode([out, retval] BSTR* pVal);
  [propput, id(17), helpstring("property FilterByArrivalPayerCode")] HRESULT
FilterByArrivalPayerCode([in] BSTR newVal);
  [propget, id(18), helpstring("property FilterByArrivalStationCode")] HRESULT
FilterByArrivalStationCode([out, retval] BSTR* pVal);
  [propput, id(18), helpstring("property FilterByArrivalStationCode")] HRESULT
FilterByArrivalStationCode([in] BSTR newVal);
  [id(19), helpstring("method ClearAllFilters")] HRESULT
ClearAllFilters([out,retval] VARIANT_BOOL* result);
  [id(20), helpstring("method QueryAndSaveDocumentPrintableForm")] HRESULT
QueryAndSaveDocumentPrintableForm([in] BSTR docId, [in] BSTR path, [out,retval]
VARIANT_BOOL* result);
  [id(21), helpstring("method QueryAndSaveFdu92PrintableForm")] HRESULT
QueryAndSaveFdu92PrintableForm([in] BSTR docId, [in] BSTR path, [out,retval]
VARIANT_BOOL* result);
  [id(22), helpstring("method QueryAndSaveGu45PrintableForm")] HRESULT
QueryAndSaveGu45PrintableForm([in] BSTR docId, [in] BSTR path, [out,retval]
VARIANT_BOOL* result);
  [id(23), helpstring("method QueryAndSaveGu46PrintableForm")] HRESULT
QueryAndSaveGu46PrintableForm([in] BSTR docId, [in] BSTR path, [out,retval]
VARIANT_BOOL* result);
  [id(24), helpstring("method QueryFdu92")] HRESULT QueryFdu92([in] BSTR id,
[out,retval] IGohubFdu92** fdu92);
```

```
  [id(25), helpstring("method QueryNextFdu92")] HRESULT QueryNextFdu92([in] LONG
revision, [out,retval] IGohubFdu92** fdu92);
  [id(26), helpstring("method QueryGu46")] HRESULT QueryGu46([in] BSTR id,
[out,retval] IGohubGu46** gu46);
  [id(27), helpstring("method QueryNextGu46")] HRESULT QueryNextGu46([in] LONG
revision, [out,retval] IGohubGu46** gu46);
  [id(28), helpstring("method QueryGu45")] HRESULT QueryGu45([in] BSTR id,
[out,retval] IGohubGu45** gu45);
  [id(29), helpstring("method QueryNextGu45")] HRESULT QueryNextGu45([in] LONG
revision, [out,retval] IGohubGu45** gu45);
  [id(30), helpstring("method RejectFdu92")] HRESULT RejectFdu92(BSTR id);
  [id(31), helpstring("method RejectGu46")] HRESULT RejectGu46(BSTR id);
  [id(32), helpstring("method QueryEData")] HRESULT QueryEData([in] BSTR eDataId,
[out,retval] IGohubEData** eData);
  [id(33), helpstring("method QueryNextEData")] HRESULT QueryNextEData([in]
ULONGLONG revision, [out,retval] IGohubEData** eData);
  [id(34), helpstring("method QueryPiPackage")] HRESULT QueryPiPackage([in] BSTR
eDataId, [out,retval] IGohubPiPackage** eData);
  [id(35), helpstring("method QueryNextPiPackage")] HRESULT
QueryNextPiPackage([in] ULONGLONG revision, [out,retval] IGohubPiPackage**
eData);
  [id(36), helpstring("method AddEDataToPiPackage")] HRESULT
AddEDataToPiPackage([in] IGohubEData* eData, [in] BSTR piPackageId, [out,retval]
IGohubPiPackage** piPackage);
  [id(37), helpstring("method QueryGu27")] HRESULT QueryGu27([in] BSTR gu27Id,
[out,retval] IGohubGu27** gu27);
  [id(38), helpstring("method QueryNextGu27")] HRESULT QueryNextGu27([in] LONG
revision, [out,retval] IGohubGu27** gu27);
  [id(39), helpstring("method ReclaimGu27")] HRESULT ReclaimGu27([in] BSTR
gu27Id, [out,retval] VARIANT_BOOL* result);
  [id(40), helpstring("method DeleteGu27")] HRESULT DeleteGu27([in] BSTR gu27Id,
[out,retval] VARIANT_BOOL* result);
