Утверждаю
Директор
ООО «Сигма-Софт
Автоматизация»
_____М.И. Мальцев
“____”_____2023 г.

Программный комплекс «С-Платформа» (S-Platform)

Руководство программиста

RU.82469608.0001-02 33

Том 2

Сервер приложений и АРМ

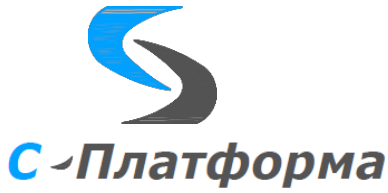
Версия 1.6

Руководитель разработки
Начальник департамента

_____И.О. Урухин
“_____”_____2023 г.

Ответственный исполнитель
Ведущий инженер-программист
_____ М.Ю. Дьяченко
“_____”_____2023 г.

ООО «Сигма Софт»
2023 г.



Утвержден

RU.82469608.0001-02 33

Программный комплекс «С-Платформа» (S-Platform)

Руководство программиста (том 2)

RU.82469608.0001-02 33

Том 2

Сервер приложений и АРМ

Версия 1.6

Листов 51

ООО «Сигма Софт»
2023 г.

АННОТАЦИЯ

Настоящий документ содержит руководство программиста программного комплекса «С-Платформа» (далее по тексту – ПК).

ПК «С-Платформа» служит для создания многоуровневых диспетчерских информационно-управляющих систем реального времени. Ядром системы является сервер приложений из состава программной платформы КОТМИ-14.

Для решения специфических информационных запросов организации может быть недостаточно стандартных средств настройки и администрирования системы. Также может понадобиться расширение функциональности АРМа клиента, разработка и подключение к нему новых модули с дополнительными возможностями, не входящих в состав ПК.

В данном руководстве описаны способы решения вышеперечисленных задач силами программистов-разработчиков организации, использующей ПК в качестве информационно-управляющей системы.

При разработке самого ПК «С-Платформа» использовались именно эти, изложенные ниже, интерфейсы и методики.

СОДЕРЖАНИЕ

Лист

1. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ ПРОГРАММЫ

1.1. Общие сведения

Программный комплекс «С-Платформа» предназначен для создания диспетчерских информационно-управляющих систем реального времени. Комплекс осуществляет сбор информации из разных источников, реализацию функции телеуправления, обработку принятых данных, их архивирование и представление пользователю информации на средствах отображения индивидуального и коллективного пользования. Также ПК «С-Платформа» обеспечивает решение ряда прикладных диспетчерских и технологических задач (включая расчетно-аналитические).

Дополнительным назначением комплекса является предоставление необходимых функций интегрирующей программной платформы для объединения ряда специализированных прикладных программных комплексов различных разработчиков в единую технологическую систему.

Комплекс может использоваться в качестве основы для построения автоматизированных систем управления технологическими процессами (АСУ ТП) и систем сбора и передачи информации (ССПИ) подстанций, систем оперативного технологического контроля и управления в ситуационных центрах, центрах обработки данных и центрах управления генерирующих и сетевых электроэнергетических компаний.

1.2. Решаемые задачи

Сервер приложений позволяет решать следующие задачи:

- резервирование серверов приложений в рамках группы и комплекса;
- запуск внешних (серверных) программ, контроль их работоспособности;
- прием и обработка информации;
- генерация событий и тревог;
- сохранение информации в базе реального времени и в долговременных архивах;
- обработка множественных подключений клиентов в режиме «клиент-сервер»;
- аутентификация пользователей;
- ведение централизованной базы с описанием объекта управления – нормативно-справочной информации (НСИ) и организация доступа к ней клиентских приложений;
- доступ к базе реального времени и долговременным архивам;

- выполнение технологических алгоритмов, написанных на встроенном языке технологического программирования;
- решение топологических задач с помощью процессора топологии;
- ведение файловой базы комплекса для хранения общих ресурсов (мнемосхем, документов, наборов ретроспективы и т.д.) и организация доступа к ней;

АРМ пользователя обеспечивает взаимодействие пользователя с сервером приложений и предоставляет ему функциональность в соответствии с требованиями решаемых задач.

Основными задачами являются:

- визуальное представление различных видов информации на экране пользователя АРМ. Информация может быть представлена в виде мнемосхем, отчетов, графиков, документов, в виде других специализированных форм и служит основой для принятия решений персоналом объекта;
- предоставление функций оперативного управления режимами работы и состоянием подчиненных объектов управления;
- администрирование нормативно-справочной информации (НСИ), настройка режимов и основных параметров работы системы;
- подготовка мнемосхем, отчетов, графиков и документов, обеспечивающих требуемое представление данных в соответствии со стандартами, принятыми в организации;
- администрирование внешнего вида и состава функциональных возможностей человеко-машинного интерфейса АРМ для различных категорий пользователей ПК.

Сервер ввода-вывода (СВВ) выполняет функции коммуникационного программного сервера или роутера данных в сети устройств разного уровня (контроллеры, устройства телемеханики, SCADA–системы, центральные приемо-передающие станции и т.д.).

СВВ обеспечивает:

- взаимодействие с другими системами и устройствами посредством последовательных каналов, IP (TCP или UDP) каналов по стандартным протоколам обмена;
- расширение поддерживаемой библиотеки протоколов обмена с помощью открытого API;

- одновременный и независимый прием данных от всех подключенных систем и устройств;
- одновременную и независимую передачу данных всем подключенным системам и устройствам;
- прием команд от вышестоящих систем и их передачу в заданные нижестоящие системы или устройства;
- промежуточную обработку данных;
- хранение последних изменений по каждому параметру.

Система анализа режимов сети представляет собой набор расчетных функций с возможностью расширения, сформированный в соответствии с требованиями конечного пользователя ПК.

Возможности САРС зависят от ее конфигурации:

1) в режиме онлайн (автоматизированные расчеты в темпе процесса):
   а) для магистральных сетей:
      - формирование текущей расчетной модели сети по данным ТМ;
      - расчет потокораспределения сети по данным телеметрии (оценка состояния);
      - расчет установившегося режима сети (режимная блокировка переключений);
      - оценка режимной надежности (на множестве возможных возмущений);
      - расчет токов КЗ;
      - расчет оптимального режима по реактивной мощности;
      - визуализация нарушенных ограничений рассчитанных режимных параметров на диспетчерской схеме и в оперативном журнале.

   б) для распределительных сетей:
      - формирование текущей расчетной модели сети по данным ТМ;
      - пофидерный расчет установившегося режима;
      - локализация мест возникновения неучтенных потерь.
2) в режиме офлайн (расчеты с использованием архивных данных):
   а) для магистральных сетей:
      - краткосрочный прогноз нагрузки (с использованием типовых суточных графиков);
      - расчет установившегося режима сети (режимная блокировка переключений);

- оценка режимной надежности (на множестве возможных возмущений);
- расчет токов КЗ;
- расчет оптимального режима по реактивной мощности;
- вариантные расчеты по расстановке средств компенсации реактивной мощности;
- расчет технических потерь электроэнергии и их структуры.

б) для распределительных сетей:

- пофидерный расчет установившегося режима;
- расчет технических потерь электроэнергии и их структуры;
- локализация мест возникновения неучтенных потерь;
- вариантные расчеты по расстановке средств компенсации реактивной мощности;
- выбор точек деления сети;
- расчет емкостных токов короткого замыкания;
- формирование профиля напряжения фидера.

3) общесистемные функции:

а) мониторинг уровней напряжения, загрузки оборудования;

б) определение режима максимально возможной нагрузки сети (утяжеление режима);

в) ввод режима в допустимую область по напряжению (положения отпаек трансформаторов);

г) визуализация результатов расчетов на расчетной схеме сети (включая динамическую раскраску элементов сети по заданным параметрам режима);

д) табличный процессор для визуализации параметров сети в табличном виде;

е) выбор исходного состояния расчетной модели сети из архива состояний сети;

ж) коррекция состояния расчетной модели сети;

з) сохранение анализируемого состояния расчетной модели сети в архиве состояний сети.

1.3. Сервер приложений

Сервер приложений предназначен для использования на серверных и персональных компьютерах, работающих под управлением операционных систем:

- Операционная система «Альт 8 СП» (и выше), разработки компании ООО «Базальт СПО».

- Операционная система специального назначения «Astra Linux Special Edition» РУСБ.10015-01 (очередное обновление 1.6 и выше), разработки компании ООО «РусБИТех-Астра».
- Операционная система специального назначения «Astra Linux Special Edition» РУСБ.10152-02 (очередное обновление 4.7 и выше), разработки компании ООО «РусБИТех-Астра».
- Операционная система специального назначения «Astra Linux Special Edition» РУСБ.10265-01 (очередное обновление 8.1 и выше), разработки компании ООО «РусБИТех-Астра».
- Microsoft Windows.

ПО сервера приложений предъявляет следующие минимальные аппаратные требования к ПЭВМ:

- Процессор:
  - Центральные процессоры с архитектурой «Эльбрус» (E2K): «Эльбрус-4С», «Эльбрус-8С» и «Эльбрус-8СВ», разработанные компанией АО «МЦСТ»;
  - Центральные процессоры с архитектурой ARMv8-A: «Baikal-M», разработанный компанией АО «Байкал Электроникс»;
  - Центральные процессоры с архитектурой x86 (Intel 80x86) и x86-64 (AMD64/Intel64/EM64T).
- Оперативная память — 8 ГБ.
- Дисковое пространство — 500 ГБ (в зависимости от глубины архивов данных и от их количества, объем необходимого дискового пространства может изменяться).
- Скорость канала связи по Ethernet — не менее 10 Мбит/сек.

Для установки и настройки сервера приложений ПК, а также для использования компонента «Монитор ПК», имеющего графический пользовательский интерфейс, требуются:

- Экран размером не менее 1024 x 768 пикселей
- Наличие манипулятора «мышь» с интерфейсом PS/2 или USB
- Наличие 101/102-кнопочной клавиатуры с русской и английской раскладкой

Сервер приложений использует для взаимодействия с SQL-базами механизм ADO или ODBC, а также для взаимодействия с базами формата mdb (MS Access) компонент MS Jet 4.0. Отсутствие вышеназванных механизмов в ОС не позволит работать серверу приложений.

Сервер приложений функционирует в режиме постоянного взаимодействия с другими серверами приложений в составе комплекса, серверами ввода-вывода и АРМ пользователей.

Поэтому для его корректной работы важно наличие надежного и быстрого канала связи. Скорость канала меньше 10 Мбит/сек может приводить к задержкам в работе ПО.

Системная служба (ScdService.exe) – это специальная программа, оформленная как системная служба WINDOWS. Начиная выполняться в момент старта ОС, служба позволяет запустить сервер приложений без вмешательства оператора, непосредственно после включения компьютера.

Другими функциями службы являются:

- контроль работоспособности, перезапуск сервера приложений в случае сбоя или аварийной остановки процесса;
- информационный обмен с программой «Монитора ПК»: передача данных о серверной конфигурации, активности серверов и серверных программ, содержания log-файлов, характеристик компьютеров и пр.
- замена серверного ini-файла на файл, полученный от «Монитора ПК».

1.4. АРМ пользователя

АРМ пользователя предназначен для использования на персональных компьютерах, работающих под управлением операционной системы Microsoft Windows.

Для ПО АРМ предъявляются следующие минимальные аппаратные требования к ПЭВМ:

- Процессор — Intel Core i5.
- Оперативная память — 8 ГБ.
- Дисковое пространство — 1 ГБ.
- Экран размером не менее 1280 x 1024 пикселей.
- Наличие манипулятора «мышь» с интерфейсом PS/2 или USB.
- Наличие 101/102-кнопочной клавиатуры с русской и латинской раскладкой.
- Скорость канала связи по Ethernet с сервером приложений — 10 Мбит/сек.

