Macroscop SDK и API

Служба технической поддержки:
Телефоны: 8-800-555-0043 (бесплатно из любой точки России)
+7 (342) 215-09-78
E-mail: support@macroscop.com
Skype: macroscop.support

Опубликовано: 03.12.2019

Оглавление

1. Общие сведения о Macroscop SDK

Macroscop SDK – это инструментарий, позволяющий создавать программное обеспечение, именуемое плагинами (внешними модулями), позволяющее расширять существующие функциональные возможности программного комплекса **Macroscop**.

Данный инструментарий предназначен для .NET программистов, желающих создавать плагины для **Macroscop**. Все исходные файлы инструментария и примеров написаны для .NET на языке C#. В качестве среды разработки предполагается использование **Microsoft Visual Studio**. Для понимания данного документа требуется владение терминологией **Macroscop** на уровне опытного пользователя. При необходимости можно обратиться к инструкциям оператора и администратора, поставляемых в комплекте с **Macroscop**.

В **Macroscop SDK** каждый плагин представляет собой наследника одного из доступных в инструментарии базовых классов (интерфейсов) и решает определенный ряд задач. На данный момент в инструментарии имеются следующие основные базовые классы (интерфейсы), которые могут быть использованы внешними разработчиками:

Имя плагина	Название	Описание
ExternalAction	Действие	Базовый класс, позволяющий добавлять новые действия для сценариев и планировщика задач по расписанию
VideoAnalyst	Видеоаналитика	Базовый класс для осуществления видеоаналитики на сервере
MotionDetector	Детектор движения	Базовый класс для реализации детектора движения
Tracker	Трекер	Базовый класс для создания трекера
RTVisualiser	Визуализатор	Базовый класс визуализатора для графического отображения специфической информации на канале в приложении **Macroscop Клиент**
ClientMenuItem	Элемент меню	Базовый класс, позволяющий создавать в приложении **Macroscop Клиент** собственный подпункт в меню **Настройка**
EventProcessor	Процессор событий	Базовый класс процессора событий. Позволяет регистрировать и генерировать собственные события, получать события из **Macroscop**, а также выполнять команды в канале. Плагины данного типа могут быть использованы для осуществления интеграции с другими системами.
IRealTimeFrameReceiver	Получатель кадров	Интерфейс получателя кадров с IP-устройств. Позволяет получать с IP-устройств (камер) видео, звук, данные детекции движения; а также управлять поворотными камерами.

Все указанные типы базовых классов (интерфейсов), а также некоторые другие вспомогательные сущности подробно рассматривается в соответствующих главах данного документа. Все плагины существуют и работают в рамках канала **Macroscop**. Таким образом, все экземпляры плагинов по умолчанию изолированы друг от друга, однако имеется возможность обмениваться данными при необходимости через статические поля плагина. Как правило, все плагины решающие в совокупности одну сложную задачу, находятся в одной сборке .NET. Такая сборка является динамически подключаемой библиотекой (DLL), функционирующей в среде **Macroscop**. Подключение сборок и регистрация плагинов происходит на этапе запуска отдельных компонентов программного комплекса (см. Регистрация плагинов в Macroscop, стр. 6).

2. Быстрый старт – типовые задачи

1. **Интеграция IP-камер**

   Для подключения IP-устройства (камеры) достаточно реализовать плагин **Получатель кадров**. Сведения о данном типе плагина предоставлены в разделе Плагин-получатель кадров (стр. 23). Шаблон плагина размещен в папке с примерами, в проекте **Camera.csproj**.

2. **Интеграция со СКУД, ОПС, POS-терминалами и т.п.**

   Интеграция может быть выполнена через плагин **Процессор событий**, способный получать события от **Macroscop**, генерировать в **Macroscop** собственные события в процессе взаимодействия с другой системой, выполнять в **Macroscop** команды в канале (включение/выключение записи, установка пресетов, ввод-вывод с камер и др.), получать доступ к архиву **Macroscop**. Сведения о данном типе плагина предоставлены в разделе Плагин Процессор событий (стр. 21). Пример плагина размещен в папке с примерами, в проекте **EventProcessor.cproj**. Если требуется получать только видео и звук из **Macroscop**, то можно использовать более простой вариант, рассмотренный в разделе HTTP-интерфейс для получения видео (стр. 34); пример получения видео по HTTP размещен в папке с примерами, в проекте **HttpVideo.cproj**.