  [id(41), helpstring("method QueryAndSaveGu27PrintableForm")] HRESULT
QueryAndSaveGu27PrintableForm([in] BSTR gu27Id, [in] BSTR path, [out,retval]
VARIANT_BOOL* result);
  [id(42), helpstring("method QueryAttachmentWithUserData")] HRESULT
QueryAttachmentWithUserData([in] BSTR attachmentId, [out,retval]
IGohubAttachment** attachment);
  [id(43), helpstring("method GetMPMonths")] HRESULT GetMPMonths([out,retval]
BSTR* result);
  [id(44), helpstring("method QueryAndSaveOrdersForMonth")] HRESULT
QueryAndSaveOrdersForMonth([in] BSTR month, [in] BSTR path);
  [id(45), helpstring("method QueryAndSaveOrdersForMonthWithRelogin")] HRESULT
QueryAndSaveOrdersForMonthWithRelogin([in] BSTR month, [in] BSTR login, [in] BSTR
password, [in] BSTR path);
  [id(46), helpstring("method SaveDocumentPrintableForm")] HRESULT
SaveDocumentPrintableForm([in] BSTR docId, [in] BSTR path, [out,retval]
VARIANT_BOOL* result);
  [id(47), helpstring("method SaveOrdersForMonthWithRelogin")] HRESULT
SaveOrdersForMonthWithRelogin([in] BSTR month, [in] BSTR login, [in] BSTR
password, [in] BSTR path);
  [id(48), helpstring("method QueryEDataForAttachment")] HRESULT
QueryEDataForAttachment([in] BSTR attachmentId, [out,retval] IGohubEData**
eData);
  [id(49), helpstring("method QueryInfServsDoc")] HRESULT QueryInfServsDoc([in]
ULONGLONG docId, [out,retval] IGohubInformServicesDocument** document);
  [id(50), helpstring("method QueryNextInfServsDoc")] HRESULT
QueryNextInfServsDoc([in] ULONGLONG revision, [out,retval]
IGohubInformServicesDocument** document);
  [id(51), helpstring("method OpenPrivateKeyFromPath")] HRESULT
OpenPrivateKeyFromPath([in] BSTR password, [in] BSTR path, [out,retval]
VARIANT_BOOL* result);
  [id(52), helpstring("method QueryFdu92ByNumber")] HRESULT
QueryFdu92ByNumber([in] BSTR registration_esr, [in]BSTR registration_num,
[out,retval] IGohubFdu92** fdu92);
```

```
  [id(53), helpstring("method QueryDispatchInfo")] HRESULT QueryDispatchInfo([in]
BSTR dispatch_esr, [in]BSTR arrival_num, [out,retval] IGohubDispatchInfo** disp);
  [id(54), helpstring("method QueryGu45ByNumber")] HRESULT QueryGu45ByNumber([in]
BSTR registration_esr, [in]BSTR registration_num, [in]BSTR registration_date,
[out,retval] IGohubGu45** gu45);
  [id(55), helpstring("method QueryGu46ByNumber")] HRESULT QueryGu46ByNumber([in]
BSTR registration_esr, [in]BSTR registration_num, [out,retval] IGohubGu46**
gu46);
  [id(56), helpstring("method QueryNextDocument2")] HRESULT
QueryNextDocument2([in] LONG revision, [out,retval] IGohubDocument** document);
[id(57), helpstring("method QueryPSTDInfo")] HRESULT QueryPSTDInfo([in]BSTR
work_kind, [in]BSTR dispatch_esr, [in]BSTR arrival_esr,
[out,retval]IGohubPSTDInfo** document);
[id(58), helpstring("method get_KeysList")] HRESULT get_KeysList([out, retval]
BSTR* pVal);
  [id(59), helpstring("method OpenPrivateKeyByIndex")] HRESULT
OpenPrivateKeyByIndex([in] BSTR password, [in] LONG MediaIndex, [in] LONG
DeviceIndex, [out,retval] VARIANT_BOOL* result);
};
 [
      object,
      uuid(F1077417-21D9-4871-84A2-9F89525E7214),
      dual,
      nonextensible,
      helpstring("IGohubError Interface"),
      pointer_default(unique)
]
interface IGohubError : IDispatch{