ПО АРМ пользователя функционирует в режиме постоянного взаимодействия с сервером приложений. Поэтому для его корректной работы важно наличие надежного и быстрого канала связи. Скорость канала меньше 10 Мбит/сек может приводить временным к задержкам в работе программы. Потеря связи приводит к останову и закрытию программы.

Для подготовки мнемонических схем используется программный инструментарий фирмы «МОДУС». Поэтому для выполнения работ, связанных с созданием, редактированием и настройкой схем, на компьютере должен быть установлен редактор «Модус» версии 5.20.

1.5. Сервер ввода-вывода

Сервер ввода-вывода (СВВ) предназначен для использования на промышленных или персональных компьютерах, работающих под управлением операционных систем:

- Операционная система «Альт 8 СП» (и выше), разработки компании ООО «Базальт СПО».
- Операционная система специального назначения «Astra Linux Special Edition» РУСБ.10015-01 (очередное обновление 1.6 и выше), разработки компании ООО «РусБИТех-Астра».
- Операционная система специального назначения «Astra Linux Special Edition» РУСБ.10152-02 (очередное обновление 4.7 и выше), разработки компании ООО «РусБИТех-Астра».
- Операционная система специального назначения «Astra Linux Special Edition» РУСБ.10265-01 (очередное обновление 8.1 и выше), разработки компании ООО «РусБИТех-Астра».
- Microsoft Windows.

Для ПО сервера ввода-вывода предъявляются следующие минимальные аппаратные требования к ПЭВМ:

- Процессор:
  - Центральные процессоры с архитектурой «Эльбрус» (E2K): «Эльбрус-4С», «Эльбрус-8С» и «Эльбрус-8СВ», разработанные компанией АО «МЦСТ»;
  - Центральные процессоры с архитектурой ARMv8-A: «Baikal-M», разработанный компанией АО «Байкал Электроникс»;
  - Центральные процессоры с архитектурой x86 (Intel 80x86) и x86-64 (AMD64/Intel64/EM64T).
- Оперативная память — 8 ГБ.
- Дисковое пространство — 1 ГБ.
- Скорость канала связи Ethernet с сервером приложений (при использовании в составе ПК) — 10 Мбит/сек.
- Наличие последовательного интерфейса (COM-порта) для устройств, использующих для связи последовательный интерфейс.

1.6. САРС

САРС может работать на серверных и персональных компьютерах, работающих под управлением операционных систем:

- Операционная система «Альт 8 СП» (и выше), разработки компании ООО «Базальт СПО».
- Операционная система специального назначения «Astra Linux Special Edition» РУСБ.10015-01 (очередное обновление 1.6 и выше), разработки компании ООО «РусБИТех-Астра».
- Операционная система специального назначения «Astra Linux Special Edition» РУСБ.10152-02 (очередное обновление 4.7 и выше), разработки компании ООО «РусБИТех-Астра».
- Операционная система специального назначения «Astra Linux Special Edition» РУСБ.10265-01 (очередное обновление 8.1 и выше), разработки компании ООО «РусБИТех-Астра».
- Microsoft Windows.

ПО САРС, работающее в режиме online, предъявляет следующие минимальные аппаратные требования к ПЭВМ:

- Процессор:
  - Центральные процессоры с архитектурой «Эльбрус» (E2K): «Эльбрус-4С», «Эльбрус-8С» и «Эльбрус-8СВ», разработанные компанией АО «МЦСТ»;
  - Центральные процессоры с архитектурой ARMv8-A: «Baikal-M», разработанный компанией АО «Байкал Электроникс»;
  - Центральные процессоры с архитектурой x86 (Intel 80x86) и x86-64 (AMD64/Intel64/EM64T).
- Оперативная память — 8 ГБ.
- Дисковое пространство — 1 ГБ.
- Скорость канала связи Ethernet — не менее 10 Мбит/сек.

Для работы АРМ аналитика-режимщика предъявляются следующие минимальные аппаратные требования к ПЭВМ:

- Процессор:
  - Центральные процессоры с архитектурой «Эльбрус» (E2K): «Эльбрус-4С», «Эльбрус-8С» и «Эльбрус-8СВ», разработанные компанией АО «МЦСТ»;
  - Центральные процессоры с архитектурой ARMv8-A: «Baikal-M», разработанный компанией АО «Байкал Электроникс»;
  - Центральные процессоры с архитектурой x86 (Intel 80x86) и x86-64 (AMD64/Intel64/EM64T).
- Оперативная память — 8 ГБ.
- Дисковое пространство — 1 ГБ.

- Экран размером не менее 1280 x 1024 пикселей.
- Скорость канала связи Ethernet — не менее 10 Мбит/сек.
- Наличие манипулятора «мышь» с интерфейсом PS/2 или USB.
- Наличие 101/102-кнопочной клавиатуры с русской и латинской раскладкой.

2. ХАРАКТЕРИСТИКА ПРОГРАММЫ

2.1. Режим работы ПК «С-Платформа»

ПК «С-Платформа» предназначен для работы в круглосуточном режиме в условиях отсутствия постоянного дежурного персонала.

2.2. Средства контроля правильности выполнения программного обеспечения ПК

Для обеспечения правильности выполнения программного обеспечения ПК реализована двухуровневая схема диагностики.

В первую очередь в ПО сервера приложений ПК встроена процедура диагностики, которая с циклом 10 минут проверяет работоспособность основных подсистем сервера приложений:

1. Работа с базой данных НСИ ПК.
2. Доступность соединений TCP/IP.
3. Контроль целостности базы данных реального времени.

В случае ошибок инициируется процедура перезапуска сервера приложений ПК.

Также сервер приложений ПК контролирует работоспособность программ-клиентов. Если какой-либо из процессов аварийно завершается, сервер ПК его снова запускает.

Во вторую очередь работоспособность сервера ПК контролируется службой серверов приложений, которая оформлена в виде службы. Основными функциями этой программы являются:

1. Запуск сервера ПК в момент старта ОС до входа пользователя в систему.
2. Запуск и останов сервера по команде программы «Монитор приложений».
3. Контроль процесса ОС, выполняющего программу сервера приложений ПК. Если этот процесс завершается, то служба снова его запускает.

3. ОБРАЩЕНИЕ К ПРОГРАММЕ

3.1. Написание клиентского приложения. Протокол прикладного уровня MDX

Протокол прикладного уровня MDX (Message Data Exchange) обеспечивает соединение клиента и сервера приложений (далее –сервер) поверх сетевого протокола TCP/IP. Со стороны клиента предоставляются следующие возможности:

1. подключение к серверу (создание сессии);
2. управление данными;
3. управление событиями.

Для подключения к серверу требуется указать сетевое имя компьютера, на котором сервер расположен, имя и пароль пользователя, зарегистрированные в системе ПК.

Управление данными включает:

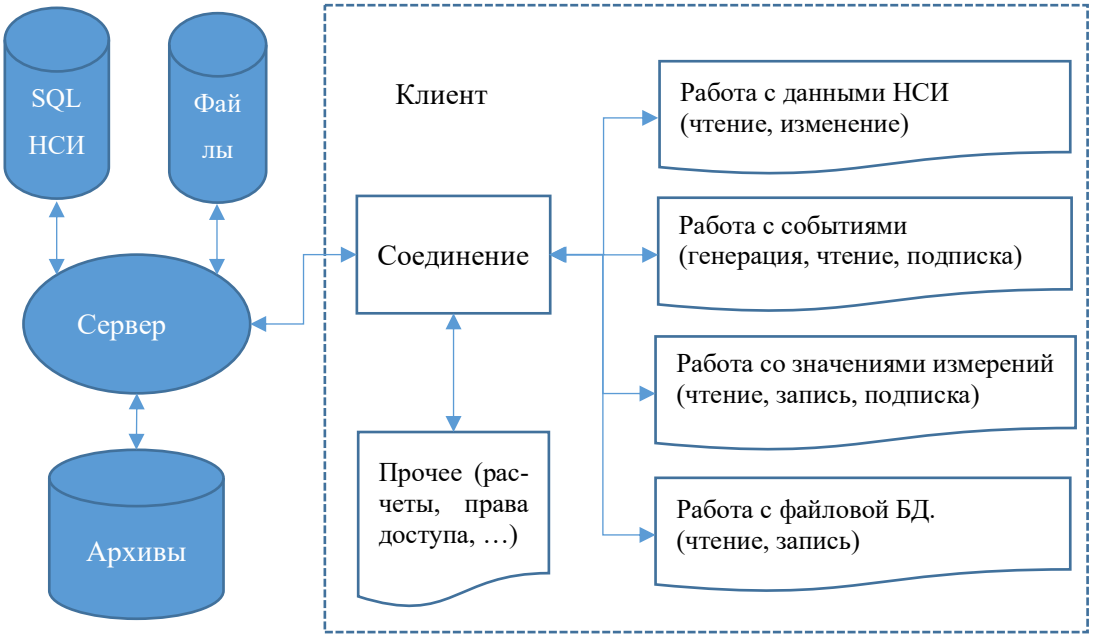
- получение результатов SQL-запросов к данным НСИ;
- модификацию записей таблиц НСИ;
- чтение-запись архивных данных;
- выполнение специальных процедур, встроенных в сервер.

Управление событиями: - оповещение клиента о возникновении интересующего его события. Клиент может сформировать для себя требуемое подмножество событий, указав соответствующие энергообъекты и коды событий:

3.1.1. Использование DLL-интерфейса (ScdMDX.dll)

3.1.1.1. Обзор

Весь базовый механизм протокола MDX реализован в файле ScdMdx.dll. Для работы с сервером, требуется установить соединение и использовать интерфейс запросов, поддерживаемый сервером.



.

Рисунок 3.1 - Архитектура клиента ПК

Процедуры обработки событий связываются с объектом при его создании.

Основные интерфейсы для работы с библиотекой ScdMdx.dll описаны в файлах:

- Mdsock.h – механизм взаимодействия по TCP/IP;
- Mdrec.h – записи с полями. Элементарная единица данных;
- MdRecArr.h – таблицы в памяти. Массивы упорядоченных записей;
- Mdx.h – базовый механизм формирования протоколов прикладного уровня, подключение;
- ScdMdx.h – интерфейс серверных запросов;

Интерфейсы доступа из Delphi описаны в файле – Mdx.pas.

Ниже последовательно описаны все этапы работы с сервером ОИК, необходимые для написания клиентского приложения с использованием ScdMdx.dll. Примеры приводятся на языке С. Сам проект находится в директории Scd\Client\Examples\Vc\WriteHVed и реализует алгоритм записи архивов часовой ведомости. Файлы mc_ses.cpp, mc_db.cpp и mc_evn.cpp могут служить шаблонами для написания своего клиента ПК.