3. **Видеоаналитика**

   Любой алгоритм обработки видеопотока может быть реализован с помощью плагина **Видеоаналитика**. Все результаты работы данного плагина представляются событиями, которые могут быть далее интерпретированы плагинами **Визуализатор** и **Элемент меню**. Указанные плагины описаны в разделах Плагин Видеоаналитика (стр. 10), Плагин Визуализатор (стр. 17) и Плагин Элемент меню (стр. 19); пример использования размещен в папке с примерами, в проекте **Analyst.csproj**.

3. Плагины

В данной главе описан процесс регистрации плагинов, а также подробно рассмотрен каждый тип плагина. Наиболее важные фрагменты кода отражены в тексте.

3.1 Регистрация плагинов в Macroscop

В момент запуска отдельных приложений — компонентов **Macroscop** (**Сервер**/**Клиент**/**Конфигуратор**; далее в тексте — **хост**), — происходит поиск .NET сборок в папке **Plugins** запускаемого приложения. В каждой найденной сборке должен быть реализован интерфейс **IPlugin**, представленный ниже:

```
public interface IPlugin
{
        /// <summary>
        /// Возвращает уникальный идентификатор модуля
        /// </summary>
     Guid Id { get; }

        /// <summary>
        /// Возвращает название модуля
        /// </summary>
     string Name { get; }

        /// <summary>
        /// Возвращает название производителя модуля
        /// </summary>
        string Manufacturer { get; }

        /// <summary>
        /// Инициализация модуля.
        /// Вызывается хостом на этапе регистрации модуля в системе.
        /// </summary>
        /// <param name="host">Интерфейс хоста</param>
       void Initialize(IPluginHost host);
 }
```

Пример реализации данного интерфейса:

```
public class ModuleDef : IPlugin
{
    public Guid Id
    {
      get { return new Guid("17EE3457-8FC2-4C0F-B133-EF11D0C4F38C"); }
    }

    public string Name
    {
       get { return "Модуль поиска оставленных предметов"; }
    }

    public string Manufacturer
    {
        get { return "Macroscop"; }
    }

    public void Initialize(IPluginHost host)
    {
       host.RegisterAnalyst(typeof(AnalystExample));
       host.RegisterExternalEvent(typeof(ObjectLeftEvent));
    }
}
```

Как только указанный интерфейс был обнаружен хостом, вызывается метод инициализации (**Initialize**), в качестве аргумента которому передается интерфейс **IPluginHost**, предоставляющий сервисные методы хоста:

```
public interface IPluginHost
{
        /// <summary>
        /// Получает интерфейс менеджера протоколов
        /// </summary>
        /// <returns></returns>
        IMcLogMgr GetLogManager();

       /// <summary>
       /// Регистрация устройства.
       /// </summary>
   void RegisterDevType(DevType_RegInfo regInfo);

   /// <summary>
   /// Регистрация получателя кадров.
   /// </summary>
   void RegisterRTFR(RTFR_RegInfo regInfo);

        /// <summary>
        /// Регистрирует внешнее событие
        /// </summary>
        void RegisterExternalEvent(Type eventType);

       /// <summary>
       /// Регистрирует внешнее действие
       /// </summary>
       void RegisterExternalAction(Type actionType);

       /// <summary>
       /// Регистрирует пункт меню в Macroscop клиенте
       /// </summary>
       void RegisterMenuItem(Type menuItemType, List<Guid> requiredPluginIDs);

// ... Прочие, не показанные здесь, функции регистрации
}
```

Сервисные методы хоста предоставляют следующие возможности:

- Протоколирование работы плагинов с помощью интерфейса **IMcLogMgr**, получаемого соответствующим методом **GetLogManager()**.
- Регистрация в системе плагинов для решения различных задач.

