

Система лицензирования и защиты конфигураций платформы 1С:Предприятие 8, версия 3.0

Руководство разработчика

Общие положения	1
Подключение	2
Методы менеджера объектов	3
Получение лицензии	5
Создание защищенных объектов	6
Ограничение функционала конфигурации на основе параметров ключей защиты	8
Получение / использование лицензионных параметров	9
Получение информации о сервере СЛК	10
Получение системных логов	11
Приложение: Классы исключений	11
Приложение: Свойства и методы объекта компоненты	13
Приложение: Методы менеджера объектов	15
Приложение: Демонстрационная конфигурация	29
Приложение: Создание файлов данных	31
Приложение: Расположение конфигурационных файлов	34
Приложение: Расположение системных логов	34

Общие положения

СЛК реализует контроль лицензий по сеансам информационной базы, при этом при работе в клиент-серверном режиме (сервер приложений 1С, веб сервер) каждый сеанс будет занимать отдельную лицензию.

При работе в файловом режиме все подключения с одного компьютера будут занимать одну лицензию.

Также система позволят контролировать только наличие ключа без учета количества лицензий или, наоборот, получать лицензию только для определенных параметров ключа, например, ИНН/КПП или версии продукта (см. [Получение лицензии и параметры контроля](#)).

С целью защиты оригинального авторского кода и экранных форм реализован механизм защиты обработок и отчетов (защищенные объекты). Защищенные объекты исключены из конфигурации и помещены в специальный файл данных, подготовленный разработчиком при помощи редактора файлов данных (см. [Создание файлов данных](#)).

Файлы защищенных объектов рекомендуется включать в конфигурацию в виде общих макетов, но для совместимости с предыдущими версиями СЛК возможно размещать их на сервере СЛК. Также возможно размещение файлов данных в произвольных макетах, например, обработок или отчетов.

По использованию защищенных объектов см. [Создание защищенных объектов](#).

Подключение

Работа с СЛК в конфигурации реализуется при помощи внешней компоненты СЛК.

Упрощенно, разработчик должен выполнить следующую последовательность действий:

1. Подключить компоненту СЛК и создать экспортируемый ею объект
2. Указать в свойствах созданного объекта параметры связи с сервером СЛК
3. Выполнить запуск объекта, указав используемую серию ключей
4. Создать менеджер объектов

Затем при помощи менеджера объектов можно получать лицензию, создавать защищенные объекты и т.п.

```
Процедура ПодключитьКомпоненту() Экспорт
    // Подключение компоненты из общего макета, в который должен быть
    // загружен zip-архив с модулями компоненты для всех поддерживаемых
    // операционных систем.
    //
    // Файл архива с компонентой в поставке имеет вид licenceaddin-{%version%}-template.zip

    // Обязательно указываем тип компоненты ТипВнешнейКомпоненты.Native
    Если НЕ ПодключитьВнешнююКомпоненту("ОбщийМакет.КомпонентаСЛК", "Licence",
ТипВнешнейКомпоненты.Native) Тогда
        СисИнфо = Новый СистемнаяИнформация;
        ВызватьИсключение "Ошибка подключения компоненты СЛК " + СисИнфо.ТипПлатформы;
    КонецЕсли;
КонецПроцедуры
```

```
Функция ОбъектКомпоненты() Экспорт
    // Подключаем компоненту только в случае необходимости (см. блок Исключение)
    Попытка
        // Считаем, что компонента уже подключена
        Возврат Новый("AddIn.Licence.LicenceExtension");
    Исключение
        // Исключение возможно только если компонента еще не подключена
    КонецПопытки;
    ПодключитьКомпоненту();
    Возврат Новый("AddIn.Licence.LicenceExtension");
КонецФункции
```

```
Функция ОбъектКомпоненты() Экспорт
    // Получение объекта компоненты
    Объект = ОбъектКомпоненты();

    // Установка параметров связи с сервером СЛК
    Объект.ПараметрыСвязи = ПолучитьПараметрыСвязи();

    Если НЕ Объект.Запуск(СЛК.Серия()) Тогда
        ВызватьИсключение Объект.ПолучитьОписаниеОшибки();
    КонецЕсли;
```

```

// Получаем имя менеджера объектов
Имя = Объект.ПолучитьМенеджерОбъектов();
Если Имя = Неопределено Тогда
    ВызватьИсключение Объект.ПолучитьОписаниеОшибки();
КонецЕсли;

// Отключаем предупреждения о загрузке обработок из непроверенных источников
Защита = Неопределено;
Попытка
    Защита = Новый ("ОписаниеЗащитыОтОпасныхДействий");
    Защита.ПредупреждатьОбОпасныхДействиях = Ложь;
Исключение
    // Исключение возможно на предыдущих версиях платформы без механизма
    // защиты от опасных действий
КонецПопытки;

// Создание менеджера объектов
Попытка
    Если Защита = Неопределено Тогда
        МенеджерОбъектов = ВнешниеОбработки.Создать(Имя, БезопасныйРежим());
    Иначе
        МенеджерОбъектов = ВнешниеОбработки.Создать(
            ВнешниеОбработки.Подключить(ПоместитьВоВременноеХранилище(
                Новый ДвоичныеДанные(Имя)), , БезопасныйРежим(), Защита));
    КонецЕсли;
    // Настраиваем менеджер объектов
    МенеджерОбъектов.УстановитьОбъектКомпоненты(Объект, Защита);
Исключение
    // Логгируем и получаем описание ошибки
    ВызватьИсключение Объект.ОшибкаСозданияОбъекта(ОписаниеОшибки());
КонецПопытки;
Возврат МенеджерОбъектов;
КонецФункции

```

Важно: Для пользователя конфигурации должна быть определена роль, у которой установлено право «Активные пользователи».

Важно: Подключение компоненты и работа с СЛК должны выполняться только в серверном контексте, что позволит избежать компоненты установки клиентским приложением на локальный компьютер (тонкий и веб клиенты).

Подключение компоненты, создание экспортируемого объекта и менеджера объектов рекомендуется поместить в отдельный серверный модуль.

Для повышения производительности затратные по времени операции (подключение компоненты, создание защищенных объектов и менеджера объектов) необходимо вынести в дополнительный серверный модуль, у которого установлен режим повторно возвращаемых значений «На время сеанса».

См. также описание методов и свойств объекта компоненты ([ПараметрыСвязи](#), [Запуск](#)) а также демонстрационную конфигурацию, общие модули [СЛК](#) и [СЛКПовтИсп.](#)

Методы менеджера объектов

Методы менеджера объектов делятся на две группы: методы, вызывающие и не вызывающие исключения. Методы, вызывающие исключения, в случае ошибки вызывают исключения и, следовательно, должны использоваться в конструкции обработки исключений:

```
Попытка
    Лицензия = МенеджерОбъектов().ПолучитьЛицензию();
Исключение
    // Обработка ошибки на основе встроенной функции ОписаниеОшибки()
КонецПопытки
```

Методы, не вызывающие исключения, реализованы в виде функций, название которых начинается со слова **Попытка** и возвращающие в случае ошибки **Ложь** или **Неопределено**. При их использовании для получения описания ошибки необходимо вызвать функцию менеджера объектов **ПолучитьОписаниеОшибки**:

```
Лицензия = Менеджер.ПопыткаПолучитьЛицензию();
Если Лицензия = Неопределено Тогда
    // Обработка ошибки на основе функции Менеджер.ПолучитьОписаниеОшибки()
КонецЕсли
```

Использование тех или иных методов определяется предпочтениями разработчика.