3.1.1.2. Подключение к серверу

Первоначальный этап работы с библиотекой ScdMdx.dll – инициализация внутренних структур библиотеки. В качестве параметра функции требуется задать максимальное количество одновременно открытых сессий данного приложения с серверами.

```
#include "Mdx.h"
```

```
MDXerror err;

err = MDXInit();

If (err) return; // Ошибка, если Err != 0;
```

Следующий этап – подключение к серверу. Сразу после открытия сессии, имеет смысл создать программные объекты (не путать с OLE-объектами) для работы с данными и сообщениями.

```
#include "Mdx.h"

MDXSes SesId;

MDXObj ObjDB, ObjEvn;

// Создание сессии и подключение

MDXCliSes *ses = nullptr;

If (MDXCliCreate(&ses) != MDX_RESERR_OK) return;

If (MDXCliOpen(ses, “localhost”, 1312, “admin”, “”) != MDX_RESERR_OK)

 return;
```

После выполнения требуемых действий заключительной этап – разрыв связи с сервером и освобождение дескриптора сессии.

```
#include "Mdx.h"

MDSesClose(ses);

MDXCliFree(ses);

ses = nullptr;
```

К базовым функциям интерфейса подключения и работы с сервером относятся:

```
MDXCallErr MDXCliRequest(       // Запрос (получение результата):
  MDXCliSes *Ses,               // дескриптор сессии,
  const char *Name,             // имя процедуры,
  MDXCliParam **Prm,            // (out) параметры запроса,
  MDXCliParam **Res=nullptr);   // (in) адрес для приема результата

MDXCallErr MDXCliMessage(       // Сообщение (действие):
  MDXCliSes *Ses,               // дескриптор сессии,
  const char *Name,             // имя сообщения,
  MDXCliParam **Prm);           // (out) параметры сообщения

MDXCallErr MDXCliPost(          // Завершение формирования действия:
  MDXCliSes *Ses,               // дескриптор сессии,
  MDXCliParam *Prm);            // параметры
```

```
MDXCallErr MDXCliNameExist(     // Проверка существования имени:
  MDXCliSes *Ses,               // дескриптор сессии,
  const char *Name,             // имя процедуры или сообщения,
  int *Type);                   // номер в списке (или -1 если отсутствует)

MDXCallErr MDXCliTypeName(      // Проверка существования имени:
  MDXCliSes *Ses,               // дескриптор сессии,
  int Type,                     // номер в списке,
  char *TypeName);              // имя типа (MDX_NAME_MAX);
                                // Управление транзакциями БД
MDXCallErr MDXCliBlockBeg(MDXCliSes *Ses);
MDXCallErr MDXCliBlockEnd(MDXCliSes *Ses, bool Commit=true, MDXCliParam **Res=nullptr);
MDXCallErr MDXCliBlockNone(MDXCliSes *Ses);

MDXCallErr MDXCliWaitResult(    // Ожидание результата процедуры
  MDXCliSes *Ses,               // дескриптор сессии,
  MDXCliParam **Res,            // адрес для приема результата,
  int Timeout=INFINITE);        // тайм-аут ожидания (ms)

MDXCallErr MDXCliGetMessage(    // Получение сообщения из очереди:
  MDXCliSes *Ses,               // дескриптор сессии,
  MDXCliParam **Prm,            // параметры сообщения,
  int Timeout=INFINITE);        // тайм-аут ожидания (ms)
```

Представленные выше функции обеспечивают работу с расширенным интерфейсом запросов, предоставляемым сервером. Этот интерфейс, в нотации языка c++, на данный момент обеспечивает следующие функции:

```
class MDXService
{
public:
                            // Сессия
  inline bool Active() { return (m_ses->Idn!=0); }
  bool Open(const char *SrvAddr, const char *Login, const char *Password, unsigned
PortNo=PORT_MDX_NUM);
  void Close();
                            // Пользователь (свойства, группы, права)
  bool UserDescription(MDXCliParam **Res);
                            // Подписка на события и изменение параметров
  bool SubEvent(const MDRInt *Items=NULL, int Count=0);
  bool SubParam(const MDRInt *Items=NULL, int Count=0);
                            // Работа с событиями
  bool EventNew(MDRDouble Time, MDRDouble Dtcp, MDRInt Code, const char *Obj, const char
*Caption,
                const char *Comment=NULL, const MDRByte *Data=NULL, MDRInt DataLen=0);
  bool EventKwit(const char *Key, MDRInt Code);
  bool EventRequest(const int *Code, int CodeNum, double DTH, double DTL, bool Kwit,
MDXCliParam **Res);
  bool EventRequest(const char *Sql, MDXCliParam **Res);
                           // Пердача значений параметров (тэгов, по ключу).
  bool TagWrite(const char *Tag, MDRDouble Time, MDRByte Value, MDRInt Flags, MDRDouble
TimeDTCP);
  bool TagWrite(const char *Tag, MDRDouble Time, MDRInt Value, MDRInt Flags, MDRDouble
TimeDTCP);
  bool TagWrite(const char *Tag, MDRDouble Time, MDRDouble Value, MDRInt Flags, MDRDouble
TimeDTCP);
  bool TagWrite(const char *Tag, MDRDouble Time, const MDRByte *Value, MDRInt Len, MDRInt
Flags, MDRDouble TimeDTCP);
  bool TagWrite(const MDRByte *Value, MDRInt Len);
                            // Запись измерений (по идентификатору)
  bool MeasWrite(MDRInt Id, MDRDouble Time, MDRByte Value, MDRInt Flags, MDRDouble
TimeDTCP);
```

```
  bool MeasWrite(MDRInt Id, MDRDouble Time, MDRInt Value, MDRInt Flags, MDRDouble
TimeDTCP);
  bool MeasWrite(MDRInt Id, MDRDouble Time, MDRDouble Value, MDRInt Flags, MDRDouble
TimeDTCP);
  bool MeasWrite(MDRInt Id, MDRDouble Time, const MDRByte *Value, MDRInt Len, MDRInt
Flags, MDRDouble TimeDTCP);
  bool MeasWrite(const MDRByte *Value, MDRInt Len, MDXCliParam **Res = nullptr);
                            // Чтение измерений (по идентификатору)
  bool MeasRead(MDRInt Id, MDRDouble Time, MDXCliParam **Res);
  bool MeasRead(MDRInt Id, MDRDouble TimeLow, MDRDouble TimeHigh, MDXCliParam **Res);
  bool MeasRead(const MDRByte *ReqBuf, MDRInt Len, MDXCliParam **Res);

  bool ManualWrite( MDRInt  ID, MDRDouble DT, MDRInt FLG, MDRDouble DTCP, MDRByte vByte );
                            // Работа с БД
  bool SqlRequest(const char *Sql, MDXCliParam **Res);
  bool SqlUpdate(const char *TblName, MDArray *Arr, MDXCliParam **Res = nullptr);
  bool SqlExec(const char *Sql);
                            // Работа с файлами (файловая БД)
  bool FileList(const char *Folder, const char *FileName, MDXCliParam **Res);
  bool FileGet(const char *Folder, const char *FileName, MDXCliParam **Res);
  bool FilePut(const char *Folder, const char *FileName, MDRByte *Data, MDRInt Len);
  bool FileDel(const char *Folder, const char *FileName);
                            // Работа с тренажером
  bool IcSave(const char *Folder, const char *FileName, const char *DataTime, MDXCliParam
**Res);
  bool IcLoad(const char *Folder, const char *FileName, int iModelTime, int iNumIC,
MDXCliParam **Res);
  bool ChangeStatus( int iStatus, MDXCliParam **Res);
                            // Базовые операции обмена данными с сервером
  MDXCliParam *Request(const char *Name, MDXCliParam **Res=NULL);
  MDXCliParam *Message(const char *Name);
  void Post(MDXCliParam *Prm);

  void BlockBeg();
  void BlockEnd(bool Commit=true, MDXCliParam **Res=NULL);

  bool WaitResult(MDXCliParam **Res, int Timeout=INFINITE);

  inline int MsgCount() { return m_ses->MsgCount; }
  MDXCliParam *GetMessage(int Timeout=INFINITE);

  void PrmFree(MDXCliParam *Prm);
                            // Имена типов записей протокола
  int NameExist(const char *Name);
  char *TypeName( int Type );

  void GetSrvVersion();     // Получение версии сервера
                            // Запрос с ожиданием завершения операции
  MDArray *DoSqlRequest(const char *Sql);
  void *DoMeasRequest(const MeasValRead &Req, int *reslen);
};
```

3.1.1.3. Элементарные типы данных

В пользовательских функциях и структурах MDX используются следующие базовые типы данных (см. mdrec.h):

№	Тип	Описание
1	mdr_Tbool	Логический (True, False)
2	mdr_Tbyte	Байт, без знаковое целое (0..255)
3	mdr_Tsmall	Целое, 2 байта (-32,768..32,767)

4	mdr_Tint	Целое, 4 байта (-2,147,483,648..2,147,483,647)
5	mdr_Tfloat	Вещественное, 4 байта (3.4E +/- 38 (7 digits))
6	mdr_Tdouble	Вещественное, 8 байт (1.7E +/- 308 (15 digits))
7	mdr_Tchar	Текст фиксированной длины (1..255 символов)
8	mdr_Tmemo	Текст произвольной длины
9	mdr_TunixDT	Время в UNIX формате, Sec, 4 байта (mm-dd-yyyy hh:mm:ss)
10	mdr_TUnixMDT	Время в UNIX формате, Sec и Msec, 8 байт (mm-dd-yyyy hh:mm:ss.ms)

В mdrec.h файле основные типы описаны как:

```
typedef   bool          MDRBool;
typedef   char          MDRByte;
typedef   short         MDRSmall;
typedef   long          MDRInt;
typedef   float         MDRFloat;
typedef   double        MDRDouble;
typedef   long          MDRUnixDT;
typedef   double        MDRUnixMDT;
typedef   char*         MDRBlob;
typedef   char*         MDRMemo;
typedef   char*         MDRChar;
```

Соответственно универсальный тип данных и описание поля представлены как:

```
union MDRVar

{

   MDRBool          vBool;

   MDRByte          vByte;

   MDRSmall         vSmall;

   MDRInt           vInt;

   MDRFloat         vFloat;

   MDRDouble        vDouble;

   MDRText          vText;

   MDRUnixDT        vUnixDT;

   MDRUnixMDT       vUnixMDT;

;

struct MDRFieldDef

{

    char FldName[MDR_MAX_NAME + 1];/* Наименование поля */
```

```
    int FldSize;                                                    /*
Размер поля */
    MDRFldType FldType;                                  /* Тип поля */
};
```

И, наконец, описание структуры, которая используется при доступе к полям записи в обработчиках событий MDX, выглядит следующим образом:

```
/* Вариантное поле */
struct MDRFieldVal
{
    MDRFieldDef Fld;
    MDRVar Val;
    bool IsNull;
};
```

Данные, описываемые с помощью этой структуры, предназначены только для чтения и изменению не подлежат.

3.1.1.4. Управление данными

3.1.1.4.1. Таблицы в памяти

Таблицы в памяти представляют собой упорядоченный массив записей одинаковой структуры (см. mdarray.h). Понятие структуры и функции доступа к полям записи описаны в файле MdRec.h.

Типы полей соответствуют основным типам данных MDX.

В одной записи может быть до 255 полей. Доступ к полям записи осуществляется в два этапа:

1. получение описателя поля с помощью функций MDRTypFld или MDRTypFldN;
2. чтение или изменение значения поля функциями MDRGet*, MDRSet*;

В файле MdRecArr.h описаны функции и структуры для работы с таблицей в памяти.

Таблицу можно создать и удалить функциями MDACreate и MDADestroy. При этом само описание таблицы будет размещено в структуре MDArray, передаваемой в качестве параметра.

Формирование структуры полей производится с помощью MDAFldAdd. Данная функция может быть вызвана несколько раз подряд. Новые поля будут добавлены к списку уже существующих.

Ключи (Key) или индексы – это списки, различным образом упорядочивающие записи таблицы. Для одной таблицы можно задать до 16 ключей и каждый ключ может содержать до 6 полей таблицы включительно. Добавление или удаление ключей осуществляется функциями MDAKeyAdd и MDAKeyDel динамически. Т.е. добавление ключа формирует новый упорядоченный список на записях таблицы. Удаление – уничтожает список, но не записи таблицы. При добавлении записей в таблицу, для которой ключи не определены, автоматически создается ключ с пустым списком полей.