Для того чтобы указать, при каких условиях нужно регистрировать плагин, необходимо создать текстовый файл с расширением ***.dep** и именем, совпадающим с названием dll-библиотеки, содержащей реализацию плагина. Данный файл, описывающий зависимости, может обладать, например, следующим содержанием:

```
<?xml version="1.0"?>
<PluginDependence xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <Files>
      <DependenceFilename>core.dll</DependenceFilename>
      <DependenceFilename>..\..\..\MacroscopSDK.dll</DependenceFilename>
      <DependenceFilename>..\..\abc*.dll</DependenceFilename>
   </Files>
   <DependenceRegistryKey>
      HKEY_LOCAL_MACHINE\SOFTWARE\Macroscop
   </DependenceRegistryKey>
   <DependenceRegistryValue>Path</DependenceRegistryValue>
</PluginDependence>
```

До версии Macroscop 1.9 данный файл является обязательным для регистрации плагина, после — нет. Данный файл должен быть в той же папке, что и сам плагин, для того, чтобы он имел действие. В теге **<Files>** может быть любое количество файлов. Поддерживается символ *, обозначающий любое количество любых символов. Пара тегов **<DependenceRegistryKey>** и **<DependenceRegistryValue>** имеют смысл только вместе.

3.2 Плагин Действие

Плагин действие позволяет расширить список функций в сценариях и планировщике задач по расписанию. Для создания такого типа плагина необходимо наследоваться от класса *ExternalAction*:

```
/// <summary>
/// Базовый класс действия.
/// </summary>
[Serializable]
public abstract class ExternalAction : IAction
{
   [NonSerialized]
   protected IActionHost actionHost;

   /// <summary>
   /// Инициализация. Вызывается хостом перед началом работы.
   /// </summary>
   /// <param name="host"></param>
   public virtual void Initialize(IActionHost host)
   {
        actionHost = host;
   }

    /// <summary>
    /// Пользовательский элемент настройки действия. Вызывается конфигуратором.
   /// Должен возвращать тип UserControl (WPF).
    /// </summary>
   public abstract object GetGUISettingsControl();
```

```
    /// <summary>
    /// Показывает правильно ли на данный момент
    /// сконфигурировано ли действие
    /// </summary>
    public abstract bool IsConfigurated
    {
        get;
    }

    /// <summary>
    /// Свойство показывает, привязано ли действие к
    /// конкретному каналу. Иными словами, влияет ли действие каким-либо
    /// образом на канал.
    /// </summary>
    public virtual bool IsChannelIndependent
    {
        get
        {
          //по умолчанию действие к каналу никак не привязано
          return true;
        }
    }

     /// <summary>
    /// Запуск действия на выполнение
    /// </summary>
    public abstract void Run(RawChannelEvent channelEvent);

    /// <summary>
    /// Выполнение команд в канале. Заполняется сервером, может использоваться в
    /// методе Run (см. выше)
    /// </summary>
    [NonSerialized]
    public ExecuteCommandDelegate ExecuteCommand;
}
```

В классе наследнике требуется задать следующие атрибуты:

- **ActionGUIName** — название действия, которое будет фигурировать в графическом интерфейсе в конфигураторе.
- **GuidAttribute** — идентификатор действия.

Опциально может быть задан атрибут **ActionNeedsEventArgument**, который показывает, что для вызова метода **Run** плагина со стороны сервера обязательно следует передавать объект события. Это также означает, что действие выполняется только по событию и не может выполняться в задачах по расписанию, при выполнении которых события не возникают.

Также требуется определить (переопределить) методы базового класса. Рассмотрим более подробно метод инициализации, в который передается интерфейс **IActionHost** со стороны хоста. Интерфейс выглядит следующим образом:

```
/// <summary>
/// Интерфейс, предоставляющий возможности
/// хоста при инициализации плагина действия.
/// </summary>
public interface IActionHost
{
    /// <summary>
    /// Получение информации о канале,
    /// в котором запущен текущий экземпляр
    /// плагина действия.
    /// </summary>
    RawChannelInfo GetChannelInfo();
```

```
    /// <summary>
    /// Сохраняет любой сериализуемый объект в конфигурацию
    /// Macroscop. Используется в конфигураторе.
    /// </summary>
    /// <param name="id">Идентификатор объекта</param>
    /// <param name="obj">Объект</param>
    void SaveObject(Guid id, object obj);

    /// <summary>
    /// Получает ранее из сериалзиованный объект из конфигурации.
    /// Используется в конфигураторе.
    /// </summary>
    /// <param name="id"></param>
    /// <returns></returns>
    object GetObject(Guid id);
}
```