Сообщения об ошибках начинаются с имени класса исключения в круглых скобках, например:

(EExecuteError) Ошибка при соединении с "SERVER1:9099" (Error: Connection problems, no response received.)

Разработчик конфигурации может использовать класс исключения для реализации дополнительной логики при обработке ошибок, например, при ошибках связи открывать окно настроек параметров и т.п.

```
// ОписаниеОшибки - описание ошибки, полученное встроенной
// функцией ОписаниеОшибки() или функцией менеджера объектов
// ПолучитьОписаниеОшибки()
КлассИсключения = СЛКОбщие.ПолучитьКлассИсключения(ОписаниеОшибки);
// Если сетевая ошибка, открываем форму настроек параметров связи
Если КлассИсключения = "EExecuteError" Или КлассИсключения = "ELocalServerNotFound" Тогда
    ОткрытьФорму("Обработка.СЛК.Форма.ПараметрыСвязи");
КонецЕсли;
```

Пример извлечения класса исключения см. в функции **ПолучитьКлассИсключения** общего модуля **СЛКОбщие** в демонстрационной конфигурации.

Полный список классов исключений см. в **Приложении 1: Классы исключений**.

Получение лицензии

Для получения лицензии используются методы менеджера объектов [ПолучитьЛицензию](#) / [ПопыткаПолучитьЛицензию](#), которые возвращают фиксированную структуру параметров ключа защиты, лицензия которого была занята (подробнее см. описание метода [ПолучитьЛицензию](#)).

По умолчанию сервер СЛК возвращает лицензию любого свободного ключа используемой серии, однако, разработчик может предусмотреть дополнительные условия выбора ключей. При этом лицензии будут занимать только у ключей с соответствующими значениями параметров, остальные ключи будут игнорироваться.

См. ниже [Использование параметров ключа](#), [Проверка только наличия ключа](#).

Использование кода доступа

При помощи метода [УстановитьКодДоступа](#) разработчик может указать системе получать лицензию (а также список лицензий и счетчики) только тех ключей, для которых в консоли сервера СЛК установлен соответствующий код доступа.

```
// Доступ по коду, установленному в консоли
МенеджерОбъектов.УстановитьКодДоступа(КодДоступа);
```

Данный метод отбора ключей может применяться, когда необходимо разделить доступ разных клиентов / пользователей к ключам защиты. Например, при работе конфигурации в разделенном режиме или в случае необходимости раздачи сервером СЛК лицензий через Интернет.

Использование параметров ключа

При помощи метода [УстановитьПараметрыКонтроляЛицензий](#) разработчик может указать системе получать только лицензии ключей с соответствующими ИНН, КПП, и Версией продукта.

Например, при указании ИНН/КПП конфигурация получит лицензию, только если на сервере СЛК установлены ключи с указанными значениями ИНН/КПП. Аналогично, при указании версии продукта, получение лицензии будет возможно, только если установлены ключи с соответствующим значением этого параметра.

```
// Контроль только наличие ключа
МенеджерОбъектов.УстановитьПараметрыКонтроляЛицензий(Истина);
```

```
// Контроль по ИНН/КПП – при указании этих параметров конфигурация
// получит лицензию только если на сервере СЛК будут свободные ключ с
// соответствующими значениями ИНН/КПП.
МенеджерОбъектов.УстановитьПараметрыКонтроляЛицензии(, 77000000, 77000);

// Контроль только наличия ключа с определенными ИНН/КПП
МенеджерОбъектов.УстановитьПараметрыКонтроляЛицензии(Истина, 77000000, 77000);

// Контроль определенной версии продукта
МенеджерОбъектов.УстановитьПараметрыКонтроляЛицензии(,,, 2);

// Установка параметров по умолчанию
МенеджерОбъектов.УстановитьПараметрыКонтроляЛицензии();
```

Вызов этого метода должен выполняться в защищенном коде, например, в предопределенном методе защищенного объекта [ПриСозданииОбъекта](#) (см. пример ниже при описании предопределенных методов защищенных объектов).

Из соображений безопасности повторная установка параметров возможна только с теми же значениями, которые были использованы при первом вызове. В случае повторного вызова с другими значениями дальнейшая работа менеджера объектов блокируется.

Проверка только наличия ключа

Для контроля только наличия ключа защиты без учета количества лицензий необходимо использовать метод [УстановитьПараметрыКонтроляЛицензий](#), передавая в первом параметре значение **Истина**:

```
// Контроль только наличие ключа
МенеджерОбъектов.УстановитьПараметрыКонтроляЛицензии(Истина);
```

После установки этого значения последующие вызовы методов [ПолучитьЛицензию](#) / [ПопыткаПолучитьЛицензию](#) не будут занимать лицензию.

Из соображений безопасности такой вызов должен быть помещен в предопределенной процедуре [ПриСозданииОбъекта](#) всех защищенных объектов. В противном случае, при создании защищенного объекта режим контроля только наличия ключа будет отменен.

Создание защищенных объектов

Для создания защищенных объектов используются методы менеджера объектов [СоздатьОбъект](#) / [ПопыткаСоздатьОбъект](#), в которые передается имя объекта в файле данных.

Файл данных создается разработчиком конфигурации при помощи редактора файлов данных и размещается или в макетах конфигурации, или в папке исполняемого модуля сервера СЛК (см. также [Создание файлов данных](#)).

Рекомендуется загружать файл данных в общий макет конфигурации. В этом случае перед именем объекта необходимо указать имя макета, например:

```
// Создание объекта "ЗащищеннаяОбработка" из файла данных,
// загруженного в общий макет "ОбъектыСЛК"
Объект = МенеджерОбъектов().СоздатьОбъект("ОбъектыСЛК.ЗащищеннаяОбработка");

// Создание объекта из файла XXXX.datafile на сервере СЛК (XXXX – серия ключей)
Объект = МенеджерОбъектов().СоздатьОбъект("ЗащищенныйОбъект");
```

Кроме этого, возможно размещение файлов данных в произвольных макетах (например, обработок или отчетов). В этом случае необходимо принудительно загрузить и зарегистрировать файл данных:

```
// Загрузка файла из макета обработки "Обработка.ОбъектыСЛК"
// и регистрация под именем "ВнешниеОбъекты"
Если НЕ Менеджер.МакетЗагружен(ИмяМакета) Тогда
    МенеджерОбъектов().ЗагрузитьИзМакета(Обработка.ОбъектыСЛК, "ВнешниеОбъекты");
КонецЕсли;
// Создание объекта из загруженного файла "ВнешниеОбъекты"
Объект = МенеджерОбъектов().СоздатьОбъект("ВнешниеОбъекты.ЗащищенныйОбъект");
```

Подробнее см. демонстрационную конфигурацию, форму [ПроверкаКоманд](#) обработки СЛК.