  [propget, id(1), helpstring("property Code")] HRESULT Code([out, retval] LONG*
pVal);
  [propget, id(2), helpstring("property Title")] HRESULT Title([out, retval]
BSTR* pVal);
  [propget, id(3), helpstring("property Text")] HRESULT Text([out, retval] BSTR*
pVal);
};
[
      object,
      uuid(ABDA6C07-5320-4F28-B995-FADE037D0A82),
      dual,
      nonextensible,
      helpstring("IGohubClient Interface"),
      pointer_default(unique)
]
interface IGohubClient : IDispatch{
  [id(1), helpstring("method GetLastError")] HRESULT GetLastError([out,retval]
IGohubError** result);
  [id(2), helpstring("method Connect")] HRESULT Connect([in] BSTR host, [in] LONG
port, [out,retval] IGohubConnection** connection);
  [id(3), helpstring("method CreateDocument")] HRESULT CreateDocument([in] BSTR
xmlText, [out,retval] IGohubDocument** document);
  [id(4), helpstring("method LoadDocument")] HRESULT LoadDocument([in] BSTR path,
[out,retval] IGohubDocument** result);
  [id(5), helpstring("method LoadAttachment")] HRESULT LoadAttachment([in] BSTR
typeCode, [in] BSTR name, [in] BSTR regNumber, [in] BSTR regDate, [in] BSTR
validFrom, [in] BSTR validTo, [in] BSTR path, [out,retval] IGohubAttachment**
result);
  [id(6), helpstring("method MountFileKey")] HRESULT MountFileKey([in] BSTR
keyId, [in] BSTR keyDir, [out,retval] VARIANT_BOOL* result);
  [id(7), helpstring("method UnmountFileKey")] HRESULT UnmountFileKey([in] BSTR
keyId, [out,retval] VARIANT_BOOL* result);
  [id(8), helpstring("method QueryMountedKeys")] HRESULT
QueryMountedKeys([out,retval] LONG* result);
  [id(9), helpstring("method GetMountedKeyId")] HRESULT GetMountedKeyId([in] LONG
index, [out,retval] BSTR* keyId);
```

```
  [id(10), helpstring("method GetMountedKeyDir")] HRESULT GetMountedKeyDir([in]
LONG index, [out,retval] BSTR* keyDir);
  [id(11), helpstring("method LoadEData")] HRESULT LoadEData([in] UINT codeType,
[in] BSTR xmlPath, [in] BSTR name, [in] BSTR regNumber, [in] BSTR regDate, [in]
BSTR validFrom, [in] BSTR validTo, [in] BSTR pdfPath, [out,retval] IGohubEData**
result);
  [id(12), helpstring("method LoadEDataSimple")] HRESULT LoadEDataSimple([in]
UINT codeType, [in] BSTR xmlPath, [out,retval] IGohubEData** result);
  [id(13), helpstring("method CreateGu27")] HRESULT CreateGu27([in] BSTR xmlText,
[out,retval] IGohubGu27** gu27);
  [id(14), helpstring("method LoadGu27")] HRESULT LoadGu27([in] BSTR path,
[out,retval] IGohubGu27** result);
  [id(15), helpstring("method LoadSmgsAttachment")] HRESULT
LoadSmgsAttachment([in] BSTR smgsTypeCode, [in] BSTR name, [in] BSTR regNumber,
[in] BSTR regDate, [in] BSTR validFrom, [in] BSTR validTo, [in] BSTR path,
[out,retval] IGohubAttachment** result);
  [id(16), helpstring("method LoadAttachmentWithUserData")] HRESULT
LoadAttachmentWithUserData([in] BSTR typeCode, [in] BSTR name, [in] BSTR
regNumber, [in] BSTR regDate, [in] BSTR validFrom, [in] BSTR validTo, [in] BSTR
path, [in] BSTR pathUserData, [out,retval] IGohubAttachment** attachment);
  [id(17), helpstring("method LoadSmgsAttachmentWithUserData")] HRESULT
LoadSmgsAttachmentWithUserData([in] BSTR smgsTypeCode, [in] BSTR name, [in] BSTR
regNumber, [in] BSTR regDate, [in] BSTR validFrom, [in] BSTR validTo, [in] BSTR
path, [in] BSTR pathUserData, [out,retval] IGohubAttachment** attachment);
