НПФ «СИГМА-ИС»

Учёт рабочего времени универсальный

Руководство программисту по разработке драйверов

Редакция 1

ОГЛАВЛЕНИЕ

1. Назначение руководства

Данное руководство предназначено для описания принципов создания драйверов (библиотек) доступа к протоколам событий систем СКУД, не входящих в комплект поставки ПО УРВ.

Разработка драйвера (библиотеки) позволяет предоставить ПО УРВ возможности осуществить стандартизированный доступ к протоколам событий СКУД сторонних разработчиков, не внося изменений в исходный текст ПО УРВ.

Разрабатываемые библиотеки подразделяются на два типа:

- стандартные библиотеки (драйвера) доступа к протоколам событий;
- библиотеки (драйвера) доступа к протоколам событий в режиме работы ПО УРВ по умолчанию.

Библиотеки разрабатываются на языке программирования C# в среде программирования Microsoft Visual Studio 2005.

Каждая разрабатываемая библиотека должна иметь префикс наименования в формате swt_наименование_драйвера.dll, например: swt_shu024.dll.

Обеспечение унификации передаваемых данных между ПО УРВ и драйвером (библиотекой) обеспечивается стандартными классами, реализованными в библиотеке URTReferences.dll. Разрабатываемый драйвер должен иметь ссылку (References) на данную библиотеку.

Каждый драйвер (библиотека) имеет стандартное заголовочное описание, включающее уникальный идентификатор, наименование, версию и поддерживаемый режим работы. Данное описание позволяет обеспечить его уникальность и идентификацию программным обеспечением учёта рабочего времени как драйвер доступа к протоколу событий СКУД.

Реализация доступа к структуре ОШС, зонам доступа и событиям, а также протоколам событий осуществляется путём стандартизации перечня требуемых функций, их входных и выходных параметров.

Работа с драйвером (библиотекой) осуществляется путём его динамической загрузки и вызовом требуемых функций.

2. Описание библиотеки URTReferences.dll

Данная библиотека предназначена для реализации стандартных классов, обеспечивающих стандартизацию передаваемой информации между ПО УРВ и драйвером (библиотекой).

Данная библиотека включает в свой состав следующие классы.

Базовый класс организационно-штатной структуры - организация

```
namespace URTObjects
{
	/// Класс - базовый класс ОШС (организация)
	public class ORGObj
	{
		public int IdObj = 0;			// Идентификатор объекта ОШС
		public int IdTypeObj = 0;		// Тип объекта: 1-организация, 2-
						подразделение, 3-должность, 4-
						сотрудник
		public int IdParentObj = 0;	// Ссылка на идентификатор родителя
		public string NameObj = "";	// Наименование, текст
		public Guid xGuid;			// GUID - идентификатор источника
						(драйвера)
		public byte[] zBlob;			// Структура в двоичном виде,
						информация драйвера
		 public ORGObj()
		 {
		 }
	}
}
```

Класс организационно-штатной структуры - подразделение

```
namespace URTObjects
{
	/// Класс - подразделение
	public class ORGPodr : ORGObj
	{
		 public ORGPodr()
		 {
		 }
	}
}
```

Класс организационно-штатной структуры - должность

```
namespace URTObjects
{
	/// Класс - должность организации
	public class ORGDoljn : ORGObj
	{
		public int IdDicDol;	// Идентификатор типа должности из словаря
					типов должностей
		 public ORGDoljn()
		 {
		 }
	}
}
```

Класс организационно-штатной структуры - сотрудник

```
namespace URTObjects
{
      /// Класс - сотрудник организации
      public class ORGUser : ORGObj
      {
            public string Family = "";     // Фамилия
            public string Name = "";       // Имя
            public string Otchestvo =      // Отчество
            "";
            public string TabNumber =      // Табельный номер
            "";
            public byte[] zFingers;        // Специализированные данные о
                                           сотруднике, используемые в режиме
                                           прямой работы с драйвером
             public ORGUser()
             {
             }
      }
}
```