Предопределенные методы защищенных объектов

При создании объекта функцией вызываются предопределенные экспортные процедуры [ПередКонтролемЛицензии](#) и [ПриСозданииОбъекта](#), которые должны быть реализованы в модуле объекта. В этих методах разработчик может реализовать собственную логику, например, установку параметров контроля лицензий или запрет создания объекта в зависимости от значений лицензионных параметров / памяти ключей защиты.

```
Процедура ПриСозданииОбъекта(МенеджерОбъектов, Отказ, ОписаниеОшибки) Экспорт
    // Сохраняем менеджер объектов во внутренней переменной модуля
    _менеджерОбъектов = МенеджерОбъектов;

    // Пример контроля по ИНН/КПП – при указании этих параметров конфигурация
    // получит лицензию только если на сервере СЛК будут свободные ключ с
    // соответствующими значениями ИНН/КПП.
    _менеджерОбъектов.УстановитьПараметрыКонтроляЛицензии(, 77000000, 77000);

    // Пример проверки только наличия ключа
    // _менеджерОбъектов.УстановитьПараметрыКонтроляЛицензии(Истина);

КонецПроцедуры
```

Доступ к формам защищенных объектов

Для доступа к формам защищенных объектов необходимо создать защищенный объект и передать в клиентский контекст имя нужной формы:

```
// Получение полного имени объекта для доступа к формам
&НаСервере
Функция ВспомПолноеИмяОбъекта (Имя)
    Объект = СЛК.МенеджерОбъектов().СоздатьОбъект(Имя);
    Возврат Объект.Метаданные().ПолноеИмя();
КонецФункции

&НаКлиенте
Процедура ОткрытьФормуОбъекта (Команда)
    Попытка
        // Получаем имя формы по умолчанию
        ЛокИмяФормы = ВспомПолноеИмяОбъекта(ИмяОбъекта) + ".Форма";
        ПолучитьФорму(ЛокИмяФормы).Открыть();
    Исключение
        Сообщить(ОписаниеОшибки());
    КонецПопытки;
КонецПроцедуры
```

Ограничение функционала конфигурации на основе параметров ключей защиты

Система позволяет разработчику реализовывать различные схемы ограничения пользовательского функционала – как на основе параметров (свойств) ключей защиты, так и при помощи лицензионных параметров, как в предыдущих версиях (см. [ниже](#)).

Ограничение / управление функционалом конфигурации на основе параметров ключей защиты может быть использовано в решениях, выпускаемых / издаваемых фирмой 1С на основе значений полей, устанавливаемых при создании ключей.

Например, версии продукта (ProductVersion), флагов (Flags) функциональности или рег. номера (RegNo) / артикула (Article).

Кроме этого, для ключей решений, размещаемых на облачных сервисах фирмы 1С, устанавливаются специальные значения поля тип сервиса (ServiceType), подробнее см. описание структуры лицензии в команде [ПолучитьЛицензию](#).

Пример использования версии продукта

Например, конфигурация использует **числовое поле версия продукта (ProductVersion)** для разделения функционала различных редакций.

Разработчик получает информацию об установленных ключах (см. команды [ПолучитьЛицензию](#) / [ПолучитьЛицензии](#) и описание структуры лицензии) и на основе значения версии продукта ограничивает / разрешает определенный

функционал.

Например, какие-то документы или отчеты будут доступны только при наличии ключей с версией продукта больше или равной **1**, другие – больше или равной **2** и т.п.

Принципиально, этот параметр может быть впоследствии изменен и сервер СЛК при очередном обмене данными с Центром Лицензирования обновит информацию у конечного клиента.

Соответственно, если версия продукта изменится, то для пользователя произойдет прозрачный переход с одной редакции на другую.

Очевидно, получение и проверку значений полей, на основе которых выполняется ограничение функционала, необходимо выполнять в защищенном коде.

Получение / использование лицензионных параметров

Аналогично предыдущим версиям СЛК 1.1 / 2.0 / 2.1 для ограничения пользовательского функционала система представляет возможность работы с лицензионными параметрами, привязанными к конкретным ключам защиты. Также для каждого параметра можно указать значение по умолчанию, которое будет использоваться в случае, если для конкретного ключа значение не определено. В зависимости от значений параметров разработчик может разрешать или запрещать пользователю те или иные действия.

Лицензионные параметры хранятся в файле, создаваемым разработчиком (подробнее см. **Создание файлов данных**). Этот файл, аналогично файлу с защищенными объектами, может размещен или в макете конфигурации, или в папке исполняемого модуля сервера СЛК.

Для получения значения параметра используются методы менеджера объектов `ПолучитьПараметр` / `ПопыткаПолучитьПараметр`, в которые передается серийный номер ключа защиты, для которого необходимо получить значение, и имя параметра в файле данных.

Серийный номер ключа можно получить при помощи команд `ПолучитьЛицензию` / `ПопыткаПолучитьЛицензию` или `ПолучитьЛицензии` / `ПопыткаПолучитьЛицензии`.

При размещении файла параметров в общем макете перед именем параметра необходимо указать имя макета, например:

```
// Получение лицензии и С/Н ключа
Лицензия = МенеджерОбъектов.ПолучитьЛицензию();
// Получение значения параметра "ТипПриложения" из файла параметров,
// загруженного в общий макет "ПараметрыСЛК"
Значение = МенеджерОбъектов.ПолучитьПараметр(Лицензия.KeyNo, "ПараметрыСЛК.ТипПриложения");

// Получение значения из файла XXXX.paramfile на сервере СЛК (XXXX – серия ключей)
Значение = МенеджерОбъектов.ПолучитьПараметр(Лицензия.KeyNo, "ТипПриложения");
```

Кроме этого, возможно размещение файлов данных в произвольных макетах (например, обработок или отчетов). В этом случае необходимо принудительно загрузить и зарегистрировать файл данных:

```
// Загрузка параметров из макета обработки "Обработка.ВнешниеПараметры"
// и регистрация под именем "ВнешниеПараметры". В последний параметре указывается тип
// файла в макете:
// 0 – защищенные объекты (по умолчанию)
// 1 – параметры
МенеджерОбъектов.ЗагрузитьИзМакета(Обработка.ОбъектыСЛК, "ВнешниеПараметры", , 1);
// Получение значения
Значение = МенеджерОбъектов.ПолучитьПараметр(Лицензия.KeyNo, "ВнешниеПараметры.ТипПриложения");
```

Подробнее см. демонстрационную конфигурацию, форму [ПроверкаКоманд](#) обработки СЛК.

Получение информации о сервере СЛК

В СЛК 3.0 реализована возможность получения в конфигурации подробной информации о сервере СЛК при помощи методов менеджера объектов

[ПолучитьИнформациюОСервере](#) / [ПопыткаПолучитьИнформациюОСервере](#), которые возвращают фиксированную структуру параметров сервера СЛК, с которым установлено соединение (подробнее см. [ПолучитьИнформациюОСервере](#)):

```
Менеджер = МенеджерОбъектов();
// Получение информации о сервере
Инфо = Менеджер.ПолучитьИнформациюОСервере();
Возврат Инфо.Version + " (" + Инфо.OSType + " " + Инфо.CPU + ", " + Инфо.ConsoleUrl + ") " +
// Проверка возможности обновления
? (Инфо.VersionOutdated, " – рекомендуется обновить до версии " + Менеджер.Версия(), "");
```

Кроме этого для удобного получения адреса консоли сервера реализованы методы [ПолучитьАдресКонсоли](#) / [ПопыткаПолучитьАдресКонсоли](#), который можно использовать для отображения консоли встроенными средствами платформы (см. пример в демонстрационной конфигурации в форме [КонсольСервера](#) обработки [СЛК](#)).

Получение системных логов

В СЛК 3.0 реализована запись основных событий и ошибок в лог файлы (их расположение см. в [приложении](#)). Для упрощения поиска этих файлов реализована возможность их получения непосредственно в конфигурации методами [ПолучитьЛог](#) / [ПопыткаПолучитьЛог](#):

```
// Получение серверного лога в виде двоичных данных
Лог = СЛК.МенеджерОбъектов().ПолучитьЛог();
Лог.Записать("D:\licenceserver.log");

// Получение лога компоненты в виде строки
Лог = СЛК.МенеджерОбъектов().ПолучитьЛог(Ложь, 1);
```

Метод реализован как у менеджера объектов, так и объекта компоненты, что позволяет получить лог компоненты до создания менеджера объектов (например, в случае невозможности соединиться с севером).