  [id(18), helpstring("method LoadFdu92")] HRESULT LoadFdu92([in] BSTR ident,
[in] BSTR path, [out,retval] IGohubFdu92** result);
  [id(19), helpstring("method LoadGu46")] HRESULT LoadGu46([in] BSTR ident, [in]
BSTR path, [out,retval] IGohubGu46** result);
};
[
      object,
      uuid(044603D2-574E-463C-87EC-DCD98C30F319),
      dual,
      nonextensible,
      helpstring("IGohubAttachment Interface"),
      pointer_default(unique)
]
interface IGohubAttachment : IDispatch{
  [propget, id(1), helpstring("property Id")] HRESULT Id([out, retval] BSTR*
pVal);
  [propget, id(2), helpstring("property TypeCode")] HRESULT TypeCode([out,
retval] BSTR* pVal);
  [propget, id(3), helpstring("property Name")] HRESULT Name([out, retval] BSTR*
pVal);
  [propget, id(4), helpstring("property Description")] HRESULT Description([out,
retval] BSTR* pVal);
  [propget, id(5), helpstring("property RegNumber")] HRESULT RegNumber([out,
retval] BSTR* pVal);
  [propget, id(6), helpstring("property RegDate")] HRESULT RegDate([out, retval]
BSTR* pVal);
  [propget, id(7), helpstring("property ValidFrom")] HRESULT ValidFrom([out,
retval] BSTR* pVal);
  [propget, id(8), helpstring("property ValidTo")] HRESULT ValidTo([out, retval]
BSTR* pVal);
  [id(9), helpstring("method Save")] HRESULT Save([in] BSTR path, [out, retval]
VARIANT_BOOL* result);
  [id(10), helpstring("method Send")] HRESULT Send([in] IGohubConnection*
connection, [out,retval] VARIANT_BOOL* result);
  [id(11), helpstring("method Close")] HRESULT Close(void);
  [propget, id(12), helpstring("property SmgsTypeCode")] HRESULT
SmgsTypeCode([out, retval] BSTR* pVal);
  [id(13), helpstring("method SaveUserData")] HRESULT SaveUserData([in] BSTR
path, [out,retval] VARIANT_BOOL* result);
};
[
```

```
	object,
	uuid(685B21FA-43A7-4ACC-9A57-AB7AC332942C),
	dual,
	nonextensible,
	helpstring("IGohubEData Interface"),
	pointer_default(unique)
]
interface IGohubEData : IDispatch{
  [propget, id(1), helpstring("property Id")] HRESULT Id([out, retval] BSTR*
pVal);
  [propget, id(2), helpstring("property Revision")] HRESULT Revision([out,
retval] ULONGLONG* pVal);
  [propget, id(3), helpstring("property RevisionDate")] HRESULT
RevisionDate([out, retval] BSTR* pVal);
  [propget, id(4), helpstring("property Version")] HRESULT Version([out, retval]
BSTR* pVal);
  [propget, id(5), helpstring("property DocType")] HRESULT DocType([out, retval]
UINT* pVal);
  [propget, id(6), helpstring("property Status")] HRESULT Status([out, retval]
INT* pVal);
  [propget, id(7), helpstring("property AttachmentId")] HRESULT
AttachmentId([out, retval] BSTR* pVal);
  [id(8), helpstring("method Save")] HRESULT Save([in] BSTR path, [out, retval]
VARIANT_BOOL* result);
  [id(9), helpstring("method Send")] HRESULT Send([in] IGohubConnection*
connection, [out,retval] VARIANT_BOOL* result);
  [id(10), helpstring("method LoadData")] HRESULT LoadData([in] BSTR path, [out,
retval] VARIANT_BOOL* result);
  [id(11), helpstring("method Update")] HRESULT Update([in] IGohubConnection*
connection, [out,retval] VARIANT_BOOL* result);
  [id(12), helpstring("method Close")] HRESULT Close(void);
};
[
	object,
	uuid(62C21013-07D6-4d9d-83C4-9FA6E770B2EA),
	dual,
	nonextensible,