Данный интерфейс позволяет получить информацию о канале, к которому привязан экземпляр плагина. Эта информация включает в себя имя канала и его идентификатор. Идентификатор должен быть использован при выполнении команд в канале (см. делегат **ExecuteCommand**). Подробная информация о доступных командах приведена в разделе Плагин Процессор событий (стр. 21) Кроме того, данный интерфейс с помощью методов **SaveObject** и **GetObject** позволяет сохранять и загружать любой сериализуемый объект по идентификатору на этапе задания пользователем конфигурации **Macroscop**. Такой механизм позволяет сохранять и извлекать настройки, которые являются одинаковыми для всех экземпляров плагинов.

Метод **GetGUISettingsControl** должен возвращать элемент типа **UserControl**, содержащий графический интерфейс для настройки данного экземпляра плагина в конфигураторе. Весь графический интерфейс должен быть выполнен с использованием WPF (Windows Presentation Foundation). Свойство **IsConfigurated** показывает, правильно ли пользователь сконфигурировал действие в графическом интерфейсе; если неправильно, то конфигурацию невозможно будет применить до тех пор, пока ошибки не будут исправлены.

Для регистрации **Действие** в системе, необходимо на этапе загрузки сборки с плагином в методе инициализации класса, реализующего интерфейс **IPlugin**, вызвать метод **RegisterExternalAction** интерфейса хоста (см. Регистрация плагинов в Macroscop, стр. 6).

3.3 Плагин Видеоаналитика

Данный тип плагина осуществляет покадровый анализ видеопотока, что позволяет создавать сервисные детекторы, например, детектор оставленных предметов, детектор саботажа и др. Для создания такого типа плагина, необходимо создать наследника базового класса **VideoAnalyst**:

```
/// <summary>
/// Класс видеоаналитика. Используется для обработки кадров и карт движения.
/// </summary>
public abstract class VideoAnalyst : IDisposable
{
   /// <summary>
   /// Инициализация аналитика. Вызывается хостом перед началом процесса обработки.
   /// </summary>
   /// <param name="Id">Идентификатор канала</param>
   /// <param name="archiveEventsReader">Интерфейс для доступа в архив</param>
   /// <param name="mdZones">Зоны детектирования движения.</param>
   /// <param name="settings">Настройки аналитика.</param>
   public abstract void Initialize(Guid Id, IArchiveEventsReader archiveEventsReader,
      List<MDZone> mdZones, PluginSettings settings);
```

```
/// <summary>
/// Метод обработки кадра и карты движения. Вызывается хостом.
/// </summary>
/// <param name="image">Кадр</param>
/// <param name="motionMap">Карта движения, может быть null</param>
/// <param name="background">Фон от детектора движения. Равен null, если
/// NeedBackground == false</param>
public abstract void Process(ImageData image, MotionMap motionMap,
   BackgroundImage background);

/// <summary>
/// Генерирует в канале ранее зарегистрированное внешнее событие.
/// Заполняется хостом. Вызывается аналитиком.
/// </summary>
public GenerateEventDelegate GenerateEvent;

/// <summary>
/// Поддерживает ли аналитик работу с частично продекодированными кадрами.
/// True означает, что на вход могут подаваться уменьшенные видеокадры,
/// False означает, что на вход всегда будут подаваться
///видеокадры с оригинальным разрешением.
/// </summary>
public virtual bool SupportsPartlyDecodedFrames
{
 get
 {
     return false;
 }
}

public virtual bool NeedBackground
{
 get
 {
     return false;
 }
}

/// <summary>
/// Поддерживаемый формат пикселя
/// </summary>
public virtual VAPixelFormat PixelFormat
{
 get
 {
     return VAPixelFormat.BGR24;
 }
}

/// <summary>
/// Выполняет команду.
/// </summary>
/// <param name="cmdObj"></param>
public abstract object ProcessCommand(object cmdObj);

/// <summary>
/// Заменяет детектор движения в канале на заданный.
/// Заполняется хостом. Может быть null.
/// </summary>
public ReplaceMotionDetectorDelegate ReplaceMotionDetector;
```

```
    /// <summary>
    /// Освобождение ресурсов
    /// </summary>
    public abstract void Dispose();

    /// <summary>
    /// Изменяет общие или специфические настройки аналитика, если это необходимо.
    /// Вызов метода должен приводить к появлению пользовательского окна настройки.
    /// Вызывается хостом (конфигуратором).
    public abstract PluginSettings SetSettings(ISettingsHost settingsHost,
       PluginSettings settings);
}
```