Функции MDARecNew, MDARecEdit создают или копируют запись для редактирования. После изменения полей MDARecPost сохранит изменения в таблице и перестроит индексы (если они не были созданы с признаком idxNonMaintained). Функция MDARecDel удаляет запись из таблицы, или отменяет изменения, если она вызывается для дескрипторов, полученных с помощью MDARecNew, MDARecEdit.

Порядковый доступ к записи в ключевом списке возможен с помощью функции MDARec, а. поиск по ключу – MDARecSeek.

3.1.2. ИСПОЛЬЗОВАНИЕ OLE-ИНТЕРФЕСА (Scdsys.ocx)

3.1.2.1. Обзор

Библиотека Scdsys.ocx в своей основе является OLE-надстройкой над технологиями ScdMdx.dll, которые были описаны в предыдущем разделе. Она предоставляет пользователю ряд интерфейсов, с помощью которых можно легко обеспечить взаимосвязь с сервером ОИК, используя стандартный для Windows механизм COM- и ActiveX-объектов. Приведем краткое описание объектов Scdsys.ocx.

№	Объект	Интерфейсы	Описание
1	ScadaCli	IScadaCli IscadaCliEvents	Канал связи с сервером. Дополнительно обеспечивает ряд вспомогательных свойств и методов, способствующих упрощению взаимодействия: предоставляет характеристики пользователя и главного энергообъекта, упрощает работу с системообразующими таблицами объектов (T_OBJ) и полей (T_FLD), правами доступа, синхронизирует время сервера и клиента.

2	ScadaAbo	IScadaAbo IScadaAboCmd IScadaAboEvents	Абонент. К каждому каналу, в рамках одной программы, может быть подключено несколько таких объектов. Именно через них происходит обмен данными и событиями с сервером. Каждый абонент обеспечивает абстракцию независимого, монопольного использования канала.
3	MDArray MDARec	IMDArray IMDARec	Виртуальные таблицы и записи в памяти. Помимо важного самостоятельного значения, таблицы могут применяться для эффективного приема больших массивов данных с сервера.
4	ModObj	IScadaObj IScadaObjEvents	Прообраз объектов, используемых в оболочке АРМ - ScdArm.exe. Все объекты, которые предназначены для работы в среде АРМа, должны быть ActiveX- объектами, реализующими интерфейс IScadaObj (или его Dispatch – вариант) и обработчики событий из IScadaObjEvents.
5	ScadaMenu	IScadaMenu	С помощью этого объекта оболочка ScdArm.exe узнает о содержании меню IscadaObj-объекта
6	DataBag	IDataBag	Объект, реализующий абстракцию динамической, древовидной структуры данных. Используется для хранения сложных параметров в полях БД.
7	Scada-Tim	IscadaTim IScadaTimEvents	Генератор событий таймера. Вспомогательный объект. Практическая ценность неочевидна.

Все примеры в данном разделе представлены на Delphi.

3.1.2.2. Канал связи с сервером (ScadaCli)

3.1.2.2.1. Интерфейс IScadaForm

Данный интерфейс является базовым для всех ActiveX-интерфейсов, определенных в ScdSys.ocx. В нем собраны свойства объекта, отвечающие за его представление как формы на экране.

Свойство	Назначение	Примечания
Visible	Видимость	Boolean. Показывает или устанавливает, является ли форма (объект) видимой
Caption	Заголовок	String. Заголовок формы (объекта)
Color	Цвет	OLE_COLOR. Основной цвет формы (объекта)
Font	Шрифт	IFontDisp. Основной шрифт формы (объекта)
Active	Наличие фокуса	Boolean. Показывает, имеет ли форма (объект) фокус в данный момент
HelpFile	Файл помощи	String. Имя файла помощи
Enabled	Разрешение обработки	Boolean. Показывает или устанавливает, реагирует ли объект на события мыши, клавиатуры и таймера

3.1.2.2.2. Интерфейс IscadaCli

Метод	Назначение	Примечания
Open	Соединение с сервером	Перед вызовом метода предварительно должны быть заданы значения свойств SrvAddress, UserName и UserPassword. В случае успешного подключения свойство CliActive будет установлено в TRUE. Если соединения не произошло, то свойство NoUser покажет причину: неправильный адрес или характеристики пользователя.
Close	Закрытие соединения	Перед закрытием сессии, например, при сохранении локальных настроек, имеет смысл отключить режим оповещения по событиям. Это делается с помощью метода CloseEvent.
TimeSynchro	Синхронизация времени с сервером	Синхронизация всегда осуществляется неявно в момент создания сессии. Сразу после этого свойства TimeOle и TimeSec возвращают актуальное текущее время сервера соответственно в OLE- и UNIX(sec)-форматах. Преобразование между форматами можно осуществить с помощью свойств TimeOleToSec и TimeSecToOle.
ObjNameFromId ObjIdFromName	Работа с T_OBJ.	Вспомогательные функции. Определение имени объекта (поле OBJ_NAME_LAT) в таблице T_OBJ по его идентификатору (поле OBJ_ID) и наоборот. При этом в качестве дополнительной информации возвращается тип объекта (поле OBJ_OBJ_T_ID)
Lock Unlock	Управление блокировкой	Lock-Unlock вложенные директивы. При первом вызове Lock генерируется событие объекта OnLock, при последнем Unlock – событие OnUnlock. На внутреннем уровне Lock-Unlock вызываются автоматически при собственных операциях ScadaCli с сервером и обработке пользовательских запросов данных (BlockBegin-BlockEnd) выполненных с ожиданием результата.
CloseEvent	Запрет обработки сообщений	Вспомогательная функция. Может быть использована, например, перед закрытием сессии, до сохранения локальных параметров
ChangePassword	Изменение пароля	Изменяет значение пароля пользователя, требуемого при входе в систему
DataBagGet DataBagPut	Работа с таблицей T_DATA_BAG	Таблица T_DATA_BAG представляет собой хранилище произвольных данных, куда каждый может складывать все, что считает нужным. В частности, именно в ней хранятся локальные параметры и настройки объектов. Методы DataBagGet и DataBagPut автоматизируют работу с таблицей. Параметры: PrmName: string – имя параметра; IsLocal: Boolean – личный(True) или общий параметр; Value: IDataBag - Данные
Reconnect	Оптимальное подключение	При работе в системе с одним или несколькими резервными серверами, возможна балансировка нагрузки, путем равномерного распределения клиентов между ними. Метод Reconnect прозрачным образом переключает (или оставляет) клиента к оптимальному серверу. TRUE, возвращаемое методом, говорит о действительном переключении на другой резервный сервер.

Свойство	Назначение	Примечания
TimeOle TimeSec	Текущее время сервера	Текущее время сервера с точностью до sec, соответственно в OLE(VARIANT) и UNIX (long, sec) форматах.
UserPrm	Параметры пользователя	Возвращает VARIANT-значение указанного параметра для пользователя, открывшего сессию. Возможные значения параметров описаны в EUserField: ufId – идентификатор в таблице T_USRS; ufName1, ufName2, ufName3 – соответственно поля USRS_FIRST_NAME, USRS_MIDL_NAME и USRS_LAST_NAME таблицы T_USRS; ufFIO – скомпилированное имя с инициалами; ufOrg – организация (USRS_ORG); ufDep – отдел (USRS_DEP); ufPhone – телефон (USRS_PHONE); ufTab – табельный номер (USRS_TAB).
UserGrPrm	Параметры группы пользователя	Возвращает VARIANT-значение указанного параметра для группы пользователя, открывшего сессию. Возможные значения параметров описаны в EuserGrField: gfId– идентификатор в таблице T_USRGR; gfName – имя группы (USRGR_NAME); gfComm – комментарии (USRGR_COMM); gfAccess – зарезервировано.
EnergyObj	Энергообъект, владелец БД	Идентификатор энергообъекта, прописанный в таблице T_THIS
HWnd	Описатель окна	Описатель окна объекта. Используется для обмена сообщениями и подключения объектов-абонентов
AboCount	Счетчик абонентов	Количество объектов-абонентов, связанное с данным объектом ScadaCli
ObjCount ObjByIndex ObjByValue ObjIndex	Работа с таблицей T_OBJ	С помощью данных свойств можно узнать количество записей в таблице T_OBJ, получить информацию об объекте по его порядковому номеру в таблице (ObjByIndex), имени или идентификатору (ObjByValue) и, наконец, определить индекс объекта в списке по его имени или идентификатору (ObjIndex). Информация об объекте возвращается в VARIANT-массиве, элементы которого будут содержать (EObjFields): CF_Obj_Id(0) – идентификатор (OBJ_ID); CF_Obj_NameLat(1) – латинское имя (OBJ_NAME_LAT); CF_Obj_Name(2) – полное наименование (OBJ_NAME); CF_Obj_Type(3) – тип (OBJ_OBJ_T_ID); CF_Obj_ACXTBL(4) – ActiveX для таблиц (OBJ_ACXTBL); CF_Obj_ACXREC(5) – ActiveX для записей (OBJ_ACXREC); CF_Obj_ICO(6) – иконка (OBJ_ICO); CF_Obj_Hide (7) – признак скрытности (OBJ_HIDE).
FldCount FldByIndex	Работа с таблицей T_FLD	С помощью данных свойств можно узнать количество записей в таблице T_FLD, получить информацию о поле по его порядковому номеру в списке (FldByIndex),

FldByValue FldByObj		имени или идентификатору (FldByValue) и, наконец, получить полный список полей для заданного объекта (FldByObj). Информация о поле объекта возвращается в VARIANT-массиве, элементы которого будут содержать (EfieldFields): CF_Fld_Id (0) – идентификатор (FLD_ID); CF_Fld_Obj_Ref(1) – идентификатор объекта (FLD_OBJ_ID); CF_Fld_NameLat(2) – латинское имя (FLD_NAME_LAT); CF_Fld_Name(3) – наименование (FLD_NAME); CF_Fld_Type_Ref(4) – тип (FLD_FLD_T_ID); CF_Fld_Flags(5) – флаги (FLD_FLAGS); CF_Fld_Size(6) – размер (FLD_SIZE); CF_Fld_Prm(7) – параметры (FLD_PRM). При получении полного списка полей объекта с помощью FldByObj, возвращаемое VARIANT-значение представляет собой массив массивов парметров полей, по количеству полей объекта.
CliActive	Признак открытой сессии	Boolean. Сессия считается активной, если открыт канал и завершена предварительная загрузка данных объектом ScadaCli. Изменяя значение CliActive, можно открыть или закрыть сессию, аналогично использованию методов Open и Close.
SrvAddress	Адрес сервера	String. Полный сетевой адрес серверной ЭВМ или строка IP-адреса. Если задать список из 2-х и более адресов, разделенных запятыми, то попытки подключения будут производится последовательно, слева направо, до первого успешного соединения.
UserName	Имя пользователя	String. Имя пользователя при регистрации в системе (поле USRS_NAME в таблице T_USRS).
UserPassword	Пароль	String. Пароль пользователя при регистрации в системе (поле USRS_PASS в таблице T_USRS).
NoUser	Причина неудачного подключения	Boolean. При неудачном открытии сессии, TRUE – означает неправильные имя или пароль, FALSE – имя сервера.
TimeOleToSec TimeSecToOle	Преобразование OLE- и UNIX-времени	Взаимное преобразование OLE-времени в UNIX- формат (TimeOleToSec) и, обратно, времени в секундах в OLE-время (TimeSecToOle)
DataBagId	Идентификатор записи в T_DATA_BAG	Long. Это свойство дополняет набор методов для работы с таблицей параметров - T_DATA_BAG. По имени и признаку локальный(TRUE)-общий(FALSE), ищется запись и возвращается ее идентификатор. Если запись не найдена, возвращается нулевое значение.
WorkDir	Рабочий директорий	String. Имя рабочего директория. Чисто информационное свойство. Внутри ScadaCli не используется.
DragData	Хранилище данных при Drag-Drop –операциях	IDataBag. Помимо стандартных OLE-механизмов, используемых в DragDrop-операциях, можно использовать данное свойство как промежуточное хранилище данных. Внутри ScadaCli не используется.