	helpstring("IGohubPiPackage Interface"),
	pointer_default(unique)
]
interface IGohubPiPackage : IDispatch{
  [propget, id(1), helpstring("property Id")] HRESULT Id([out, retval] BSTR*
pVal);
  [propget, id(2), helpstring("property Revision")] HRESULT Revision([out,
retval] ULONGLONG* pVal);
  [propget, id(3), helpstring("property RevisionDate")] HRESULT
RevisionDate([out, retval] BSTR* pVal);
  [propget, id(4), helpstring("property ConsignmentId")] HRESULT
ConsignmentId([out, retval] BSTR* pVal);
  [propget, id(5), helpstring("property Status")] HRESULT Status([out, retval]
INT* pVal);
  [propget, id(6), helpstring("property PiPackageToEDataCount")] HRESULT
PiPackageToEDataCount([out, retval] INT* pVal);
  [id(7), helpstring("method PiPackageToEData")] HRESULT PiPackageToEData([in]
INT index, [out, retval] IGohubPiPackageToEData** result);
  [id(8), helpstring("method Save")] HRESULT Save([in] BSTR path, [out, retval]
VARIANT_BOOL* result);
  [id(10), helpstring("method Close")] HRESULT Close(void);
};
[
	object,
	uuid(88AAFC89-0426-4551-BD1C-F57F63D0C335),
	dual,
	nonextensible,
	helpstring("IGohubPiPackageToEData Interface"),
```

```
	pointer_default(unique)
]
interface IGohubPiPackageToEData : IDispatch{
  [propget, id(1), helpstring("property Id")] HRESULT Id([out, retval] BSTR*
pVal);
  [propget, id(2), helpstring("property EDataId")] HRESULT EDataId([out, retval]
BSTR* pVal);
  [propget, id(3), helpstring("property PiPackageId")] HRESULT PiPackageId([out,
retval] BSTR* pVal);
  [propget, id(4), helpstring("property Note")] HRESULT Note([out, retval] BSTR*
pVal);
  [propget, id(5), helpstring("property Status")] HRESULT Status([out, retval]
INT* pVal);
  [propget, id(6), helpstring("property EDataVersion")] HRESULT
EDataVersion([out, retval] BSTR* pVal);
};
[
	object,
	uuid(EB03BE7D-48B7-4AFD-8A94-A5012D844A17),
	dual,
	nonextensible,
	helpstring("IGohubFdu92 Interface"),
	pointer_default(unique)
]
interface IGohubFdu92 : IDispatch{
	[propget, id(1), helpstring("property Id")] HRESULT Id([out, retval] BSTR*
pVal);
	[propget, id(2), helpstring("property Status")] HRESULT Status([out,
retval] LONG* pVal);
	[propget, id(3), helpstring("property Revision")] HRESULT Revision([out,
retval] LONG* pVal);
	[propget, id(4), helpstring("property Text")] HRESULT Text([out, retval]
BSTR* pVal);
	[id(5), helpstring("method Save")] HRESULT Save([in] BSTR path, [in] LONG
codePage, [out,retval] VARIANT_BOOL* result);
	[id(6), helpstring("method Close")] HRESULT Close(void);
	[id(7), helpstring("method Sign")] HRESULT Sign([in] IGohubConnection*
connection, [out,retval] VARIANT_BOOL* result);
	[id(8), helpstring("method Send")] HRESULT Send([in] IGohubConnection*
connection, [out,retval] VARIANT_BOOL* result);
	[propget, id(9), helpstring("property HasSignature")] HRESULT
HasSignature([out, retval] VARIANT_BOOL* result);
	[propget, id(10), helpstring("property Signer")] HRESULT Signer([out,
retval] IGohubSignerInfo** pVal);
	[propget, id(11), helpstring("property SignTime")] HRESULT SignTime([out,
retval] BSTR* pVal);
};
[
	object,
	uuid(78DCD121-84B7-4D41-B15C-F9F7677A3519),
	dual,
	nonextensible,
	helpstring("IGohubGu46 Interface"),
	pointer_default(unique)
]
interface IGohubGu46 : IDispatch{