Класс наследник должен переопределить все абстрактные методы базового класса и, при необходимости, виртуальные свойства. Кроме того, производный класс должен иметь обязательный атрибут **PluginGUINameAttribute**, содержащего название аналитики, отображаемой в графической оболочке приложения **Macroscop Конфигуратор**. Если аналитика может быть настроен в конфигураторе, необходимо реализовать метод настройки **SetSettings**, и указать атрибут **PluginHasSettingsAttribute**.

Данные изображений подаются на вход аналитики в виде класса, описанного ниже:

```
public class ImageData
{
    public DateTime Timestamp;
    public byte[] Data;
    public System.Drawing.Size Size;
    public int Stride;
    public int BitPerPixel;
}
```

Поскольку каждый аналитический плагин прикрепляется пользователем к тому или иному каналу, в конфигураторе **Macroscop** объект настроек **PluginSettings** состоит из 2 частей:

1) настройки, связанные с текущим каналом безопасности
2) общие настройки аналитика, не зависящие от выбранного канала безопасности.

```
public struct PluginSettings
{
    /// <summary>
    /// Специфические настройки плагина, связанные с текущим каналом. Может быть null.
    /// Объект настроек должен быть сериализуемым.
    /// </summary>
    public object channelSpecificSettings;

    /// <summary>
    /// Общие настройки плагина. Не зависят от текущего канала. Может быть null.
    /// Объект настроек должен быть сериализуемым.
    /// </summary>
    public object generalSettings;
}
```

Если настройки аналитики едины для всех каналов, то достаточно заполнять только объект общих настроек. В противоположном случае, когда все настройки аналитики привязываются к конкретному каналу, достаточно заполнять только объект специфических настроек канала. Объекты общих и специфических настроек являются пользовательскими. Единственным требованием для них является возможность их сериализации, поскольку все настройки аналитики хранятся в общей конфигурации **Macroscop**.

Рассмотрим также метод **ProcessCommand**, который позволяет аналитике выполнять команды от плагина **Элемент меню** (см. Плагин Элемент меню, стр. 19). Данный метод получает команду в виде сериализуемого объекта, который формируется на стороне плагина **Элемент меню**. В качестве результата данный метод также должен возвращать сериализуемый объект, который впоследствии получит и обработает плагин **Элемент меню**. Такой механизм позволяет реализовать клиент-серверное взаимодействие, поскольку плагин **Видеоаналитика** работает на сервере, а плагин **Элемент меню** — всегда в клиенте.

В процессе обработки видеокадров аналитика должна передавать результаты своего анализа серверу с помощью генерации событий. Для этого необходимо заранее зарегистрировать на стороне хоста (см. Регистрация плагинов в Macroscop, стр. 6) внешнее пользовательское событие, содержащее описание всех необходимых полей для интерпретации результатов работы аналитика. Регистрируется событие с помощью метода **RegisterExternalEvent** интерфейса **IPluginHost** на этапе инициализации модуля. Пользовательское событие должно быть унаследовано от базового класса **RawChannelEvent**:

```
/// <summary>
/// Родительский класс в иерархии событий, происходящих в канале.
/// </summary>
[Serializable]
public abstract class RawChannelEvent
{
    /// <summary>
    /// ВременнАя метка события.
    /// </summary>
    public DateTime EventTime;

    /// <summary>
    /// Комментарий к событию.
    /// </summary>
    public string Comment;

    /// <summary>
    /// Является ли событие локальным.
    /// Локальные события не отправляются за пределы текущего хоста.
    /// </summary>
    public bool IsLocal = false;

    /// <summary>
    /// Сохранять ли событие в БД
    /// </summary>
    public bool Save = true;

    /// <summary>
    /// Режим сохранения события в БД
    /// </summary>
    public abstract EventArchiveSaveMode SaveMode
    {
        get;
    }

    public RawChannelEvent()
    {
        EventTime = DateTime.UtcNow;
    }
}
```