ClipData	Хранилище данных при операциях с буфером обмена	IDataBag. Помимо стандартных OLE-механизмов, используемых при операциях с буфером обмена, можно использовать данное свойство как промежуточное хранилище данных. Внутри ScadaCli не используется.
RightObj RightProc	Права доступа	Int. Возвращает значения прав доступа для данного пользователя соответственно из таблиц T_OBJ_R и T_PROC_R. Объекты и процедуры могут быть указаны как по имени, так и по идентификатору в таблице. Биты прав доступа для объекта следующие (EobjRights): 0 – orRead, 1 – orWrite, 2 – orNew, 3 – orDel;
Recon-nectAuto	Автоматическое переподключение	Boolean. Если данное свойство установлено в TRUE, то при аварийном разрыве связи, объект ScadaCli будет пытаться повторно подключиться к одному из серверов, указанных в списке SrvAddress. Данное подключение будет осуществлено прозрачно для пользователя с полным сохранением текущего контекста.
InProcess	Признак занятости	Boolean. Значение равно TRUE, если CliActive = TRUE и объект ScadaCli, либо один из абонентов находятся в состоянии обработки события или формирования блока запроса. Иначе – FALSE.

3.1.2.2.3. События объекта

Событие	Назначение	Примечания
OnSesClose	Закрытие сессии сервером	Возникает в следствии завершения сессии сервером
OnSesAbort	Аварийный разрыв связи	Аварийное завершение сессии. Например, повреждение сети, выключение ЭВМ сервера и т.п.
OnSesUser	Изменение характеристик пользователя	Изменение характеристик пользователя на сервере (имя, пароль и т.п.). Новые характеристики доступны через свойство UserPrm.
OnLock OnUnlock	Обработка блокировки	Следствие использования методов Lock-Unlock.

3.1.2.2.4. Установка и разрыв соединения

```
function ScdConnect(Cli: TScadaCli): Boolean;
begin
  Cli.UserName := 'Ковтун';
  Cli.UserPassword := '1';
  Cli.SrvAddress := 'Oic1';
  Cli.Open;
  if not Cli.CliActive then
```

```
    if Cli.NoUser then
      Application.MessageBox('Имя или пароль неверны',
       'Вход в систему', MB_OK + MB_ICONSTOP)
    else
      Application.MessageBox('Сервер не найден',
        'Вход в систему', MB_OK + MB_ICONSTOP);
  Cli.WorkDir := 'D:\Scd\Temp';
  Result := Cli.CliActive;
end;
ScadaCli.Open;
Result := ScadaCli.CliActive

FScadaCli := TScadaCli.Create(Self);
    with FScadaCli do begin
      Name := 'ScadaCli';
      Parent := Self;
      Visible := False;
      OnSesAbort := ScadaCliSesAbort;
      OnSesClose := ScadaCliSesClose;
      OnSesUser := ScadaCliSesUser;
      OnLock := ScadaCliLock;
      OnUnlock := ScadaCliUnlock;
      ReconnectAuto := FConnBest;
    end;
```

3.1.2.3. Абонент (ScadaAbo)

3.1.2.3.1. Интерфейс IDataRec

Данный интерфейс используется для доступа к записи данных в обработчиках событий абонента OnRowValue и OnNotify. Запись данных состоит из одного или нескольких полей (FieldsCount). Каждое поле имеет имя, содержит данные определенного типа. Доступ к полям может производиться по индексу или по имени поля.

Метод	Назначение	Примечания
GetValue	Получение значения и параметров поля	GetValue(Field: OleVariant; out AType: Integer; out ASize: Integer): OleVariant; Метод возвращает все сразу: значение, тип и размер данных в байтах. Поле можно указать по индексу или его имени.
Свойство	Назначение	Примечания
FieldsCount	Количество полей в записи	Integer
FieldName	Имя поля	FieldName[Index: Integer]: String; Получение имени поля по индексу
FieldIndex	Индекс поля	FieldIndex[const Name: String]: Integer; Определение индекса поля по его имени. Если поля с таким именем не найдено, возвращается –1;
FieldValue	Значение поля	FieldValue[Field: Variant]: Variant;
FieldType	Тип поля	FieldType[Field: Variant]: EFieldType;
FieldSize	Размер поля	FieldSize[Field: Variant]: Integer;

3.1.2.3.2. Интерфейс IScadaAbo

Основной интерфейс, через который осуществляется работа с данными сервера: НСИ, архивами и событиями.

Метод	Назначение	Примечания
BlockBegin BlockEnd	Формирование блочного запроса	Все запросы к серверу на чтение, изменение данных или запуск серверных процедур выполняются в рамках блока. При выполнении команды BlockEnd можно отказаться от изменений, сделанных в БД (IsCommit=FALSE) и, ожидать результатов запроса (IsWait=TRUE) или продолжить выполнение без ожидания (IsWait=FALSE)

Proc	Запуск серверной процедуры	Proc(TblHandle: Integer; ReplaceEq: WordBool; DoNotify: WordBool);
		Пример использования. Чтение архива:
		with ScadaAbo. do begin
		RequestPrm['PROC']:='READ_ARCH';
		RequestPrm['TABLE_NAME']:='T_ARCH_TI';
		Proc(0,False,False);
		FieldValue('ID',eftInt,101);
		FieldValue('DT',eftInt,0); // Текущее
		Post;
		end;
		Три параметра, используемые в процедуре, связаны возможностью использования виртуальных таблиц для приема данных запроса. Если создать пустой объект MDArray и указать MDArray.Handle в качестве первого параметра, то данные будут приниматься в эту таблицу. Параметр ReplaceEq управляет режимом обработки совпадающих (по ключу) записей. Если DoNotify=FALSE,

		то, при наличии таблицы, обработчики OnProcBegin и OnProcEnd вызываться не будут. Если используемая виртуальная таблица не имеет структуры, то при приеме данных, автоматически будет сформирован список полей, соответствующий полям принимаемой записи. Если же список полей уже имеется в наличии, то при приеме данных будут использоваться уже существующие поля и индексы.
Sql	Выполнение SQL-запроса	Sql(TblHandle: Integer; ReplaceEq: WordBool; DoNotify: WordBool); Пример использования. Чтение таблицы T_TI: with ScadaAbo. do begin RequestPrm['SQL']:='SELECT * FROM T_TI'; RequestPrm['NAME']:='T_TI'; Sql(0,False,False); Post; end; Назначение параметров SQL-метода полностью совпадает с параметрами, описанными для Proc;
RowNew	Добавление новой строки НСИ	Пример использования. Добавление в T_RTS: with ScadaAbo. do begin RequestPrm['TABLE']:='T_RTS'; RowNew; FieldValue('RTS_USER_ID',eftInt,13); FieldValue('RTS_NAME',eftChar,'Тест'); // . . . Post; end;
RowEdit	Изменение строки НСИ	Пример использования. Изменение имени в T_DOC: with ScadaAbo. do begin RequestPrm['TABLE']:='T_DOC'; RequestPrm['KEY']:='DOC_ID=35'; RowEdit; FieldValue('DOC_NAME',eftMemo,'Доc2'); Post; end;
RowDelete	Удаление строки НСИ	Пример использования. Удаление схемы в T_SCH: with ScadaAbo. do begin RequestPrm['TABLE']:='T_SCH'; RequestPrm['KEY']:='SCH_ID=10'; RowDelete; Post; end;
Post	Завершение команд	Все команды блока (Proc, Sql, RowNew, RowEdit, RowDelete) должны завершаться командой Post. Между ними, при необходимости, могут быть заданы значения полей с помощью FieldValue.
FieldValue	Задание значения поля	Параметрами являются <имя поля>, <тип> и <присваиваемое значение>. Возможные типы поля описаны в EfieldType (eftBool, eftByte, eftSmall, eftInt, eftFloat, eftDouble, eftChar, eftMemo, eftBlob, eftUnixDT, eftUnixMDT, eftCurrency)

EventConnect	Подключение к оповещению по событиям	Параметры EventConnect – два variant-массива. Первый – список кодов требуемых кодов событий, второй – список идентификаторов энергообектов, на которых эти события интересуют пользователя-абонента. События изменения НСИ для энергообъекта – владельца БД всегда принимаются по умолчанию. Повторный вызов EventConnect обнуляет старый список оповещения для данного абонента и заменяет его на новый.
		With ScadaAbo. do begin
		`Cod:=VarArrayCreate([0,1],varInteger);`
		Cod[0]:=101; Cod[1]:=102;
		`Eno:=VarArrayCreate([0,2],varInteger);`
		Eno[0]:=7; Eno[1]:=10; Eno[2]:=13;
		EventConnect(Cod, Eno);
		end;
		Если в списке встречается «0», то это означает - для всех элементов (энергообъектов или кодов событий)

FieldDef	Параметры поля таблицы-результата	В обработчиках событий OnSqlBegin и OnProcBegin можно узнать структуру полей записи с помощью FieldDef. Свойство FieldsCount возвращает количество полей в записи. FieldDef для каждого поля (по индексу) выдает соответственно: тип, размер и наименование.
Свойство	Назначение	Примечания
RequestPrm	Задание параметров запроса	Параметром запроса называется пара <Имя>=<Значение >. Совокупность таких параметров для каждой команды передается на сервер и используется им для конкретизации действий. Все параметры, в том числе и те, которые сервером не используются, возвращаются с результатом запроса и становятся доступны в обработчиках OnSqlBegin и OnProcBegin с помощью свойства OnAnswerPrm. Примеры применения RequestPrm см. выше, в примерах команд Proc, Sql и т.п.
OnAnswerPrm	Параметры результата запроса	С помощью свойства OnAnswerPrm можно при получении результата запроса узнать значения параметров, переданные на сервер командами Proc и Sql. Все параметры, в том числе и те, которые сервером не используются, возвращаются с результатом запроса и становятся доступны в обработчиках OnSqlBegin и OnProcBegin.
CliHWnd	Описатель окна объекта ScadaCli	С помощью данного свойства объект ScadaAbo связывается с объектом ScadaCli. Пример: ScadaAbo.CliHWnd:=ScadaCli.hWnd;
ScadaCli	Интерфейс IscadaCli	IScadaCli. Если с данным абонентом не связан объект ScadaCli, то NIL.
LastErrorCode LastErrorStr	Код и строка последней ошибки	Код и строка последней ошибки, возникшей в результате обработки блока данных на сервере. Обычно, после возникновения такой ошибки, изменения БД, сделанные в данном блоке –откатываются, генерируется

		сообщение об ошибке, дальнейшее выполнение команд блока прекращается.
FieldsCount	Количество полей в записи результата запроса	Актуально в теле обработчиков OnProcBegin и OnSqlBegin. Характеристики полей можно получить по индексу с помощью метода FieldDef.
AboActive	Рабочее состояние абонента	Абонент ScadaAbo подключен к ScadaCli и ScadaCli.CliActive=TRUE
InRequestBlock	Абонент подготавливает блок запроса	Объект абонента находится в процессе подкотовки блока запроса, т.е. BlockBegin-BlockEnd;
InAnswerBlock	Абонент принимает данные	Объект абонента находится в процессе приема и обработки данных, полученных в результате запроса к серверу.