Приложение: Классы исключений

Базовые

ELicence	Базовый класс исключений
ESystem	Системная ошибка
EServerResult	Класс-признак, что исключение произошло на сервере СЛК, является префиксом перед реальным классом исключения, например: EServerResult.EVersionMismatch

Исключения связи с сервером СЛК

EExecuteError	Ошибка выполнения запроса к серверу СЛК
EVersionMismatch	Несоответствие версии сервера и компоненты
ELocalServerNotFound	Сервер СЛК не обнаружен на локальном компьютере
ELocalServerPreviousVersion	На локальном компьютере запущен сервер СЛК предыдущей версии

Исключения запуска и блокировки сеанса

ENotStarted	Объект компоненты не запущен (метод Запуск не вызван или при его выполнении произошла ошибка)
EAlreadyStarted	Объект компоненты уже запущен (повторный вызов метода Запуск)
ENotInitialized	Менеджер объектов не инициализирован (не вызван метод УстановитьОбъектКомпоненты)
EAlreadyInitialized	Менеджер объектов уже инициализирован (повторный

	вызов метода УстановитьОбъектКомпоненты)
ELicenceFreeLock	Компонента заблокирована из-за освобождения лицензии и отключения от сервера СЛК
ELicenceControlParamsLock	Компонента заблокирована из-за повторной установки параметров контроля лицензий методом УстановитьПараметрыКонтроляЛицензий (в случае несоответствия новых значений уже установленным)
EPreviousVersionLoaded	Уже загружена предыдущая версия компоненты

Исключения, связанные с установкой лицензий

ELicenceValues	Базовый класс
ELicenceBadSignature	Неверная сигнатура
ELicenceBadCRC	Неверная контрольная сумма
ELicenceUnknownVersion	Неизвестная версия
ELicenceUnsupportedVersion	Неподдерживаемая версия
ELicenceAnotherComputerReply	Файловый ответ предназначен для другого компьютера
ELicenceAlreadyInstalled	Лицензия с указанным С/Н уже установлена
EDemoLicenceAlreadyInstalled	Доступная демонстрационная лицензия уже установлена
ELicenceCenterRequest	Ошибка при выполнении запроса в Центр Лицензирования
ELicenceCenterError	Ошибка Центра Лицензирования
ELicenceNotInstalled	Лицензия не установлена
ELicenceNotUninstallable	Лицензия не может быть деактивирована

Исключения, связанные с файлами данных

ETemplateAlreadyLoaded	Файл данных из указанного макета уже загружен
EDataFileNotFound	Не найден файл данных (макет с данными)
EBadFileFormat	Неверный/неизвестный формат файла
EUnsupportedFileFormat	Неподдерживаемый формат файла
EBadFileKeyID	Файл создан ключом другой серии
EObjectNotFound	Защищенный объект не найден
EParamNotFound	Лицензионный параметр не найден

Исключения времени выполнения

ELicenceNotFound	Лицензия не найдена
ELicenceIsNotEnabled	Лицензия недоступна
EBasicLicenceNotFound	Нет доступных основных лицензий, но присутствуют дополнительные
EEnabledLicenceNotFound	Нет доступных лицензий, но присутствуют заблокированные/недоступные
ELicenceCountExceeded	Превышено количество лицензий

ESessionNotFound	Не найден указанный сеанс ИБ
ESessionLicenceNotFound	Не найдена лицензия, связанная с указанным сеансом ИБ
ELicenceIncorrectPassword	Неверный пароль при операции с пользовательской памятью
ESessionNotLogged	Сеанс ИБ не зарегистрирован на сервере СЛК
ESerieNotDefined	Серия не определена

Приложение: Свойства и методы объекта компоненты

ПолучитьОписаниеОшибки (GetErrorDescription)

Синтаксис

ПолучитьОписаниеОшибки()

Тип

Функция, возвращает строку

Описание:

Большинство методов объекта реализовано в виде функций, возвращающих в случае ошибки **Ложь** или **Неопределено**. Функция **ПолучитьОписаниеОшибки** используется для получения информации об ошибке последней операции.

ОшибкаСозданияОбъекта (CreateObjectFailed)

Синтаксис

ОшибкаСозданияОбъекта(ОписаниеОшибки)

Тип

Функция, возвращает строку

Параметры

ОписаниеОшибки (строка, обязательный)

Системное описание ошибки полученное функцией ОписаниеОшибки() при обработке исключения создания менеджера объектов

Описание:

Функция используется для передаче объекту компоненты информации об ошибке создания менеджера объектов (см. обработку исключения в **примере создания менеджера объектов**).

Версия (Version)

Синтаксис

Версия

Тип

Свойство (строка, только чтение)

Описание:

Возвращает версию компоненты

ПроверитьВерсию (CheckVersion)

Синтаксис

ПроверитьВерсию(Значение)

Тип

Функция, возвращает булево или **Неопределено** в случае ошибки

Параметры

Значение (строка, обязательный)

Значение версии, которая проверятся

Описание:

Возвращает **Истину**, если версия компоненты больше или равна переданному значению.

ПараметрыСвязи (ConnectionString)

Синтаксис

ПараметрыСвязи

Тип

Свойство (строка, чтение/запись)

Описание:

Предназначено для установки параметров связи с сервером СЛК. В качестве значения передается строка вида «ServerAddr(host)=<значение>;ServerPort(port)=<значение>», где **ServerAddr** (или **host**) – имя или IP-адрес компьютера, где установлен сервер СЛК; **ServerPort** (или **port**) – порт для связи, например:

```
ОбъектКомпоненты.ПараметрыСвязи = "ServerAddr=SERVER1;ServerPort=9099";
```

Устанавливаемые параметры автоматически не сохраняются, поэтому разработчик должен реализовать сохранение средствами платформы. Например, в демонстрационной конфигурации для этого используется стандартное **ХранилищеОбщихНастроек** (см. функции **ПолучитьПараметрыСвязи** / **СохранитьПараметрыСвязи** общего модуля **СЛК** и форму настроек **Обработка.СЛК.ПараметрыСвязи**).

Запуск (Start)

Синтаксис

Запуск(Серия)

Тип

Функция, возвращает булево

Параметры

Серия (строка, обязательный)

Серия ключей, используемая для защиты конфигурации

Описание:

Выполняет запуск системы и устанавливает соединение с сервером СЛК. В случае успеха возвращает **Истину**.

ПолучитьМенеджерОбъектов (GetObjectManager)

Синтаксис

ПолучитьМенеджерОбъектов()

Тип

Функция, возвращает строку или **Неопределено** в случае ошибки

Описание:

Возвращает имя файла менеджера объектов. В дальнейшем менеджер объектов должен быть подключен из этого файл при помощи встроенных средств платформы (ВнешниеОбработки.Подключить, подробнее см. **Подключение**).

Приложение: Методы менеджера объектов

УстановитьОбъектКомпоненты (SetComponentObject)

Синтаксис

УстановитьОбъектКомпоненты(Объект, Защита)

Тип

Процедура

Параметры

Объект (объект компоненты, обязательный)

Объект внешней компоненты

Защита (объект ОписаниеЗащитыОтОпасныхДействий, необязательный, по умолчанию

Неопределено)

Предназначен для отключения предупреждения механизма защиты от опасных действий при создании внешних обработок / отчетов

Описание:

Связывает менеджер объектов с объектом компоненты. Подробнее см. пример кода в разделе **Подключение**.