Каждое пользовательское событие должно иметь ряд обязательных атрибутов:

- **GuidAttribute** — определяет в явном виде уникальный идентификатор события;
- **Serializable** — позволяет выполнять сериализацию события.

Кроме того, имеется ряд необязательных (опциональных) атрибутов:

- **EventNameGUI** — задает название события в графическом интерфейсе хоста. Если атрибут не задан, то событие не будет отображаться в конфигураторе в разделе сценариев;
- **EventNameDatabase** — название таблицы в базе данных, в которую будут сохраняться экземпляры события; атрибут должен использоваться в случае, если у события есть поля, которые должны быть сохранены в БД (важно отметить, что свойство события **SaveMode** должно принимать в этом случае значения **Special** или **Both**);
- **EventGeneratesAlarmByDefault** — событие по умолчанию является тревожным, что означает автоматическую привязку к событию действия **Генерация тревоги** при создании нового канала в конфигураторе **Macroscop**.
- **EventGenerationFrequency** — указывает на частоту генерации данного события, требует параметр **EventGenerationFrequencyMode** (см. ниже). Данный атрибут влияет на работу сервера при сохранении событий в БД и имеет рекомендательный характер. Если атрибут не был указан, то по умолчанию используется режим **Middle**. Этот режим является предпочтительным. Режим **Low** не рекомендуется для использования, поскольку данные события записываются в особую, критичную для функционирования, базу данных.

```
/// <summary>
/// Частота генерации события.
/// </summary>
public enum EventGenerationFrequencyMode
{
    /// <summary>
    /// События генерируется с частотой,
    /// близкой к частоте анализа кадров
    /// </summary>
    High = 0,
    /// <summary>
    /// События генерируются с частотой, сопоставимой
    /// с половиной частоты анализа кадров
    /// </summary>
    Middle
}
```

Если пользовательское событие имеет поля, которые должны сохраняться в БД, необходимо для каждого из полей указывать атрибут **EventFieldSaveable**. В качестве параметров атрибут требует указать **Order** (порядковый номер поля, отсчет начинается с 0) и **IsIndexable** (флаг, указывающий, нужно ли индексировать данное поле).

Поддерживаются следующие типы полей события, к которым применимы указанные выше атрибуты: **int**, **bool**, **long**, **double**, **DateTime**, **string**, **Guid**, **byte[]**.

Свойство **SaveMode** определяет, будет ли событие сохраняться в БД. Событие может храниться как в базовой таблице, в которой находятся только поля класса **RawChannelEvent**, так и в специальной, содержащей все поля события. Также имеется возможность сохранять событие в обе таблицы. Использование базовой таблицы позволяет фиксировать сам факт возникновения события в системе. В будущем пользователь может читать из базовой таблицы события штатными средствами **Macroscop**.

Пользовательское событие может быть представлено следующим образом:

```
[GuidAttribute("389EDCE2-54BB-4C2C-9984-51B7516A5DDF")]
[EventNameGUI("Оставлен предмет")]
[EventDatabaseName("objectleft")]
[EventGeneratesAlarmByDefault]
[EventGenerationFrequency(EventGenerationFrequencyMode.Low)]
[Serializable]
public class ObjectLeftEvent : RawChannelEvent
{
   [EventFieldSaveable(0, true)]
   [EventFieldNameGUI("Предмет")]
   private string objectName;

   public ObjectLeftEvent(string objectName)
   {
       this.objectName = objectName;
   }

   public override EventArchiveSaveMode SaveMode
   {
       get  { return EventArchiveSaveMode.Both; }
   }
}
```

Для регистрации плагина **Видеоаналитика** необходимо на этапе загрузки сборки с плагином в методе инициализации класса, реализующего интерфейс **IPlugin**, вызвать метод **RegisterAnalyst** интерфейса хоста (см. Регистрация плагинов в Macroscop, стр. 6).