Класс организационно-штатной структуры – типовая (штатная) должность

```
namespace URTObjects
{
      /// Класс - тип должности
      public class ORGDicDoljn
      {
            public int IdObj = 0;          // Идентификатор типа должности
            public string NameObj = "";    // Наименование, текст
            public Guid xGuid;             // GUID - идентификатор источника
                                           (драйвера)
            public byte[] zBlob;           // Структура в двоичном виде,
                                           информация драйвера
             public ORGDicDoljn()
             {
             }
      }
}
```

Класс – рабочая зона

```
namespace URTObjects
{
    // Класс - рабочая зона
    public class JZone
    {
            public int JZoneId;            // Идентификаторы рабочей зоны
            public string Name = "";       // Наименование рабочей зоны
            public Guid xGuid;             // GUID - идентификатор источника
                                           (драйвера)
            public byte[] zBlob;           // Структура в двоичном виде,
                                           информация драйвера
        public JZone()
             {
             }
    }
}
```

Класс – событие рабочей зоны

```
namespace URTObjects
{
    // Класс - событие
    public class EventJZ
    {
            public int EventId = 0;          // Идентификатор события
            public int JZoneId = 0;          // Идентификатор рабочей зоны
            public string EventName =        // Наименование события
            "";
            public int EventType = 1;        // Тип события: 1-входа; 0-выхода
            public Guid xGuid;               // GUID - идентификатор источника
                                             (драйвера)
            public byte[] zBlob;             // Структура в двоичном виде,
                                             информация драйвера
        public EventJZ()
            {
            }
    }
}
```

Класс – рабочая зона элемента организационно-штатной структуры

```
namespace URTObjects
{
    // Класс - рабочая зона элемента ОШС
    public class ObjJZone
    {
            public int Obj = 0;              // Идентификатор элемента ОШС
            public int Zone = 0;             // Идентификатор рабочей зоны
        public ObjJZone()
            {
            }
    }
}
```

Класс – статистика по протоколу событий драйвера (библиотеки)

```
namespace URTObjects
{
    // Класс - статистика по протоколу
    public class ProtocolStatistic
    {
            public DateTime pStart =         // Дата и время начала протокола
            DateTime.Today;
            public DateTime pStop =          // Дата и время окончания протокола
            DateTime.Today;
            public int EventCount = 0;       // Количество событий протокола
        public ProtocolStatistic ()
            {
            }
    }
}
```

Класс – событие протокола в формате ПО УРВ

```
namespace URTObjects
{
```

```
        /// Класс - событие протокола ПО УРВ
        public class ProtocolEvent
        {
              public int IdUser = 0;          // Идентификатор сотрудника
              public DateTime DT;             // Дата и время события
              public int IdZone = 0;          // Идентификатор рабочей зоны
              public int EvType = 1;          // Тип события: 1-входа; 0-выхода
               public ProtocolEvent()
               {
               }
        }
}
```

Класс, предназначенный для работы с протоколом событий ПО УРВ

```
namespace URTDataBase
{
    // Класс - работа с БД ПО УРВ
    public class URTDataBase
    {
            public bool isConnected = false;        // Признак наличия
                                                    подключения
            public void Connect(bool pDemoMode)     // Функция подключения к БД
                                                    ПО УРВ
            public void Disconnect()                // Функция отключения от БД
                                                    ПО УРВ
            public bool                             // Функция добавления
            AddProtocolEvent(ProtocolEvent          события протокола в БД ПО
            pProtocolEvent)                         УРВ
            public bool                             // Функция удаления событий
            DeleteProtocolEventsByPeriod(DateTime   протокола из БД ПО УРВ за
            pStart, DateTime pStop)                 период
        public URTDataBase ()
            {
            }
    }
}
```

Описание параметров функций

public void Connect(bool pDemoMode)

Входные параметры	Выходные параметры
Булево – признак работы в демонстрационном режиме. При использовании в драйвере необходимо передавать параметр **False**.	нет

public void Disconnect()

Входные параметры	Выходные параметры
нет	нет

public bool AddProtocolEvent(ProtocolEvent pProtocolEvent)

Входные параметры	Выходные параметры
Класс – событие протокола ПО УРВ в формате класса ProtocolEvent	Булево – результат операции (True – добавлено, False – не добавлено)

public bool DeleteProtocolEventsByPeriod(DateTime pStart, DateTime pStop)

Входные параметры	Выходные параметры
pStart – Дата и время начала интервала; pStop - Дата и время окончания интервала.	Булево – результат операции (True – удалены, False – не удалены)