	[propget, id(1), helpstring("property Id")] HRESULT Id([out, retval] BSTR*
pVal);
	[propget, id(2), helpstring("property Status")] HRESULT Status([out,
retval] LONG* pVal);
	[propget, id(3), helpstring("property Revision")] HRESULT Revision([out,
retval] LONG* pVal);
	[propget, id(4), helpstring("property Text")] HRESULT Text([out, retval]
BSTR* pVal);
```

```
	[id(5), helpstring("method Save")] HRESULT Save([in] BSTR path, [in] LONG
codePage, [out,retval] VARIANT_BOOL* result);
	[id(6), helpstring("method Close")] HRESULT Close(void);
	[id(7), helpstring("method Sign")] HRESULT Sign([in] IGohubConnection*
connection, [out,retval] VARIANT_BOOL* result);
	[id(8), helpstring("method Send")] HRESULT Send([in] IGohubConnection*
connection, [out,retval] VARIANT_BOOL* result);
	[propget, id(9), helpstring("property HasSignature")] HRESULT
HasSignature([out, retval] VARIANT_BOOL* result);
	[propget, id(10), helpstring("property Signer")] HRESULT Signer([out,
retval] IGohubSignerInfo** pVal);
	[propget, id(11), helpstring("property SignTime")] HRESULT SignTime([out,
retval] BSTR* pVal);
};
[
	object,
	uuid(C2F92D3C-295A-4453-B4CF-B33D2C8966E3),
	dual,
	nonextensible,
	helpstring("IGohubGu45 Interface"),
	pointer_default(unique)
]
interface IGohubGu45 : IDispatch{
	[propget, id(1), helpstring("property Id")] HRESULT Id([out, retval] BSTR*
pVal);
	[propget, id(2), helpstring("property Status")] HRESULT Status([out,
retval] LONG* pVal);
	[propget, id(3), helpstring("property Revision")] HRESULT Revision([out,
retval] LONG* pVal);
	[propget, id(4), helpstring("property Text")] HRESULT Text([out, retval]
BSTR* pVal);
	[id(5), helpstring("method Save")] HRESULT Save([in] BSTR path, [in] LONG
codePage, [out,retval] VARIANT_BOOL* result);
	[id(6), helpstring("method Close")] HRESULT Close(void);
	[propget, id(7), helpstring("property HasSignature")] HRESULT
HasSignature([out, retval] VARIANT_BOOL* result);
	[propget, id(8), helpstring("property Signer")] HRESULT Signer([out,
retval] IGohubSignerInfo** pVal);
	[propget, id(9), helpstring("property SignTime")] HRESULT SignTime([out,
retval] BSTR* pVal);
};
[
	object,
	uuid(4B840AC2-7128-4989-A56C-2570E8435C48),
	dual,
	nonextensible,
	helpstring("IGohubGu27 Interface"),
	pointer_default(unique)
]
interface IGohubGu27 : IDispatch{
	[propget, id(1), helpstring("property Id")] HRESULT Id([out, retval] BSTR*
pVal);
	[propget, id(2), helpstring("property Text")] HRESULT Text([out, retval]
BSTR* pVal);
	[propget, id(3), helpstring("property Revision")] HRESULT Revision([out,
retval] LONG* pVal);
	[propget, id(4), helpstring("property HasSignature")] HRESULT
HasSignature([out, retval] VARIANT_BOOL* pVal);
	[propget, id(5), helpstring("property Signer")] HRESULT Signer([out,
retval] IGohubSignerInfo** pVal);
	[propget, id(6), helpstring("property SignTime")] HRESULT SignTime([out,
retval] BSTR* pVal);
	[id(7), helpstring("method Save")] HRESULT Save([in] BSTR path, [in] LONG
codePage, [out,retval] VARIANT_BOOL* result);
```

```
	[id(8), helpstring("method Sign")] HRESULT Sign([in] IGohubConnection*
connection, [out,retval] VARIANT_BOOL* result);
	[id(9), helpstring("method Send")] HRESULT Send([in] IGohubConnection*