3.4 Плагин Детектор движения

Плагин данного типа позволяет реализовать для **Macroscop** альтернативный детектор движения. Для этого необходимо создать наследника класса **MotionDetector**:

```
/// <summary>
/// Детектор движения.
/// Базовый абстрактный класс для всех детекторов движения.
/// </summary>
public abstract class MotionDetector
{
    /// <summary>
    /// Зоны детектирования.
    /// </summary>
    protected List<MDZone> zones = new List<MDZone>();

    /// <summary>
    /// Добавляет 1 зону с маской, занимающей весь кадр.
    /// </summary>
    public void AddFullFrameZone()
    {
        MDZone zone = new MDZone();
        zones.Add(zone);
    }

    /// <summary>
    /// Добавляет 1 зону с маской, занимающей весь кадр, с указанием параметров  ///
зоны.
    /// </summary>
    public void AddFullFrameZone(float minObjWidth, float minObjHeight)
    {
        MDZone zone = new MDZone(0, minObjWidth, minObjHeight);
        zones.Add(zone);
    }
```

```
    /// <summary>
    /// Добавление зоны.
    /// </summary>
    /// <param name="zone">Добавляемая зона.</param>
    public void AddZone(MDZone zone)
    {
        zones.Add(zone);
    }

    /// <summary>
    /// Количество зон. Только чтение.
    /// </summary>
    public int ZonesCount
    {
        get
        {
            return zones.Count;
        }
    }

    #region Параметры необходимые для штатного детектора движения
        …
    #endregion

    /// <summary>
    /// Детектирование движения в заданном кадре.
    /// </summary>
    /// <param name="width">Ширина кадра.</param>
    /// <param name="height">Высота кадра.</param>
    /// <param name="bgr24bytes">Массив байт в формате bgr24, составляющих
      ///кадр.
      /// </param>
    /// <param name="timestamp">Временная метка.</param>
    /// <returns>Возвращает карту движения. Индекс > 0 означает наличие
      /// движения в рассматриваемом пикселе. Значение индекса в карте движения
      /// соответствует индексу движущегося объекта.
      ///</returns>
      public abstract MotionMap Detect(int width, int height, int offset,
                                       int stride, byte[] bgr24bytes, DateTime timestamp);
}
```

Поле **zones** заполняется сервером перед началом работы данного типа плагина через методы **AddFullFrameZone** и **AddZone**. Все поля в регионе **Параметры необходимые для штатного детектора движения** используются только встроенным детектором движения и должны быть оставлены без изменений.

Основная логика работы плагина концентрируется в методе **Detect**, на вход которому подается кадр в формате **bgr24** (каналы **Blue**, **Green**, **Red**, 8 бит на каждый канал). Результатом данного метода должна быть карта движения **MotionMap**. Для создания карты движения требуется два массива:

1) Карта наличия движения как такового. Представляет собой двумерный массив целых чисел. Каждое число массива (индекс, отличный от нуля) идентифицирует движущийся объект; **0** (нуль) свидетельствует об отсутствии движения.
2) Массив движущихся объектов. Доступ к элементам массива осуществляется по индексам карты движения.

Для регистрации плагина **Детектор движения** необходимо на этапе загрузки сборки с плагином в методе инициализации класса, реализующего интерфейс **IPlugin**, вызвать метод **RegisterMotionDetector** интерфейса хоста (см. Регистрация плагинов в Macroscop, стр. 6).