3.1.2.3.3. События объекта

Большинство обработчиков событий объекта, связано с процессом приема и обработки данных, полученных в результате запроса к серверу. Последовательность вызова этих процедур следующая:

```
OnBlockBegin;
  OnProcBegin(var TblHandle: Integer; var ReplaceEq: WordBool);//OnSqlBegin;
    OnRowValue(const RecData: IDataRec; IsSql: WordBool);
  OnProcEnd(TblHandle: Integer);//OnSqlEnd
  OnError(ErrCode: Integer; const ErrStr: WideString);
OnBlockEnd(IsCommit: Boolean);
```

Каждый объект абонента, всегда подключен к оповещению о событиях изменения НСИ, возникающих на сервере, при корректировке таблиц НСИ. При необходимости, подключение к другим событиям, осуществляется с помощью метода EventConnect. В любом случае, процесс обработки сообщений использует единственный обработчик:

```
OnNotify(EventCode: Integer; const RecData: IDataRec);
```

Событие	Назначение	Примечания
OnBlockBegin OnBlockEnd	Начало и конец блока результата	Обработчики OnBlockBegin и OnBlockEnd вызываются всегда, для каждого блока команд, выполненного данным абонентом на сервере.
OnProcBegin OnProcEnd	Начало и конец	В обработчике OnProcBegin можно установить (или изменить) виртуальную таблицу или ее параметры.

	результата процедуры	Параметры, сформированные перед запуском процедуры Proc, доступны через свойство OnAnswerPrm Если в команде «Proc» была задана виртуальная таблица для принятия данных и, параметр DoNotify был TRUE, то OnProcBegin и OnProcEnd. вызываться не будут.
OnSqlBegin OnSqlEnd	Начало и конец результата SQL-запроса	Аналогично обработке OnProcBegin-OnProcEnd.
OnRowValue	Запись (строка) результата	Вызывается для каждой записи результата Proc- или Sql-запроса. Сколько записей – столько вызовов. Если задана виртуальная таблица, то данные складываются непосредственно в нее и OnRowValue не вызывается.
OnError	Ошибка	Если в процессе обработки блока запроса на сервере произошла ошибка, то дальнейшее выполнение команд блока прекращается, произведенные изменения «откатываются», генерируется данное событие с кодом и строкой ошибки и блок завершается
OnNotify	Запись (строка) события	OnNotify(EventCode: Integer; const RecData: IDataRec); Если абонент подключен к оповещению по событиям, то данный обработчик будет вызван для каждого такого события, зарегистрированного на сервере.
OnAboActive	Изменение режима работы	Вызывается, если изменяется свойство абонента AboActive.

3.1.2.4. Дополнительные возможности

3.1.2.4.1. Таблицы в памяти.IMDArray, IMDARec

Таблицы в памяти (MDArray) представляют собой набор однотипных записей (MDARec), связанных между собой различными отношениями порядка.

Формирование и просмотр структуры записи (полей) осуществляется с помощью методов и свойств: FieldAdd, FieldGet, FieldCount.

Работа с индексами (ключами), задающими порядок записей в таблице: KeyAdd, KeyDel, KeyGet, ClearKey, KeyCount.

Создание, удаление и редактирование записей: RecNew. RecEdit. RecPost, RecDel, ClearRec, RecCount.

Сортировка, поиск и примой доступ к записям: Sort, Seek, Rec, RecIdx.

Метод ClearAll, полностью освобождает все данные, связанные с таблицей.

Метод	Назначение	Примечания
FieldAdd	Добавление нового поля	FieldAdd(const AName: String; AType: EFieldType; ALen: Integer). Применяется при первоначальном формировании структуры записи. Имена полей не должны совпадать. Параметр длины имеет смысл только для строкового типа фиксированной длины. При изменении

		структуры полей все существующие записи данных уничтожаются.
FieldGet	Получение характеристик поля	FieldGet(AFld: Variant; out ANamIdx: Variant; out AType: EFieldType; out ASize: Integer); Обращение к полю можно осуществлять как по имени, так и по индексу. При этом, в зависимости от выбранного метода, в параметре ANamIdx, будет возвращено альтернативное значение.
KeyAdd	Добавление ключа (индекса)	KeyAdd(const AFields: String; AOptions: Integer); С помощью ключей задается порядок записей в наборе. Ключ определяется набором имен полей (параметр AFields), перечисленных через «;» и дополнительными опциями (AOptions). Значение Aoptions, комбинация флагов AOptions: IdxPrimary(1)-первичный ключ; IdxUnique(2)-уникальный, не может быть записей с совпадающим ключем; IdxDescending(3)-упорядочить по убыванию (порядок по умолчанию - возрастание); IdxCaseInsensitive(4)-не учитывать регистр при сравнении строк; IdxExpression(5)-(зарезервировано); IdxNonMaintained(6)-индекс не корректируется, при изменении записей. Перестройка индекса должна осуществляться явно командой Sort.
KeyDel	Удаление ключа	KeyDel(AKey: Integer) Невозможно удалить последний ключ, если в таблице есть записи.
KeyGet	Получение характеристик ключа	KeyGet(AKey: Integer; out AFields: String; out AOptions: Integer);
Copy	Копирование из другой таблицы	Copy(const ASrc: IMDArray; ARec: WordBool; AKey: WordBool) Копирование структуры полей другой таблицы. Если ARec или AKey = TRUE, то будут также скопированы все записи и (или) ключи таблицы-источника.
ClearRec	Удаление всех записей	Структура полей и ключи остаются
ClearKey	Удаление ключей	Структура полей и записи остаются. Если в таблице есть записи, то останется один ключ с пустым набором индексных полей.
ClearAll	Очистка таблицы	Удаляются все записи, все ключи таблицы и, наконец, очищается список полей.
RecNew	Создание новой записи	RecNew(const ATarget: IMDARec): IMDARec; Принцип добавления новой записи следующий: командой RecNew создается пустая запись, устанавливаются значения ее полей, командой RecPost, запись размещается в таблице. Если объект ATarget = nil, то будет создан и возвращен новый объект IMDARec, если же ATarget <> nil, то новая запись будет связана с ATarget и RecNew вернет его значение. Пример: ATbl: IMDArray; ARec: IMDARec; ARec:=ATbl.RecNew(nil);

		ARec.Value['ID']:=177; ARec.Value['NAME']:='ТИ'; ATbl.RecPost(ARec); Для отказа от добавления уже созданной записи, нужно применить к ней метод RecDel.
RecEdit	Изменение существующей записи	RecEdit(const ARec: IMDARec; const ATarget: IMDARec): IMDARec; Данный метод создает копию редактируемой записи и возвращает ее в качестве результата. После внесения изменений, метод RecPost заменит исходную запись на ее измененный вариант. Для отказа от фиксации изменений нужно применить к записи, возвращенной RecEdit метод RecDel.
RecPost	Фиксация изменений	RecPost(const ARec: IMDARec); Фиксирует изменения, начатые с помощью команд RecNew или RecEdit.
RecDel	Удаление записи	RecDel(const Arec: IMDARec); Удаляет запись из таблицы. Если метод применяется к указателям на записи, полученным с помощью RecNew или RecEdit, то это означает отказ от изменений.
Seek	Поиск записи	Seek(AKey: Integer; AKeyVal: Variant; out ARecIdx: Integer; const ATarget: IMDARec): IMDARec; Поиск записи осуществляется по ключу, индекс которого задается в AKey. AKeyVal – искомое значения, в соответствии со списком полей для данного ключа. Когда полей несколько, то AKeyVal – массив Variant-ов. Если запись найдена, то Seek возвращает указатель на нее (или ATarget, если он был задан). Иначе возвращается nil. В любом случае ARecIdx – индекс найденной или первой большей записи из списка для данного ключа.
Sort	Сортировка по ключу	Sort(AKey: Integer) Принудительная сортировка. Имеет смысл для ключей со свойством IdxNonMaintained.
Rec	Запись из списка	Rec(AKey: Integer; AIdx: Integer; const ATarget: IMDARec): IMDARec; Для каждого ключа записи упорядочены по-разному. Поэтому для получения записи нужно указать ключ и индекс ее в списке.
Свойство	Назначение	Примечания
FieldCount	Количество полей	Integer
KeyCount	Количество ключей	Integer
RecIdx	Индекс записи в списке	RecIdx[AKey: Integer; const ARec: IMDARec]: Integer; Для любого ключа можно определить индекс записи в списке.
RecCount	Количество записей	Integer
LastError	Ошибка	EADArrayErr Если какая-нибудь операция закончилась неудачно, то тип последней ошибки можно узнать с помощью данного свойства:

		RAE_OK – все нормально; RAE_NOMEM – недостаточно ОП; RAE_TRANSLATE – возникает при добавлении полей или ключей. Например, имя поля уже существует или, поле, указанное в ключе, отсутствует в списке полей; RAE_NOFIELDS – поля в таблице не определены; RAE_OTHERFIELDS – попытка утвердить изменения если запись и таблица разной структуры; RAE_DATAREC – структура данных записи нарушена; RAE_NOREC – при выполнении метода RecPost, не найдена исходная, изменяемая запись; RAE_KEYUNIQUE – запись не может быть добавлена или изменена, т.к. нарушается уникальность ключа.
RecChanges	Количество изменений	Integer Данное поле можно обнулять. Каждое изменение записей наращивает значение RecChanges
Handle	Описатель	Integer
Tag	Поле пользователя	Integer

Запись, это собственно та структура, которая хранит сами данные. Некоторые ее методы и свойства дублируют аналогичные в таблице: FieldGet, FieldCount.

Работа со значениями полей осуществляется с помощью: Value, AsText, IsNull.

Метод	Назначение	Примечания
FieldGet	Получение характеристик поля	FieldGet(AField: Variant; out AnamIdx: Variant; out AType: EFieldType; out ASize: Integer); Аналогично FieldGet для IMDArray
Свойство	Назначение	Примечания
Handle	Описатель записи	Integer
AsText	Значение поля как текст	AsText[AField: Variant]: String; Текстовое значение поля. Поле может быть указано как по имени, так и по индексу в списке полей.
Value	Значение поля	Value[AField: Variant]: Variant;
IsNull	Признак пустого поля	IsNull[AField: Variant]: WordBool;
IsEdit	Признак измененного значения	IsEdit[AField: Variant]: WordBool;
ArrHandle	Описатель таблицы	Integer
FieldCount	Количество полей	Integer Аналогично FieldCount для IMDArray

3.1.2.4.2. Хранилище данных. IDataBag

Объекты, реализующие интерфейс IDataBag, используются в системе для работы с данными древовидной структуры, похожей на структуру каталогов Windows.

«Дерево» состоит из поименованных узлов, каждый из которых имеет VARIANT-значение и может быть владельцем произвольного количества других, подчиненных ему узлов. Списки подчиненных узлов упорядочены по именам. Имена внутри одного списка должны быть уникальны. Всегда существует главный, корневой узел всего «дерева».

В большинстве методов и свойств интерфейса используется параметр «Path» (путь), с помощью которого указывается узел, к которому будет применено действие. Путь – это список имен и (или) индексов, по которому можно последовательно добраться до искомого узла, начиная с корневого. Корневой узел имени не имеет и идентифицируется пустой строкой или NULL-значением Path.

Для удобства работы можно связать временное положение корневого узла с любым существующим узлов в дереве. Это делается с помощью свойства RootPath; При этом свойство RootIsEmpty будет показывать, совпадает ли виртуальный корневой узел с реальным.

В качестве VARIANT-значений узлов могут быть использованы массивы.