УстановитьКодДоступа (SetAccessCode)

Синтаксис

УстановитьКодДоступа(Значение)

Тип

Процедура, устанавливает код доступа для получения лицензий

Параметры

Значение (строка, обязательный)

Значение кода доступа

Аналог, не вызывающий исключения

ПопыткаУстановитьКодДоступа (TrySetAccessCode), в случае ошибки возвращается

Ложь.

Описание:

Подробнее см. **Получение лицензий и параметры контроля**

УстановитьПараметрыКонтроляЛицензии (SetLicenceControlParams)

Синтаксис

УстановитьПараметрыКонтроляЛицензии(ТолькоНаличие, ИНН, КПП, ВерсияПродукта)

Тип

Процедура, устанавливает параметры контроля лицензии

Параметры

ТолькоНаличие (булево, необязательный)

Если **Истина**, то для получения лицензии будет достаточно только наличие ключа

ИНН (число, необязательный)

КПП (число, необязательный)

ВерсияПродукта (число, необязательный)

Параметры, используемые для отбора ключей при получении лицензии

Аналог, не вызывающий исключения

ПопыткаУстановитьПараметрыКонтроляЛицензии (TrySetLicenceControlParams), в случае ошибки возвращается **Ложь**.

Описание:

Подробнее см. **Получение лицензий и параметры контроля**

ПолучитьЛицензию (GetLicence)

Синтаксис

ПолучитьЛицензию()

Тип

Функция, возвращает структуру

Аналог, не вызывающий исключения

ПопыткаПолучитьЛицензию (TryGetLicence), в случае ошибки возвращается

Неопределено.

Описание:

Получает лицензию для текущего сеанса информационной базы и возвращает структуру параметров ключа защиты:

Type	Тип (USB для аппаратных, Virtual для программных)
FileName	Имя файла или символическая ссылка
Enabled	Признак доступности для получения лицензий
IsBlank	Признак неактивированного ключа (болванки)
BlankKeyNo	Серийный номер «болванки»
CanUninstall	Признак возможности деактивации и получения резервного кода
Workable	Признак работоспособности (исправности для аппаратных или соответствия конфигурации компьютера для программных)
OpenError	Описание ошибки открытия
Version	Версия ключа
KeyNo	Серийный номер ключа
KeyID	Серия ключей
KeyType	Тип ключа: 3 – Основной 4 – Дополнительный 5 – Демонстрационный / ПРОМО
ServiceType*	Внутренний тип сервиса, устанавливаемый только при использовании ключа в специальных сервисах 1С: 0 – Обычный вариант использования 1 – Демонстрационный сервер 2 – ГРМ 3 – Fresh 4 – ...
LicenceCount	Количество лицензий
ActivationCode	Код активации, использовавшийся при активации ключа
ActivationDate	Дата активации
Period	Срок действия в днях

ExpireDate	Дата окончания действия
DaysLeft	Оставшиеся дни
Expired	Признак, что срок действия истек
Flags*	Флаги функциональности, строка из 32 символов «0» / «1»
ProductVersion*	Версия продукта, число 0 до 65535
StartDate*	Дата начала действия
ITN**	ИНН
IEC**	КПП
ProductName	Наименование продукта / серии ключей
RegNo***	Регистрационный номер
Article	Артикул номенклатуры
ArticleName	Наименование номенклатуры
SupportDate	Дата начала действия договора ИТС для данного рег. номера
SupportEndDate	Дата окончания действия договора ИТС
SupportPeriod	Срок действия договора ИТС в днях
SupportType	Тип договора ИТС: 0 – неизвестный / неопределенный 1 – 1С:ИТС Проф 2 – 1С:ИТС Техно
IndustrySupportDate	Дата начала действия сервиса ИТС Отраслевой для данного рег. номера
IndustrySupportEndDate	Дата окончания действия сервиса ИТС Отраслевой
IndustrySupportPeriod	Срок действия сервиса ИТС Отраслевой в днях
IndustrySupportType	Тип сервиса: 0 – неизвестный / неопределенный 1 – 1С:ИТС Отраслевой Базовый 2 – 1С:ИТС Отраслевой 1-й Категории 3 – 1С:ИТС Отраслевой 2-й Категории 4 – 1С:ИТС Отраслевой 3-й Категории 5 – 1С:ИТС Отраслевой 4-й Категории 6 – 1С:ИТС Отраслевой 5-й Категории
UpdateDate	Дата и время последнего обновления информации
Message	Сообщение, установленное для данной лицензии в Центре Лицензирования
MessageType	Тип сообщения: 0 – информационное 1 – предупреждение 2 – ошибка
UpgradeDate	Дата и время апгрейда, если был выполнен по данному рег. номеру

- * Устанавливаются при создании ключа в Центре Лицензирования
- ** Устанавливаются при активации ключа у конечного пользователя
- *** Рег. номер и последующие поля устанавливаются только для продуктов выпускаемых / издаваемых фирмой 1С

ПолучитьЛицензии (GetLicences)

Синтаксис

ПолучитьЛицензии(ПоВсемСериям, ТолькоДоступные)

Тип

Функция, возвращает массив структур

Параметры

ПоВсемСериям (булево, необязательный, по умолчанию Ложь)

Если Ложь, возвращаются только ключи текущей серии, если Истина – все установленные на сервере СЛК

ТолькоДоступные (булево, необязательный, по умолчанию Ложь)

Если Истина, то возвращаются только ключи, у которых может быть получена лицензия; если Ложь – все, в т.ч. неисправные.

Аналог, не вызывающий исключения

ПопыткаПолучитьЛицензии (GetLicences), в случае ошибки возвращается

Неопределено.

Описание:

Возвращает массив структур-параметров ключей защиты (описание полей см. ПолучитьЛицензию) ключей, установленных на сервере СЛК

СоздатьОбъект (CreateObject)

Синтаксис

СоздатьОбъект(Имя)

Тип

Функция, возвращает защищенный объект

Параметры

Имя (строка, обязательный)

Имя объекта в файле данных

Аналог, не вызывающий исключения

ПопыткаСоздатьОбъект (TryCreateObject), в случае ошибки возвращается

Неопределено.

Описание:

Создает объект по его имени в файле данных (подробнее см. **Создание защищённых объектов**).

ПолучитьСписокОбъектов (GetObjectList)

Синтаксис

ПолучитьСписокОбъектов(МакетИлиИмя)

Тип

Функция, возвращает массив строк

Параметры

МакетИлиИмя (строка или объект метаданных макет, необязательный)

Наименование общего макет или сам макет, содержащий файл данных.

Псевдоним файла данных на сервере СЛК

Аналог, не вызывающий исключения

ПопыткаПолучитьСписокОбъектов (TryGetObjectList), в случае ошибки возвращается **Неопределено**.

Описание:

Возвращает список имен защищенных объектов, содержащихся в указанном файле данных.

ЗагрузитьИзМакета (LoadFromTemplate)

Синтаксис

ЗагрузитьИзМакета(Макет, Имя, ОбновитьСуществующий, ТипМакета)

Тип

Процедура

Параметры

Макет (объект метаданных макет, обязательный)

Произвольный макет, содержащий файл данных.

Имя (строка, обязательный)

Имя, под которым регистрируется файл

ОбновитьСуществующий (булево или число, необязательный, по умолчанию Ложь)

Признак обновления уже загруженного файла:

0 или Ложь – Не обновлять, возвращать ошибку

1 или Истина – Обновлять

2 – Не обновлять, не возвращать ошибок

ТипМакета (число, необязательный, по умолчанию 0)

Тип файла в макете:

0 – защищенные объекты

1 – лицензионные параметры

Аналог, не вызывающий исключения

ПопыткаЗагрузитьИзМакета (TryLoadFromTemplate), в случае ошибки возвращается **Ложь**.