Класс – форма отображения выполняемого процесса

```
namespace URTObjects
{
    public partial class ProgressForm : Form
    {
            lbProgressName   Текстовый компонент (Label) для отображения
                             наименования выполняемого процесса
            prgBar           Компонент (ProgressBar) отображения выполнения
                             процесса
        public ProgressForm()
        {
            InitializeComponent();
        }
    }
}
```

3. Описание драйвера (библиотеки)

2.1. Заголовочная информация

Каждый драйвер должен содержать публикуемый класс, имеющий наименование **Importer**, и имеющий публикуемые свойства, приведённые ниже. Данные свойства предназначены для однозначной и уникальной идентификации драйвера (библиотеки).

```
public class Importer
{
        public string NameImporter =    // Наименование драйвера
        "Программное обеспечение        (библиотеки)
        Рубеж-08";
        public Guid ImpGuid = new       // GUID драйвера (библиотеки)
        Guid("2EBCE718-2FC1-4BE9-
        BB0C-0B57D11F9EF5");
        public string Version =         // Версия
        "1.0";
        public bool isDefaultMode =     // Поддержка режима поумолчанию
        false;
}
```

2.2. Описание реализуемых функций

Драйвер (библиотека) должен обеспечивать реализацию следующих функций.

```
public string Get_ImporterName()    // Функция получения наименования драйвера
                                    (библиотеки)
public Guid Get_ImporterGuid()      // Функция получения GUID драйвера
                                    (библиотеки)
public string                       // Функция получения версии драйвера
Get_ImporterVersion()               (библиотеки)
public bool Get_DefaultMode()       // Функция получения признака поддержки
                                    режима поумолчанию
public ArrayList                    // Функция получения перечня рабочих зон
Get_JobZoneList()
public void Get_OSHS(ref            // Функция получения организационно-
ArrayList pObjList, ref             штатной структуры организации
ArrayList pDicDolList)
public void Get_JobZoneEvent(ref    // Функция получения события рабочей зоны
string evName, ref byte[]
EvData)
public void                         // Функция получения статистики по
Get_ProtocolStatistic(ref           протоколу драйвера (библиотеки)
ProtocolStatistic pPs)
public void Get_Protocol(DBInfo     // Функция получения протокола событий за
xDBInfo, DateTime pStart,           указанный период времени
DateTime pStop,

ref ArrayList xListOSHS, ref
ArrayList xListJZones,

ref ArrayList xListJZonesEvents,
ref ArrayList xListObjJZones)
```

Детальное описание параметров функций

public string Get_ImporterName()

Входные параметры	Выходные параметры
нет	Строка – Наименование драйвера (библиотеки)

public Guid Get_ImporterGuid()

Входные параметры	Выходные параметры
нет	GUID – Глобальный уникальный идентификатор драйвера (библиотеки)

public string Get_ImporterVersion()

Входные параметры	Выходные параметры
нет	Строка – Версия драйвера (библиотеки)

public bool Get_DefaultMode()

Входные параметры	Выходные параметры
нет	Булево – Поддержка драйвером (библиотекой) режима работы по умолчанию

public ArrayList Get_JobZoneList()

Входные параметры	Выходные параметры
нет	Массив – Массив рабочих зон в формате класса JZone

public void Get_OSHS(ref ArrayList pObjList, ref ArrayList pDicDolList)

Входные параметры	Выходные параметры
Параметры передаются по ссылке ref 1) Массив – Перечень элементов организационно-штатной структуры в формате класса ORGObj (ORGDoljn, ORGDoljn, ORGUser); 2) Массив – Перечень типовых должностей организации в формате класса в формате класса ORGDicDoljn	нет

public void Get_JobZoneEvent(ref string evName, ref byte[] EvData)

Входные параметры	Выходные параметры
Параметры передаются по ссылке ref 1) Строка – наименование события; 2) Двоичный массив – сериализованный элемент (класс) реализации события в драйвере (библиотеке)	нет

public void Get_ProtocolStatistic(ref ProtocolStatistic pPs)