Метод	Назначение	Примечания
SetNode	Присвоение значения узлу	SetNode(Path: Variant; Val: Variant): WordBool; Если узлы пути не найдены, но указаны по именам, то они создаются. Последнему узлу присваивается значение Val. Если значение присвоено, метод возвращает TRUE;
DelNode	Удаление узла	DelNode(Path: Variant); Удаление узла приводит к удалению его значения и всех подчиненных ему узлов дерева. Нельзя удалить корневой узел.
Clear	Очистка корневого узла	Удаляются данные и все подчиненные узлы для корневого узла

Свойство	Назначение	Примечания
Name	Имя узла	Name[Path: Variant]: String; Получение имени узла. Пустая строка, если узел не найден
Index	Индекс узла в списке подчиненных	Index[Path: Variant]: Integer; Получение индекса узла в списке подчиненных узлов которому он принадлежит. –1, если не найден
ChildCount	Количество подчиненных узлов	ChildCount[Path: Variant]: Integer; Количество узлов, непосредственно подчиненных указанному
Data	Значение узла	Data[Path: Variant]: Variant;

RootPath	Реальный путь к виртуальному корневому узлу	RootPath: Variant; Путь NULL или пустая строка соответствует реальному корневому узлу дерева
Block	Данные корневого и подчиненных ему узлов	String Данные корневого и подчиненных ему узлов, упакованные в строку
BagTree	Данные корневого и подчиненных ему узлов	BagTree[Path: Variant]: IDataBag; Данные корневого и подчиненных ему узлов, скопированные в новый объект
RootIsEmpty	Признак совпадения реального и виртуального корня	WordBool

3.2. АРМ пользователя

3.2.1. Наращивание функциональности

3.2.1.1. Обзор

Программа АРМ пользователя (ScdArm.exe) по своей архитектуре является типичным контейнером ActiveX-объектов. Она обеспечивает загрузку и поддержку функциональных модулей системы, контроль и обновление версий используемых файлов и библиотек, отвечает за общий сценарий работы с сервером.

Однако, сами функциональные модули, не являются частью программы АРМ. Они представляют собой библиотеки ActiveX-объектов, которые могут быть разработаны независимо, на любых языках, поддерживающих OLE-технологию Microsoft.

Для того, чтобы ActiveX-объект мог использоваться в составе оболочки АРМ, он должен реализовать интерфейс IscadaObj или его Dispatch-вариант. Данный интерфейс, наряду с многими другими, определен в ScdSys.ocx.

3.2.1.2. Интерфейс IScdDisp

Данный интерфейс предназначен для взаимодействия с произвольной объектной моделью из языка Tscript.

Свойство	Назначение	Примечания
Prop	Доступ к свойствам объекта	Prop([in] VARIANT AName) VARIANT Prop([in] VARIANT AName, [in] VARIANT Value); AName – имя или идентификатор свойства. Если свойство индексированное, то AName – массив, первый элемент которого имя или идентификатор свойства;

Метод	Назначение	Примечания
Call	Выполнение метода объекта	Call([in] VARIANT AName, [in] VARIANT Param): VARIANT; AName – имя или идентификатор метода; Param – параметр или массив параметров.
Событие	Назначение	Примечания
OnEvent	Событие	OnEvent([in] VARIANT Name, [in] VARIANT Param); Оповещение о событиях объекта. Name – имя или идентификатор события; Param – дополнительные параметры;

3.2.1.3. Интерфейс IscadaObj

Метод	Назначение	Примечания
ObjDataCancel	Отказаться от изменений	Данный метод дает директиву объекту – отказаться от сделанных изменений данных, если они есть. О наличии таких изменений можно судить по значению свойства ObjDataIsEdit, а обработчик событий OnObjData должен информировать об изменении состояния данных.
ObjDataSave	Сохранить данные	Директива – сохранить измененные данные.

ObjStateLoad
ObjStateSave
	Загрузка-сохранение состояния в БД	ObjStateLoad(Ench: WordBool);
		ObjStateSave(Ench: WordBool);
		Директива – загрузить (сохранить) текущее состояние в БД. Ench – признак «расширенного» набора свойств.
		Обычно, объекты хранят состояние в таблице T_DATA_BAG, используя методы ScadaCli. DataBagGet и ScadaCli.DataBagPut. Например,
		
		procedure TModObj.ObjStateLoad(Ench: WordBool);
		var IB: IDataBag;
		begin
		IB := ScadaCli.DataBagGet('STATE_' +
		Copy(ClassName,2,50), True);
		InternalStateSet(Ench, IB);
		end;
		
		procedure TModObj.ObjStateSave(Ench: WordBool);
		var IB: IDataBag;
		begin
		IB := InternalStateGet(Ench);
		ScadaCli.DataBagPut('STATE_' +
		Copy(ClassName, 2, 50), True, IB);
		end;

ObjProcessStart ObjProcessStop	Начало и конец работы	Начало и конец работы модуля. В начале работы модуль считывает данные с сервера и инициализирует свои структуры данных. В конце – все данные очищаются. Если модуль активен, то ObjInProcess = TRUE

ObjParamGet ObjParamPut	Чтение-установка параметров	ObjParamGet(Param: Variant): Variant; ObjParamPut(Param: Variant; Value: Variant); Объект может не поддерживать работу с параметрами, однако работа с параметрами локального состояния желательна: 'STATE' и 'STATEENCH'. Это те же самые данные, которые сохраняются и загружаются в ObjStateLoad и ObjStateSave.
Свойство	Назначение	Примечания
ObjCliHWnd	Описатель окна объекта ScadaCli	Обычно, для работы с данными сервера, сразу после своего создания, объект создает своего абонента - ScadaAbo. Первое, что делает АРМ, создав объект, это передает ему описатель канала через это свойство. С его помощью ActiveX подключает абонента к каналу.
ObjMenu	Меню	Одна из функций АРМ, отображение меню активного объекта. Для этого он использует данное свойство. Подробное описание интерфейса IscadaMenu см. ниже в следующем разделе
ObjDataIsEdit	Признак наличия изменений	Данное свойство показывает наличие в объекте не сохраненных данных.
ObjCliActive	Канал активен	TRUE, обычно означает, что у объекта есть объект ScadaAbo, он подключен к объекту канала (ScadaCli) и данный канал активен.
ObjInProcess	Процесс активен	Объект находится в процессе работы: общается с сервером, обрабатывает сообщения мыши и клавиатуры и т.п. Дело в том, сто функциональный модуль может быть подключен к каналу, но бездействовать. Начало и конец процесса обработки должны регламентироваться директивами ObjProcessStart и ObjProcessStop
ObjInNetEvent	Признак обработки сообщения	Если объект занят (например, принимает или обрабатывает данные) и на данный момент не расположен выполнять других директив, то данный признак может сигнализировать об этом

3.2.1.4. События объекта

Механизм событий используется для оповещения контейнера об изменении состояний объекта, чтобы он мог адекватно на них реагировать.

Событие	Назначение	Примечания
OnObjCaption	Изменение заголовка	Произошло изменение заголовка (свойства Caption) объекта
OnObjMenu	Изменение меню	Произошло изменение меню (свойство ObjMenu) объекта
OnObjData	Изменение данных	Изменилось состояние данных. Данные были скорректированы, но еще не сохранены (свойство ObjDataIsEdit)
OnObjNotify	Оповещение контейнера	OnObjNotify(var Param: OleVariant); С помощью данного события объект может оповещать свой контейнер о чем угодно. Главное, чтобы контейнер понимал, чего от него хотят.

3.2.1.5. Меню. IscadaMenu

Свойство	Назначение	Примечания
Caption	Текст меню	String

ShortCut	Горячая клавиша	Integer.
		Пример формирования кода (Menus.pas):
		
		Function ShortCut(Key: Word; Shift: TShiftState): TShortCut;
		begin
		  Result := 0;
		  if WordRec(Key).Hi <> 0 then Exit;
		  Result := Key;
		  if ssShift in Shift then Inc(Result, scShift);
		  if ssCtrl in Shift then Inc(Result, scCtrl);
		  if ssAlt in Shift then Inc(Result, scAlt);
		end;

HelpContext	Идентификатор контекстной помощи	Integer.
Checked	Отмеченный	Boolean
RadioItem	Группа	Boolean
Enabled		Boolean
Visible		Boolean
Bitmap		
GroupIndex		Integer
Name	Имя	String
Tag	Доп. значение	Integer
ItemNext	Следующий в списке	IscadaMenu
ItemChild	Первый подчиненный	IscadaMenu

4. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

4.1. Общие положения.

Входными данными для сервера приложений ПК являются:

1. Конфигурационный ini-файл программы, считываемый на этапе загрузки. Из файла сервер приложений берет всю необходимую информацию о серверных комплексах, в составе которых он работает;
2. База данных НСИ ПК;
3. Запросы от АРМ ПК.

Выходными данными сервера приложений ПК являются:

1. Log-файл работы сервера ПК;
2. Ответы на запросы АРМ ПК;
3. База данных реального времени;
4. Архив событий ПК;
5. Результаты расчетов на языке технологического программирования TScript.

4.2. Настройка серверной части ПК

Описывается в документt «С-Платформа. Руководство системного программиста (администратора)»).

4.3. Организация расчетов

4.3.1. Общие положения

Все расчеты, проводимые в ПК, описываются в таблице **CALCULATIONS** базы данных на языке **TScript**. Возможности языка TScript расширены с помощью библиотеки внешних функций (описаны в приложении). Расчеты выполняются в трех случаях:

1) Автоматически, если в архив записываются параметры, которые участвуют в каких-то расчетах. Результаты расчетов записываются в соответствующие архивы с расчетными параметрами.
2) Через механизм виртуальных расчетных параметров.

   Для них расчеты выполняются при каждом запросе параметра из архива.
3) Циклически. Для чего разработана подсистема циклических расчетов.

4.3.2. Автоматические расчеты

Для организации автоматических расчетов приняты ряд соглашений, благодаря которым имеется возможность определить, в какие расчеты входит записываемое в архив данное, выполнить эти расчеты и записать результаты в соответствующий архив.

Определена функция доступа D, которая на языке Tscript выглядит следующим образом:

```
D (pId,pTime)
{
        Result = _ReadData(pId,pTime);
};
```

Результаты расчета укладываются в соответствующий архив с расчетными данными (**ARCH_CON_CALC=TRUE** и **ARCH_CON_WR_CALC=TRUE**).

Для определения архива с расчетными параметрами необходимо в поле **ARCH_CALC_PREF** указать *префикс функции* для данного архива.

Например: для архива часовой ведомости с данными ПТИ это - PTI. В этом случае, если выполняется расчет, который является функцией с именем PTI10, то это значит, что результат будет записан именно в архив часовой ведомости с данными ПТИ и идентификатор записываемого данного – 10.

Если архивы с расчетными данными идентичны по составу данных, то есть для них назначена одна и та же таблица НСИ, то можно использовать для них один и тот же префикс функции, что позволяет использовать одни и те же описания расчетов для разных архивов.

Например, одна и та же функция:

```
PTI10 (pTime)
{
    TI (1, pTime) + TI (2, pTime);
}
```

может использоваться при расчете псевдо-ТИ, данного архива часовой ведомости с псевдо-ТИ, данного архива получасовой ведомости с псевдо-ТИ. Только, естественно, соответствующим образом должна быть скорректирована функция доступа TI с тем, чтобы данные для расчетов брались из нужных архивов.

Есть исключение для архивов с ТИ, ТС, псевдо-ТИ и псевдо-ТС. Для них функции доступа и префиксы функций не указываются, так как все необходимые расчеты выполняются в сервере ввода-вывода и затем посчитанные значения подвергаются дополнительной обработке.

4.3.3. Циклические расчеты

Данная подсистема позволяет без написания дополнительных программ решить множество задач. Все, что должно циклически считаться, можно реализовать с помощью этой подсистемы.

Например:

- Интегрирование параметра с подсчетом среднего значения или электроэнергии по завершении периода интегрирования;
- Циклический расчет потерь мощности и электроэнергии в различных элементах электрической сети;
- Циклический контроль заданной совокупности параметров, с последующей выдачей сообщений об изменении их состояния;
- Заполнение архивов данными других архивов с заданной цикличностью;
- и так далее.