Описание:

Загружает файл данных из произвольного макета и регистрирует его под указанным именем. Если параметр **ОбновитьСуществующий** равен **Ложь** (или 0) и макет с таким именем уже зарегистрирован, то будет вызвано исключение **ETemplateIsAlreadyLoaded**. Если же параметр **ОбновитьСуществующий** равен 2 и макет зарегистрирован, то метод отработает без ошибок.

МакетЗагружен (TemplateLoaded)

Синтаксис

МакетЗагружен(Имя, ТипМакета)

Тип

Функция

Параметры

Имя (строка, обязательный)

Имя, под которым был зарегистрирован файл при загрузке

ТипМакета (число, необязательный, по умолчанию 0)

Тип файла в макете:

0 – защищенные объекты

1 – лицензионные параметры

Аналог, не вызывающий исключения

ПопыткаМакетЗагружен (TryTemplateLoaded), в случае ошибки возвращается **Неопределено**.

Описание:

Проверяет, был ли загружен файл данных из произвольного макета. Возвращает **Истина**, если загружен и **Ложь**, если нет.

ПолучитьСчетчики (GetCounters)

Синтаксис

ПолучитьСчетчики()

Тип

Функция, возвращает структуру

Аналог, не вызывающий исключения

ПопыткаПолучитьСчетчики (TryGetCounters), в случае ошибки возвращается **Неопределено**.

Описание:

Возвращает текущие значения счетчиков лицензий:

Licences	Количество ключей
Connections	Количество соединений
Sessions	Количество сеансов
TotalLicenceCount	Общее количество лицензий
DemoLicenceCount	Количество демонстрационных лицензий
UseLicenceCount	Количество использованных лицензий
FreeLicenceCount	Количество свободных лицензий

При формировании значений учитываются параметры контроля лицензии (подробнее см. **Получение лицензии и параметры контроля**).

ПолучитьИнформациюОСервере (GetServerInfo)

Синтаксис

ПолучитьИнформациюОСервере()

Тип

Функция, возвращает структуру

Аналог, не вызывающий исключения

ПопыткаПолучитьИнформациюОСервере (TryGetServerInfo), в случае ошибки возвращается **Неопределено**.

Описание:

Возвращает параметры сервера СЛК:

ComputerName	Имя компьютера, на котором установлен сервер СЛК
LocalIP	IP адрес
ComputerID	ID компьютера
OSType	Тип ОС
OSVersion	Версия ОС
ModuleName	Путь к исполняемому модулю сервера
PID	ID процесса
CPU	Тип (Разрядность) процесса
Version	Версия
VersionOutdated	Признак, что сервер можно обновить
ConsoleUrl	Адрес веб консоли

ПолучитьАдресКонсоли (GetConsoleUrl)

Синтаксис

ПолучитьАдресКонсоли()

Тип

Функция, возвращает строку

Аналог, не вызывающий исключения

ПопыткаПолучитьАдресКонсоли (TryGetConsoleUrl), в случае ошибки возвращается

Неопределено.

Описание:

Возвращает адрес консоли сервера СЛК, с которым установлено соединение.

ПолучитьЛог (GetLog)

Синтаксис

ПолучитьЛог()

Тип

Функция, возвращает строку или двоичные данные

Параметры

Сервер (булево, необязательный, по умолчанию Истина)

Признак получения серверного лога, если **Истина**, то возвращается лог сервера, если

Ложь - компоненты

Представление (число, необязательный, по умолчанию 0)

Представление возвращаемых данных:

0 – двоичные данные

1 – строка

Аналог, не вызывающий исключения

ПопыткаПолучитьЛог (TryGetLog), в случае ошибки возвращается **Неопределено.**

Описание:

Возвращает данные указанного лога в указанном представлении.

ПолучитьПараметрыКодаАктивации (GetActivationCodeValues)

Синтаксис

ПолучитьПараметрыКодаАктивации(КодАктивации)

Тип

Функция, возвращает структуру

Параметры

КодАктивации (строка, обязательный)

Код активации программного ключа СЛК

Аналог, не вызывающий исключения

ПопыткаПолучитьПараметрыКодаАктивации (TryGetActivationCodeValues), в случае ошибки возвращается **Неопределено**.

Описание:

Возвращает параметры кода активации:

KeyID	Серия ключей
KeyNo	Серийный номер ключа
ActivationCode	Идентификатор кода активации

УстановитьЛицензию (InstallLicence)

Синтаксис

УстановитьЛицензию(КодАктивации, ИНН, КПП, СерийныйНомерБолванки)

Тип

Функция, возвращает структуру

Параметры

КодАктивации (строка, обязательный)

Код активации программного ключа СЛК

ИНН (число, необязательный)

КПП (число, необязательный)

Параметры ключа

СерийныйНомерБолванки (число, необязательный)

Серийный номер неактивированного ключа СЛК, который необходимо активировать.

Если не указан, установка ключа выполняется на компьютере сервера СЛК.

Аналог, не вызывающий исключения

ПопыткаУстановитьЛицензию (TryInstallLicence), в случае ошибки возвращается **Неопределено**.

Описание:

Выполняет установку программного ключа, в случае успеха возвращаются его параметры (описание полей см. **ПолучитьЛицензию**).

ОбновитьЛицензии (UpdateLicences)

Синтаксис

ОбновитьЛицензии()

Тип

Функция, возвращает массив структур

Аналог, не вызывающий исключения

ПопыткаОбновитьЛицензии (TryUpdateLicences), в случае ошибки возвращается

Неопределено.

Описание:

Выполняет обновление / восстановление программных ключей для компьютера, где работает сервер СЛК. Рекомендуется использовать в ситуации, когда оборудование не менялось, но данные были потеряны, например, в результате вынужденной перестановки ОС. В случае успеха возвращается массив ключей аналогично методу ПолучитьЛицензии.

СоздатьФайловыйЗапрос (CreateFileQuery)

Синтаксис

СоздатьФайловыйЗапрос(КодАктивации, ИНН, КПП, СерийныйНомерБолванки)

Тип

Функция, возвращает строку

Параметры

КодАктивации (строка, обязательный)

Код активации программного ключа СЛК

ИНН (число, необязательный)

КПП (число, необязательный)

Параметры ключа

СерийныйНомерБолванки (число, необязательный)

Серийный номер неактивированного ключа СЛК, который необходимо активировать.

Если не указан, формируется запрос для установки на компьютере сервера СЛК.

Аналог, не вызывающий исключения

ПопыткаСоздатьФайловыйЗапрос (TryCreateFileQuery), в случае ошибки возвращается

Неопределено.

Описание:

Создает строку запроса активации программного ключа для передачи в Центр Лицензирования по электронной почте. Посылать запрос можно как непосредственно в теле письма, так и в прикрепленном файле.

*В случае использования файла, рекомендуется давать ему понятное название, включающее серию ключа, серийный номер и расширение **.txt**, например, **ЗапросАктивации.672В.7107572.txt**.*

СоздатьФайловыйЗапросОбновления (CreateUpdateFileQuery)

Синтаксис

СоздатьФайловыйЗапросОбновления()

Тип

Функция, возвращает строку

Аналог, не вызывающий исключения

ПопыткаСоздатьФайловыйЗапросОбновления (TryCreateUpdateFileQuery), в случае ошибки возвращается **Неопределено**.

Описание:

Создает строку запроса обновления программных ключей для компьютера, где работает сервер СЛК. Посылать запрос можно как непосредственно в теле письма, так и в прикрепленном файле.