Входные параметры	Выходные параметры
Параметры передаются по ссылке ref Класс – статистика по протоколу событий драйвера (библиотеки) в формате класса ProtocolStatistic	нет

```
public void Get_Protocol(DBInfo xDBInfo, DateTime pStart,
                         DateTime pStop, ref ArrayList xListOSHS,
                         ref ArrayList xListJZones,
                         ref ArrayList xListJZonesEvents,
                         ref ArrayList xListObjJZones)
```

Входные параметры	Выходные параметры
1) Класс – параметры БД ПО УРВ, в которую записываются события протокола в формате класса DBInfo; 2) Дата и время – дата и время начала интервала выборки; 3) Дата и время – дата и время окончания интервала выборки Параметры передаются по ссылке ref 4) Массив – перечень элементов ОШС в формате класса ORGObj (ORGDoljn, ORGDoljn, ORGUser) 5) Массив – перечень рабочих зон в формате класса JZone 6) Массив – перечень событий входа/выхода рабочих зон в формате класса EventJZ 7) Массив – перечень рабочих зон элементов ОШС в формате класса ObjJZone	нет

2.3. Описание встроенных функций

Драйвер (библиотека) должен иметь стандартные функции преобразования (сериализации) собственных классов в массив байтов и обратно.

```
// Функция сериализации объекта в массив байтов
public byte[] ObjectSerialize(Object pSObj)
{
    MemoryStream MS = new MemoryStream();
    BinaryFormatter bf = new BinaryFormatter();
    bf.Serialize(MS, pSObj);
    byte[] RetObj = new byte[MS.Length];
    MS.Seek(0, SeekOrigin.Begin);
    MS.Read(RetObj, 0, (int)MS.Length);
    MS.Close();
    return RetObj;
}

// Функция десериализации объекта из массива байтов
public Object ObjectDeSerialize(byte[] val)
{
    Object retDSObj = null;
    MemoryStream MS = new MemoryStream();
    MS.Write(val, 0, val.Length);
    if ((val.Length == 0) || (val.Length == 1))
    {
        MS.Close();
        return retDSObj;
    }
    BinaryFormatter bf = new BinaryFormatter();
    MS.Seek(0, SeekOrigin.Begin);
```

```
        try
        {
            retDSObj = (Object)bf.Deserialize(MS);
        }
        catch (Exception ex)
        {
            MessageBox.Show("Не удалось прочитать восстановить объект по
                             причине:\n" + ex.ToString());
        }
        MS.Close();
        return retDSObj;
    }
```

При формировании протокола событий драйверу (библиотеке) требуется проведение анализа, является ли соответствующая рабочая зона рабочей для сотрудника (зона может быть задана рабочей для вышестоящего элемента организационно-штатной структуры). Для этого в ней должна быть реализована соответствующая функция, возвращающая булево значение (true – зона рабочая, false – в противном случае).

```
// Рекурсивная функция поиска признака, является ли данная рабочая
   зона рабочей для данного сотрудника
public bool Get_PriznakZoneIsJob(ORGObj usrORGObj, JZone pJZone,
                                 ArrayList pListOSHS, ArrayList
                                 pListJZones, ArrayList
                                 pListObjJZones)
{
    bool retVal = false;
    if ((usrORGObj == null) | (pJZone == null) | (pListOSHS == null)
       | (pListJZones == null) | (pListObjJZones == null)) {return
       false;}
          // поиск, является ли данная зона рабочей для данного
          сотрудника
          for (int i=0; i<pListObjJZones.Count; i++)
          {
               ObjJZone xObjJZone = (ObjJZone)pListObjJZones[i];
               // является
               if ((xObjJZone.Obj == usrORGObj.IdObj) &
               (xObjJZone.Zone == pJZone.JZoneId))
                {
                      retVal = true;
                      break;
                }
          }
          // не является, ищем её у вышестоящих элементов
          if (retVal == false)
          {
                // поиск вышестоящего элемента
                for (int i = 0; i < pListOSHS.Count; i++)
                {
                      ORGObj xORGObj = (ORGObj)pListOSHS[i];
                      if (xORGObj.IdObj == usrORGObj.IdParentObj)
                      {
                            // поиск признака для вышестоящего
                            элемента
                            retVal = Get_PriznakZoneIsJob(xORGObj,
                            pJZone, pListOSHS, pListJZones,
                            pListObjJZones);
                            break;
```

```
                        }
                    }
                }
            return retVal;
        }
```

2.4. Особенности реализации драйвера

При формировании выходных массивов функций (элементов ОШС, типовых должностей, рабочих зон, событий входа/выхода рабочих зон, дополнительных данных о сотруднике) в процессе импорта и настройки драйвер должен обеспечивать преобразование свой внутренней реализации в двоичные массивы данных. Эти данные хранятся в свойствах соответствующих классов - `public byte[] zBlob` и `public byte[] zFingers`. ПО УРВ при проведении импорта сохраняет информацию о драйвере, через который данные были импортированы.