Для ввода нового циклического расчета необходимо:

- Создать новую таблицу с НСИ циклических расчетов. В такой таблице должно быть три поля. Существенным являются окончания названий полей (**ID, NAME, CALC_ID**). По ним идентифицируется информация в полях. Остальная часть наименований полей несущественна.

Например:

1) CALC_C_D_1_ID - идентификатор расчетного параметра (тип – длинное целое);
2) CALC_C_D_1_NAME - наименование расчетного параметра (тип - текст);
3) CALC_C_D_1_CALC_ID - идентификатор расчета. Ссылка на T_CALC. Тип – длинное целое.

- Включить описание таблицы в таблицу описания таблиц базы Системы (T_OBJ).
- Включить новую настройку в таблицу настроек циклических расчетов **T_CALC_C**. Необходимо правильно описать поля, задающие период расчета, цикл расчета и задержку расчета.
- В таблице T_CALC описать функции, которые надо циклически выполнять и включить их номера в созданную таблицу с НСИ циклических расчетов.

4.4. Входные данные АРМ ПК

4.4.1. Файл локальных настроек АРМ

Файл параметров – Scada.ini, используется программой АРМ-клиента (**ScdArm.exe**) для получения информации о доступных серверах Системы, списке пользователей и «синонимах», используемых при конфигурировании и работе АРМ.

Раздел «Серверы» содержит список доступных серверов ПК, к которым может подключиться пользователь с данной ЭВМ. Строка описания сервера состоит из фиксированной части «Сервер» с порядковым номером этой строки в списке. После символа «=» идет описательная часть сервера, состоящая из элементов, разделенных запятыми. Первый элемент – наименование сервера, которое будет отображаться в АРМ-е. Остальные элементы – сетевые имена основного и резервных серверов, которые обеспечивают распределенную работу с выбранной БД. Список сетевых имен рекомендуется начинать с имени основного сервера. Вместо имени сервера допустимо указать его IP адрес. Признак «/lib», размещенный после любого элемента в списке, разрешает обновлять исполняемые модули и библиотеки системы с БД, поддерживаемой этими серверами. Если в одном из компонентов описания сервера, разделенных запятыми, встречаются знаки пробел или запятая, то содержимое элемента нужно заключить в двойные кавычки.

По умолчанию, АРМ клиента подключается к серверу, используя порт 1212. Если это не так, то нужный номер порта можно указать сразу после имени сервера, например, «Srv3/1232». Номер порта по умолчанию для всего комплекса можно изменить, добавив его сразу после наименования комплекса.

[Серверы]

Сервер0="Сервер ОИК/lib",Srv1,Srv2

Сервер1="Тестовый сервер",Test1/1233

Сервер2="Сервер Буденновск 1/1244/lib",oik_main,172.18.38.3/1245

Сервер3="Сервер Буденновск 2",172.18.38.3,oik_test2

Последнее вхождение=1

Элемент раздела «Последнее вхождение» показывает к какому серверу из списка было сделано последнее, успешное подключение. Данный параметр устанавливается программой автоматически.

Раздел «Пользователи» содержит параметр – «Имена». Данный параметр содержит список из 10 последних имен пользователей, успешно входивших в систему с данной ЭВМ. Имена располагаются в порядке очередности вхождения. Список формируется программой автоматически

Параметры «Оптимальное подключение» и «Восстановление конфигурации» используются для хранения состояния одноименных флажков на заставке входа в программу АРМ-клиента.

[Пользователи]

Имена="Mid,Tu,wert,Sw,Телемеханик,Ковтун,ddd,sv"

Оптимальное подключение=0

Восстановление конфигурации=1

Параметры раздела «Замена строк» используются программой АРМ-клиента при формировании рабочей конфигурации модулей. При этом все вхождения подстрок, расположенные слева от знака «=», заменяются при настройке на значения, расположенные справа.

[Замена строк]

$DirMsOffice$=C:\msoff97

$DirMyFles$=D:\DataXml

Например, если в конфигурации с некоторой кнопкой связан запуск программы Excel.exe, а расположение директория MsOffice различно на разных ЭВМ, то путем использования в конфигурации подстроки «$DirMsOffice$» можно добиться корректного запуска Excel на разных ЭВМ, указав для каждого действительного расположения директория MsOffice.

Помимо замен, определенных в данной секции можно использовать следующие предопределенные имена директориев:

- $DirWin$ - директорий windows;
- $DirSys$ - директорий windows\system32;
- $DirScd$ - директорий клиента scada;
- $DirTmp$ - директорий временных файлов и данных клиента.
- $DirLib$ - аналогичен $DirTmp$;

К параметрам раздела «Разное» относятся параметры «Заставки», «Звукового сигнала» и «Схема диспетчерского щита».

С помощью первого можно управлять картинкой-заставкой, появляющейся на экране при входе в систему. В качестве его значения нужно указать имя альтернативного файла-заставки (*.bmp, *.wmf, *.emf, *.jpeg, *.jpg). Если указать пустое значение, то картинка-заставка появляться не будет. В имени файла допустимо использование строк замены, описанных выше.

Файл звукового сигнала предназначен для сигнализации пользователю об аварийном событии, зарегистрированном в системе. По умолчанию, если звуковой сигнал не указан, ищется файл с именем «$DirTmp$ \Alarm.wav» и «$DirScd$ \Alarm.wav». Если указанный файл существует, то он используется для звукового оповещения. Если же файл не указан или не найден - стандартный «Beep».

Если присутствует пункт "Схема диспетчерского щита=", то АРМ будет загружаться без запроса пароля и сразу выводить на экран заданную схему в полноэкранном режиме.

Поля в строке данных следующие:

1) Строка параметров. Та же самая, как описано для переходов и кнопок. конфигураций (см. «Графический инструментарий Модус»).
2) Номер сервера из раздела [Сервера], к которому будет произведено подключение.
3) Login – пользователя.
4) Пароль пользователя.

5, 6 - Ширина и высота окна в пикселях.

Если последние параметры не заданы, то полноэкранное окно раскрывается как обычно на полный экран монитора.

[Разное]

;Заставка=

Заставка=$DirTmp$\panel.bmp

Звук Alarm=$DirTmp$\MyAlarm.wav

Схема диспетчерского щита=""REC=332,ZOOM=58,POSX=58",0,EVA,"124",2304,1728".

Раздел «Монитор» и все его пункты используются программой ScdMon.exe для внутренних нужд и формируются ей автоматически.

4.4.2. Автоматическое обновление файлов на АРМ ПК

Механизм обновления файлов предназначен для обновления файлов ядра (ScdMdx.dll, ScdLib.bpl, ScdSys.ocx и ScdArm.exe), функциональных модулей клиентской части системы (ScdAdm.ocx, ScdTool.ocx, ScdStd.ocx и т.д.), файлов помощи и других файлов (например, таблиц Excel), новые версии которых нужно оперативно распространить на все машины клиентов Системы.

Все файлы требующие обновления (в том числе и системные) должны быть помещены в таблицы T_MOD БД откуда они будут автоматически тиражироваться на машины клиентов. При этом, передача файлов осуществляется с помощью внутреннего механизма связи Системы и нет необходимости открывать клиентам дополнительные сетевые ресурсы.

Принцип обновления этих файлов следующий:

*.exe, *.ocx, *.hlp, *.cnt – размещаются в директории Системы ($DirScd$);

*.dll, *.bpl – в системный директорий ОС ($DirSys$);

все остальные во временный директорий для текущего комплекса Системы ($DirTmp$).

Первые две группы файлов обновляются, только если для комплекса в файле Scada.ini задан ключ “/lib”. Остальные файлы обновляются всегда. Версии файлов определяются по времени последней модификации.

Если обновляются системные файлы (ScdMdx.dll, ScdSys.ocx, ScdArm.exe), то после копирования этих файлов во временный директорий, автоматически запускается вспомогательная программа ScdCpy.exe, которая переписывает и регистрирует системные файлы и, после обновления, перезапускает АРМ.

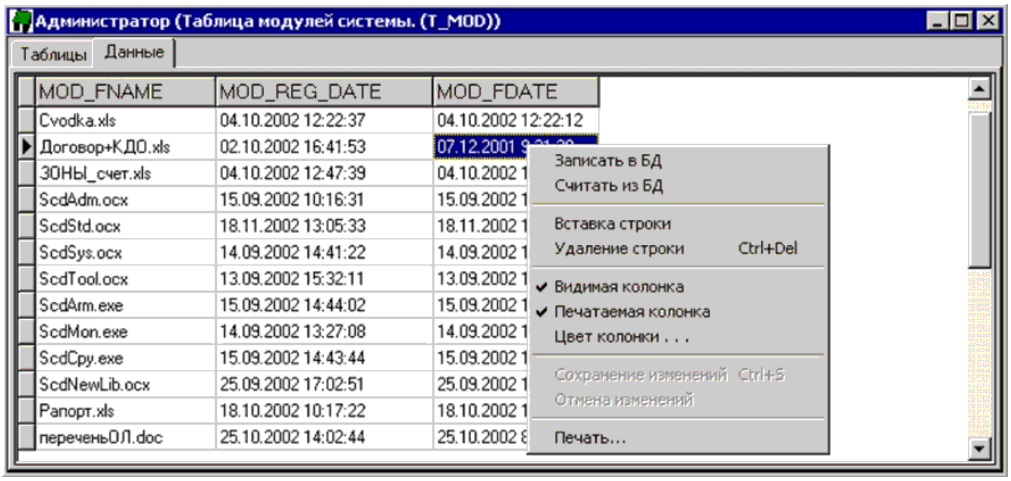


Рисунок 4.1 - Таблица модулей системы

Таблица «T_MOD» располагается обычно в разделе «НСИ ядра Scada», модуля администратора системы. С помощью пункта меню «Вставка строки», можно добавить новый файл в библиотеку, «Записать в базу» - обновить содержимое уже зарегистрированного файла, «Считать из БД» - вручную вытащить файл из БД на диск (при этом регистрация исполняемых файлов не производится).

4.5. Язык технологического программирования TScript

Описание языка и работа с ним в документе «С-Платформа. Руководство программиста. Том 3. Описание языка».

5. СООБЩЕНИЯ

В процессе работы программы, при возникновении ошибочных ситуаций, программа фиксирует данные события путем записи соответствующих диагностических сообщений в файл ScdSrv.log.

Формат сообщений единообразен и представляет собой отдельную строку текста с указанием времени добавления сообщения. Например:

=> 12/11/12 01:15:03 Создан объект меж серверной синхронизации

Инструментальная библиотека ScdMdx.dll, также может выдавать сообщения в файл ScdMdx.log. Сообщения касаются в основном событий разрыва канала связи с сервером. Пример такого сообщения:

=> 10/11/12 17:27:21.890 (0x1A10) MDSOCK: Recv_Net, ошибка разрыва канала(IDN=0x20000, TYPE=3, IP=127.0.0.1, NM=MID1, WSERR=10054)

В параметрах данного сообщения указывается идентификатор сокета соединения, тип (1-прослушивание, 2-сервер, 3-клиент), IP-адрес, имя абонента и ошибка сокета.

ПРИЛОЖЕНИЕ 1

Информационное. Перечень терминов.

Перечень терминов

ПК	- программный комплекс
НСИ	- нормативно-справочная информация
ОИК	- оперативно-информационный комплекс
ПО	- программное обеспечение
ПТК	- программно-технический комплекс
ТС	- технические средства

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Изм	Номера листов (страниц)				Всего листов (страниц) в докум.	№ документа	Входящий № сопроводительного докум. и дата	Подп.	Дата
	измененных	замененных	новых	аннулированных					