*В случае использования файла, рекомендуется давать ему понятное название, включающее серию ключа и расширение **.txt**, например, **ЗапросОбновления.672В.txt**.*

УстановитьФайловыйОтвет (InstallFileReply)

Синтаксис

УстановитьФайловыйОтвет(Ответ)

Тип

Функция, возвращает структуру или массив структур

Параметры

Ответ (строка, обязательный)

Содержимое ответа из Центра Лицензирования

Аналог, не вызывающий исключения

ПопыткаУстановитьФайловыйОтвет (TryInstallFileReply), в случае ошибки возвращается **Неопределено**.

Описание:

Выполняет установку файлового ответа из Центра Лицензирования, полученного по электронной почте. В случае установки программного ключа возвращаются его параметры аналогично методу УстановитьЛицензию; в случае обновления / восстановления возвращается массив полученных ключей аналогично методу ОбновитьЛицензии.

ПрочитатьПамять (ReadMemory)

Синтаксис

ПрочитатьПамять(СерийныйНомер, Пароль)

Тип

Функция, возвращает строку

Параметры

СерийныйНомер (число, обязательный)

Серийный номер ключа, память которого необходимо прочитать

Пароль (строка, обязательный)

Пароль на чтение

Аналог, не вызывающий исключения

ПопыткаПрочитатьПамять (TryReadMemory), в случае ошибки возвращается

Неопределено.

Описание:

Возвращает содержимое пользовательской памяти указанного ключа. По умолчанию пароли ключа равны пустой строке, для их изменения необходимо использовать метод [УстановитьПароли](#) / [ПопыткаУстановитьПароли](#).

ЗаписатьПамять (WriteMemory)

Синтаксис

ЗаписатьПамять(СерийныйНомер, Пароль, Данные)

Тип

Процедура

Параметры

СерийныйНомер (число, обязательный)

Серийный номер ключа, память которого необходимо записать

Пароль (строка, обязательный)

Пароль на чтение

Аналог, не вызывающий исключения

ПопыткаЗаписатьПамять (TryWriteMemory), в случае ошибки возвращается **Ложь**.

Описание:

Записывает данные в пользовательскую память указанного ключа. По умолчанию пароли ключа равны пустой строке, для их изменения необходимо использовать метод [УстановитьПароли](#) / [ПопыткаУстановитьПароли](#). При работе с аппаратными ключами объем данных ограничен размером пользовательской памяти устройства, для программных ключей объем данных неограничен.

ОбнулитьПамять (ResetMemory)

Синтаксис

ОбнулитьПамять(СерийныйНомер)

Тип

Процедура

Параметры

СерийныйНомер (число, обязательный)

Серийный номер ключа, память которого необходимо обнулить

Аналог, не вызывающий исключения

ПопыткаОбнулитьПамять (TryResetMemory), в случае ошибки возвращается **Ложь**.

Описание:

Сбрасывает пароли и обнуляет пользовательскую память указанного ключа.

УстановитьПароли (SetPasswords)

Синтаксис

УстановитьПароли(СерийныйНомер, ПарольНаЧтение, ПарольНаЗапись,
НовыйПарольНаЧтение, НовыйПарольНаЗапись)

Тип

Процедура

Параметры

СерийныйНомер (число, обязательный)

Серийный номер ключа, пароли которого необходимо установить

ПарольНаЧтение (строка, обязательный)

Текущий пароль на чтение

ПарольНаЗапись (строка, обязательный)

Текущий пароль на запись

НовыйПарольНаЧтение (строка, обязательный)

Устанавливаемый пароль на чтение

НовыйПарольНаЗапись (строка, обязательный)

Устанавливаемый пароль на запись

Аналог, не вызывающий исключения

ПопыткаОбнулитьПамять (TrySetPasswords), в случае ошибки возвращается **Ложь**.

Описание:

Устанавливает пароли на пользовательскую память указанного ключа. Для успешного выполнения необходимо верно указать текущие пароли.

ПолучитьПараметр (GetParam)

Синтаксис

ПолучитьПараметр(СерийныйНомер, Имя, ИспользоватьОсновнойКлюч)

Тип

Функция, возвращает строку

Параметры

СерийныйНомер (число, обязательный)

Серийный номер ключа, значение параметра которого необходимо получить

Имя (строка, обязательный)

Имя параметра в файле

ИспользоватьОсновнойКлюч (булево, необязательный, по умолчанию Ложь)

Аналог, не вызывающий исключения

ПопыткаПолучитьПараметр (TryGetParam), в случае ошибки возвращается

Неопределено.

Описание:

Получает значение лицензионного параметра для указанного ключа. Если это значение не определено и параметр **ИспользоватьОсновнойКлюч** равен **Ложь**, то возвращается значение параметра по умолчанию.

Если значение для указанного ключа не определено и параметр

ИспользоватьОсновнойКлюч равен **Истина**, то возвращается значение для основного ключа, в качестве которого считается ключ, указанный в файле параметров как главный основной ключ (**PrimaryKeyNo**). Если главный основной ключ в файле параметров не определен, то таким считается первый обнаруженный основной.

Если значение параметра не определено ни для указанного, ни для основного ключа, то возвращается значение параметра по умолчанию.

См. также **Создание файлов данных** и **Формат конфигурационного файла параметров**.

Приложение: Демонстрационная конфигурация

В демонстрационной конфигурации **LicenceDemoDb.cf** реализованы примеры использования основных возможностей СЛК – подключение системы, получение лицензии и создание защищенных объектов, установка лицензий непосредственно из конфигурации и отображение консоли сервера СЛК и т.п.

*По умолчанию демонстрационная конфигурация настроена на работу с тестовой серией ключей **672В**: значение серии установлено в функции **Серия** общего модуля **СЛК** и*

общих макетах **ОбъектыСЛК** и **ПараметрыСЛК** содержатся файлы данных подготовленные именно для этой тестовой серии.

Соответственно, для работы с другой серией ключей необходимо заменить возвращаемое функцией Серия значение и поместить в макеты файлы данных, подготовленные для нужной серии.

В конфигурации используются следующие модули и объекты метаданных:

ОбщийМодуль.СЛК

Серверный модуль, где реализованы команды СЛК – подключение компоненты, создание менеджера объектов, вызовы менеджера объектов для получения лицензий или создания объектов и т.п.

ОбщийМодуль.СЛКПовтИсп

Специальный серверный модуль для увеличения производительности

ОбщийМодуль.СЛКОбщие

Вспомогательные процедуры и функции

ОбщийМакет.КомпонентаСЛК

Макет с архивом внешней компоненты `licenceaddin-{%version%}-template.zip` из комплекта поставки

ОбщийМакет.ОбъектыСЛК

Макет с файлом защищенных объектов

Обработка.СЛК

Обработка, в формах которой реализованы примеры работы

Форма.ПроверкаКоманд

Примеры основных команд СЛК – получение лицензии, создание защищенных объектов, работа с памятью ключа и т.п.

Форма.КонсольСервера

Пример отображения веб консоли сервера СЛК при помощи встроенных средств платформы

Форма.УстановкаЛицензий

Пример установки / обновления лицензий

Форма.ПараметрыСвязи

Пример настройки параметров связи с севером СЛК

Макет.ОбъектыСЛКВМакетеОбработки

Макет с файлом защищенных объектов

Приложение: Создание файлов данных

СЛК 3.0 совместима по форматам файлов данных с предыдущими версиями, поэтому для создания файлов данных возможно использовать редактор файлов данных от СЛК 2.1.