connection, [out,retval] VARIANT_BOOL* result);
	[id(10), helpstring("method Close")] HRESULT Close(void);
	[propget, id(11), helpstring("property Status")] HRESULT Status([out,
retval] LONG* pVal);
	[id(12), helpstring("method SendReceived")] HRESULT SendReceived([in]
IGohubConnection* connection, [in] BSTR gu27Id, [out,retval] VARIANT_BOOL*
result);
	[id(13), helpstring("method SaveData")] HRESULT SaveData([in] BSTR path,
[in] LONG codePage, [in] LONG gu27Version, [out,retval] VARIANT_BOOL* result);
	[id(14), helpstring("method DataText")] HRESULT DataText([in] LONG
gu27Version, [out,retval] BSTR* pVal);
};
[
	object,
	uuid(0af49642-b12d-4fdb-b363-a7497cc78462),
	dual,
	nonextensible,
	helpstring("IInformServicesDocument Interface"),
	pointer_default(unique)
]
interface IGohubInformServicesDocument : IDispatch{
	[id(1), helpstring("method Save")] HRESULT Save([in] BSTR path,
[out,retval] VARIANT_BOOL* result);
	[id(2), helpstring("method Close")] HRESULT Close(void);
	[propget, id(3), helpstring("property Id")] HRESULT Id([out, retval]
ULONGLONG* pVal);
	[propget, id(4), helpstring("property Revision")] HRESULT Revision([out,
retval] ULONGLONG* pVal);
	[propget, id(5), helpstring("property FileName")] HRESULT FileName([out,
retval] BSTR* pVal);
	[propget, id(6), helpstring("property Comment")] HRESULT Comment([out,
retval] BSTR* pVal);
	[propget, id(7), helpstring("property CreatedDate")] HRESULT
CreatedDate([out, retval] BSTR* pVal);
	[propget, id(8), helpstring("property DocDate")] HRESULT DocDate([out,
retval] BSTR* pVal);
	[id(9), helpstring("method SaveXml")] HRESULT SaveXml([in] BSTR path,
[out,retval] VARIANT_BOOL* result);
	[propget, id(10), helpstring("property IsEmpty")] HRESULT IsEmpty([out,
retval] VARIANT_BOOL* result);
};
[
	object,
	uuid(0669AB3B-8E1B-4161-8A2E-244D5A62029A),
	dual,
	nonextensible,
	helpstring("IDispatchInfo Interface"),
	pointer_default(unique)
]
interface IGohubDispatchInfo : IDispatch{
	[id(1), helpstring("method Close")] HRESULT Close(void);
	[id(2), helpstring("method WagOwnerByIndex")] HRESULT WagOwnerByIndex([in]
ULONGLONG index, [out,retval] BSTR* pVal);
	[id(3), helpstring("method TypeByIndex")] HRESULT TypeByIndex([in]
ULONGLONG index, [out,retval] BSTR* pVal);
	[id(4), helpstring("method DateByIndex")] HRESULT DateByIndex([in]
ULONGLONG index, [out,retval] BSTR* pVal);
	[id(5), helpstring("method DescriptionByIndex")] HRESULT
DescriptionByIndex([in] ULONGLONG index, [out,retval] BSTR* pVal);
	[id(6), helpstring("method NumberByIndex")] HRESULT NumberByIndex([in]
ULONGLONG index, [out,retval] BSTR* pVal);
```

```
	[id(7), helpstring("method IsEmptyByIndex")] HRESULT IsEmptyByIndex([in]
ULONGLONG index, [out,retval] BSTR* pVal);
	[id(8), helpstring("method Count")] HRESULT Count([out,retval] LONG*
count);
};
[
	object,
	uuid(B8FFED27-656B-48C0-AD6F-8EBE09F8F8E8),
	dual,
	nonextensible,
	helpstring("IPSTDInfo Interface"),
	pointer_default(unique)
]
interface IGohubPSTDInfo : IDispatch{
	[id(1), helpstring("method Close")] HRESULT Close(void);
	[id(2), helpstring("method ArrivalDateByIndex")] HRESULT
ArrivalDateByIndex([in] ULONGLONG index, [out,retval] BSTR* pVal);
	[id(3), helpstring("method DepartureByIndex")] HRESULT