В ходе получения протокола событий из драйвера, он производит восстановление из двоичных массивов особенностей своей реализации, и формирование выходного протокола событий производит на основе её анализа.

2.5. Особенности реализации драйвера, предоставляющего возможности работы с ПО УРВ в режиме по умолчанию

Режим работы драйвера (библиотеки) с ПО УРВ по умолчанию предусматривает реализацию такого же перечня функционала, как и для обычного драйвера (библиотеки) и дополнительного ряда функциональных возможностей.

Для режима работы по умолчанию, драйвер не должен предоставлять возможности импорта организационно-штатной структуры (осуществляется возврат пустого массива элементов ОШС), а также возможностей настройки событий входа/выхода рабочих зон.

Ввод организационно-штатной структуры организации производится в программном обеспечении УРВ и хранится в его БД; настройка перечня рабочих зон и событий входа/выхода рабочих зон производится при первом запуске УРВ. Функциональные возможности ПО УРВ по импорту ОШС, рабочих зон, настройки событий входа/выхода, выбора драйвера (библиотеки) источника протокола событий в данном режиме отключаются.

Специфика реализации, требуемая для работы драйвера в режиме по умолчанию должна предусматривать реализацию следующих функций.

```
public void                          // Функция получения стандартного события
Get_StandartJZEnterEvent(JZone       входа для рабочей зоны
pJZone, ref string evName, ref
byte[] EvData)
public void                          // Функция получения стандартного события
Get_StandartJZExitEvent(JZone        выхода для рабочей зоны
pJZone, ref string evName, ref
byte[] EvData)
public void Get_UserData(ORGObj      // Функция получения дополнительных данных
pORGObj, ref byte[] FNG)             о сотруднике (отпечатков пальцев, номеров
                                     карт доступа и т.д.)
public void                          // Функция изменения информации о
Update_UserData(ORGObj xORGObj)      сотруднике (добавление, изменение), если
                                     драйвер реализует функцию хранения данных
                                     о сотрудниках в устройстве
public void                          // Функция удаления сотрудников, если
DeleteUsers(ArrayList xUsers)        драйвер реализует функцию хранения данных
                                     о сотрудниках в устройстве
```

Детальное описание параметров функций

public void Get_StandartJZEnterEvent(JZone pJZone, ref string evName, ref byte[] EvData)

Входные параметры	Выходные параметры
1) Класс – рабочая зона в формате класса JZone; Параметры передаются по ссылке ref 2) Строка – наименование события 3) Двоичный массив – сериализованный элемент (класс) реализации события входа в драйвере (библиотеке)	нет

public void Get_StandartJZExitEvent(JZone pJZone, ref string evName, ref byte[] EvData)

Входные параметры	Выходные параметры
1) Класс – рабочая зона в формате класса JZone; Параметры передаются по ссылке ref 2) Строка – наименование события 3) Двоичный массив – сериализованный элемент (класс) реализации события выхода в драйвере (библиотеке)	нет

public void Get_UserData(ORGObj pORGObj, ref byte[] FNG)

Входные параметры	Выходные параметры
1) Класс – сотрудник в формате класса ***ORGObj***; Параметры передаются по ссылке ref 2) Двоичный массив – сериализованный элемент (класс) реализации дополнительных данных о сотруднике в драйвере (библиотеке)	нет

public void Update_UserData(ORGObj xORGObj)

Входные параметры	Выходные параметры
Класс – сотрудник в формате класса ***ORGObj***	нет

```
public void DeleteUsers(ArrayList xUsers)
```

Входные параметры	Выходные параметры
Массив – массив отрудников в формате класса **ORGObj**	нет