При помощи редактора файлов данных можно только создавать файлы данных, извлечь данные из созданных файлов при помощи редактора алгоритмически невозможно

Кроме этого, в отличие от предыдущих версий редактор СЛК 3.0 может **работать без ключа защиты** – создание файлов данных выполняется при помощи открытого ключа разработчика, поставляемого в комплекте разработчика для конкретной серии ключей в виде текстового файла вида `{%Серия%}.cryptkey`.

В СЛК 3.0 основной редактор файлов реализован в виде консольной программы **licenceedit**, доступной для платформ Windows и Linux, для использования в автоматическом режиме.

Такой подход позволяет автоматизировать процесс создания файлов данных и их загрузку в макеты конфигурации.

Для удобства разработчика реализован и графический интерфейс редактора **licenceedit-ui**, также доступный для платформ Windows и Linux, который представляет собой, фактически, «обертку» для консольной программы.

Использование:

```
licenceedit <Команда> <Имя файла> [<Источник1>,<Источник2>...<ИсточникN>]
[<Параметры>...]
```

Команды:

c, co

Создание файла защищенных объектов

p, sp

Создание файла лицензионных параметров

l, list

Вывод содержимого файла

Источник (и):

Имя или маска файлов исходных данных – файлов обработок / отчетов или конфигурационного файла лицензионных параметров (формат файла см. ниже).

При создании файлов защищенных объектов, объектам присваиваются имена файлов без расширения. Например, обработка из файла «ЗащищеннаяОбработка.epf» получит имя «ЗащищеннаяОбработка».

*В ОС **Windows** при использовании в именах файлов русских букв необходимо учитывать кодировку. Например, если редактор вызывается из командного файла в кодировке *ansi* (CP1251 для русских региональных настроек), то необходимо указать это при помощи ключа **-cen(--cmdencoding)=ansi**.*

Параметры:

-s, --serie={KeyID}

Серия ключа (если не указана, то извлекается из имени создаваемого файла)

-ck, --cryptkey={FileName}

Имя файла ключа шифрования (если не указан, то принимается <KeyID>.cryptkey)

-y, --yes

Автоматически отвечать «Да» на все запросы

-cen, --cmdencoding={oem|ansi|utf8}

Кодировка параметров командной строки (только для ОС Windows; по умолчанию «oem», как при выполнении в командной строке)

-en, --encoding={system|utf8|utf16}

Кодировка исходного файла лицензионных параметров, по умолчанию «system» - ANSI для ОС Windows и UTF-8 для ОС Linux (см. формат файла параметров)

Коды возврата:

0 : Нет ошибок

1 : Неизвестная / системная ошибка

2 : Операция отменена пользователем

3 : Серия ключей не указана или указана с ошибкой

4 : Не найдены исходные данные

5 : Ошибка при получении ключа шифрования

Примеры использования

licenceedit c my.datafile "D:/Samples/*.epf" -y --serie=672B

Создание файла данных серии 672B и добавление в него всех внешних обработок из папки D:\Samples\. Если файл уже существует, то его содержимое будет автоматически перезаписано (ключ **-y**).

licenceedit l my.datafile --serie=672B

Вывод содержимого файла данных my.datafile

licenceedit p 672B.paramfile "D:/Samples/ParamFile.config" -y

Создание файла параметров серии 672B на основе конфигурационного файла из папки D:\Samples. Т.к. серия не указана явно, то извлекается из имени создаваемого файла **672B.paramfile**.

Формат конфигурационного файла параметров

Конфигурационный файл параметров в СЛК 3.0 аналогичен используемому в предыдущих версиях СЛК и является обычным текстовым ini-файлом. Однако, в СЛК 3.0 поддерживается несколько вариантов его кодировки, указываемых параметром командной строки **-en,--encoding={system|utf8|utf16}**:

system

Текущая системная кодировка – ANSI для ОС Windows и UTF-8 для ОС Linux. Для русских региональных настроек ОС Windows – CP1251, поэтому используется по умолчанию для совместимости с файлами предыдущих версий СЛК.

utf-8

Юникод UTF-8, рекомендуемый

utf-16

Юникод UTF-16

Пример конфигурационного файла параметров

[Common]

PrimaryKeyNo=<Серийный номер ключа, который должен считаться основным>

[<ИмяПараметра1>]

DefaultValue=<Значение по умолчанию>

XXX1=<Значение для ключа с серийным номером XXX1>

XXX2=<Значение для ключа с серийным номером XXX2>

[<ИмяПараметра2>]

DefaultValue=<Значение по умолчанию>

XXX1=<Значение для ключа с серийным номером XXX1>

XXX2=<Значение для ключа с серийным номером XXX2>

[<ИмяПараметраN>]

```
DefaultValue=<Значение по умолчанию>
XXX1=<Значение для ключа с серийным номером XXX1>
XXX2=<Значение для ключа с серийным номером XXX2>
```

Приложение: Расположение конфигурационных файлов

Для ОС Windows

`%ProgramData%\1C\licence\3.0\`

Где `%ProgramData%` - системная папка общих настроек. Например, для ОС Windows Vista и выше это может быть:

`C:\Program Data\`

Для ОС Linux

`/var/1C/licence/3.0/`

Конфигурационные файлы представляет себя обычные текстовые ini-файлы в кодировке UTF-8. Примеры конфигурационных файлов с комментариями (на английском) включены в установочные пакеты как сервера, так и компоненты.

Приложение: Расположение системных логов

Для сервера СЛК

Имя файла

`licenceserver.{%timestamp%}.{%pid%}.log`

Где:

`{%timestamp%}` – время создания файла в виде YYYYMMDD-HHNNSS

`{%pid%}` – идентификатор процесса, создавшего файл

Расположение в ОС Windows

`%ProgramData%\1C\licence\3.0\logs\licenceserver*.log`

Где `%ProgramData%` - системная папка общих настроек. Например, для ОС Windows Vista и выше это может быть:

`C:\Program Data\`

Расположение в ОС Linux

`/var/1C/licence/3.0/logs/licenceserver*.log`

Для компоненты

Имя файла

`licenceaddin.{%timestamp%}.{%pid%}.{%modulename%}.log`

Где:

{%timestamp%} – время создания файла в виде YYYYMMDD-HHNNSS

{%pid%} – идентификатор процесса, создавшего файл

{%modulename%} – имя файла компоненты без расширения

Расположение в ОС Windows

`%LocalAppData%\1C\licence\3.0\logs\licenceaddin*.log`

Где `%LocalAppData%` - папка настроек пользователя, от имени которого запущен процесс 1С. Например, для сервера приложений это может быть:

`C:\Users\USR1CV8\AppData\Local\1C\Licence\3.0\logs\licenceaddin*.log`

Логи компоненты, подключаемой веб-сервером (IIS, Apache) при публикации файловых баз, сохраняются в общей папке, аналогично логам сервера СЛК:

`%ProgramData%\1C\licence\3.0\logs\licenceaddin*.log`

Расположение в ОС Linux

`/home/{user}/.1C/licence/3.0/logs/licenceaddin*.log`

Где `{user}` - пользователь, от имени которого запущен процесс 1С. Например, для сервера приложений это может быть:

`/home/usr1cv8/.1C/licence/3.0/logs/licenceaddin*.log`

Режим сохранения / перезаписи

По умолчанию логи сохраняются только для текущего сеанса работы и при следующем запуске приложения (сервера СЛК или защищенной конфигурации) эти файлы удаляются / перезаписываются. Однако в целях отладки логи предыдущих сеансов работы можно хранить, для этого в конфигурационных файлах необходимо установить флаг сохранения `KeepPreviousLogs`:

[Common]

```
; Хранить логи предыдущих сеансов работы
KeepPreviousLogs=1
```