DepartureByIndex([in] ULONGLONG index, [out,retval] BSTR* pVal);
	[id(4), helpstring("method NumberByIndex")] HRESULT NumberByIndex([in]
ULONGLONG index, [out,retval] BSTR* pVal);
	[id(5), helpstring("method ReqDateByIndex")] HRESULT ReqDateByIndex([in]
ULONGLONG index, [out,retval] BSTR* pVal);
	[id(6), helpstring("method Count")] HRESULT Count([out,retval] LONG*
count);
};

[
	uuid(EB746ACA-C81E-42EE-B1C4-C435A8CD0082),
	version(1.0),
	helpstring("gohubclientcom 1.0 Type Library")
]
library gohubclientcomLib
{
	importlib("stdole2.tlb");
	[
		uuid(B1166D0A-F0FB-4526-803C-1D37F8EB5100),
		helpstring("GohubDocument Class")
	]
	coclass GohubComDocument
	{
		[default] interface IGohubDocument;
	};
	[
		uuid(77E0C042-770C-4EA1-9593-1E370E5E8EE3),
		helpstring("GohubConnection Class")
	]
	coclass GohubComConnection
	{
		[default] interface IGohubConnection;
	};
	[
		uuid(CF908D67-DAF6-43B0-9621-1DD417CFF3D7),
		helpstring("GohubClient Class")
	]
	coclass GohubComClient
	{
		[default] interface IGohubClient;
	};
	[
		uuid(D3160E70-00DC-4502-A84B-840D6929D99A),
		helpstring("GohubError Class")
	]
	coclass GohubComError
	{
```

```
        [default] interface IGohubError;
};
[
        uuid(A86B2DEA-CFC4-48F5-B904-73D638E73F95),
        helpstring("GohubSignerInfo Class")
]
coclass GohubComSignerInfo
{
        [default] interface IGohubSignerInfo;s
};
[
        uuid(C1B3AEDC-6673-470D-AFF0-48A0659A9BCD),
        helpstring("GohubAttachment Class")
]
coclass GohubComAttachment
{
        [default] interface IGohubAttachment;
};
[
        uuid(934AD530-23EA-481E-A365-CD37EC536474),
        helpstring("GohubComEData Class")
]
coclass GohubComEData
{
        [default] interface IGohubEData;
};
[
        uuid(72819B78-CC37-4bc5-A80F-E466E1D19AEB),
        helpstring("GohubComPiPackage Class")
]
coclass GohubComPiPackage
{
        [default] interface IGohubPiPackage;
};
[
        uuid(B5A1FD2F-3F4E-4acb-B884-A9AC65850DA2),
        helpstring("GohubComPiPackageToEData Class")
]
coclass GohubComPiPackageToEData
{
        [default] interface IGohubPiPackageToEData;
};
[
        uuid(5848DA64-79D3-41C1-B60C-C88D92508B11),
        helpstring("GohubComFdu92 Class")
]
coclass GohubComFdu92
{
        [default] interface IGohubFdu92;
};
[
        uuid(C2878FA7-4528-4C73-9FF2-D0AA15A91895),
        helpstring("GohubComGu46 Class")
]
coclass GohubComGu46
{
        [default] interface IGohubGu46;
};
[
        uuid(33BBBEE8-8FC6-42BD-9213-A0071F0791EC),
        helpstring("GohubComGu45 Class")
]
coclass GohubComGu45
{
        [default] interface IGohubGu45;
```

```
	};
	[
		uuid(BF5FCE75-E9A7-4282-BFB3-7B8AAB339098),
		helpstring("GohubComGu27 Class")
	]
	coclass GohubComGu27
	{
		[default] interface IGohubGu27;
	};
	[
		uuid(f7da867b-d7fa-4831-8c81-b8d450d34229),
		helpstring("GohubComInfServsDoc Class")
	]
	coclass GohubComInfServsDoc
	{
		[default] interface IGohubInformServicesDocument;
	};
	[
		uuid(C5C644CF-8353-470f-AFA9-A12F8396EF87),
		helpstring("GohubComDispatchInfo Class")
	]
	coclass GohubComDispatchInfo
	{
		[default] interface IGohubDispatchInfo;
	};
	[
		uuid(5f323e6c-3d6f-42b7-84d1-60fb25fe2dde),
		helpstring("GohubComPSTDInfo Class")
	]
	coclass GohubComPSTDInfo
	{
		[default] interface IGohubPSTDInfo;
	};


};
```