3.5 Плагин Трекер

Плагин данного типа позволяет сделать трекер для **Macroscop**. Для этого необходимо создать наследника класса **Tracker**:

```
/// <summary>
/// Базовый класс для всех трэкеров.
/// <summary>
/// Базовый класс для всех трэкеров.
/// </summary>
public abstract class Tracker
{
   /// <summary>
   /// Обработка кадра и карты движения.
   /// Результатом обработки является измененная карта движения, на которой корректно
   /// проставлены индексы движущихся объектов.
   /// </summary>
   /// <param name="width">Ширина кадра.</param>
   /// <param name="height">Высота кадра.</param>
   /// <param name="offset">Отступв в байтах до начала кадра в массиве байт.</param>
   /// <param name="stride">Страйд.</param>
   /// <param name="bgr24bytes">Байты с пикселами кадра.</param>
   /// <param name="timestamp">Временная метка.</param>
   /// <param name="detectedMap">Продетектированная карта движения.
   /// Значение может измениться после обработки.</param>
   public abstract void Process(int width, int height, int offset, int stride,
         byte[] bgr24bytes, DateTime timestamp, ref MotionMap detectedMap);
}
```

Основная логика работы реализуется методом **Process**. Данному методу на вход поступает кадр формата **bgr24** (каналы **Blue**, **Green**, **Red**, 8 бит на каждый канал) и карта движения **MotionMap**, полученная плагином **Детектор движения**. Результатом работы плагина являются идентификаторы сопровождаемых объектов, которые должны быть проставлены в карте движения.

3.6 Плагин Визуализатор

Плагин данного типа позволяет графически отображать информацию (например, рамки движущихся объектов, распознанные номера, лица и др.) содержащуюся в событиях, которые поступают в каналы приложения **Macroscop Клиент**. Для создания данного типа плагина необходимо создать наследника класса **RTVisualiser**:

```
/// <summary>
/// Класс визуализатора.
/// </summary>
public abstract class RTVisualiser
{
   /// <summary>
   /// Панель для отрисовки примитивов, текста и другой информации на канале.
   /// </summary>
   protected IDrawingPanel drawingPanel;

   /// <summary>
   /// Контейнер графических элементов. Позволяет размещать отдельные
   /// UserControl'ы в канале.
   /// </summary>
   protected Panel controlsContrainer;

   protected IPluginToolSet pluginToolset;
```

```
    /// <summary>
    /// Инициализация визуализатора. Вызывается хостом.
    /// </summary>
    /// <param name="Id">Идентификатор канала</param>
    /// <param name="pluginToolset">Интерфейс доступа в архив, для отправки команд,
    /// для подписки на события в системе.</param>
    /// <param name="drawingPanel">Панель для рисования.</param>
    /// <param name="controlsContrainer"></param>
    public virtual void Initialize(Guid Id, IPluginToolSet pluginToolset,
           IDrawingPanel drawingPanel, Panel controlsContrainer)
    {
     this.pluginToolset = pluginToolset;
     this.drawingPanel = drawingPanel;
     this.controlsContrainer = controlsContrainer;
    }

    /// <summary>
    /// Обработка события визуалайзером. Специфическая отрисовка результатов события.
    /// </summary>
    /// <param name="channelId">Идентификатор канала</param>
    /// <param name="chEv">Событие.</param>
    /// <param name="isAlarm">Является ли событие тревожным</param>
    public abstract void ProcessEvent(Guid channelId, RawChannelEvent chEv,
           bool isAlarm);

    /// <summary>
    /// Делегат для регистрации подпункта во всплывающем меню
    /// канала в клиенте Macroscop.
    /// Заполняется хостом. Вызывается визуализатором.
    /// </summary>
    /// <param name="item"></param>
    public RegisterChannelMenuItemHandler RegisterChannelMenuItem;

    /// <summary>
    /// Очистка всего, что нарисовано визуализатором.
    /// </summary>
    public abstract void Clear();

    /// <summary>
    /// Освобождение всех ресурсов. В перегружаемых метода в наследниках обязательно
    /// вызывать Release базового класса.
    /// </summary>
    public virtual void Release()
    {
     drawingPanel = null;
     controlsContrainer = null;
     pluginToolset = null;
     RegisterChannelMenuItem = null;
    }
}
```

Каждый плагин **Визуализатор** должен определять метод **ProcessEvent**, на вход которому передается очередное событие, поступившее в канал. Все события генерируются **Macroscop** или другими плагинами.

Плагин может визуализировать любые типы событий, а их фильтрацию можно осуществить с помощью оператора **if** и ключевого слова **is**, например:

```
if (chEv is CounterEvent)
{
    ...
}
```

Визуализатор может регистрировать свой подпункт во всплывающем меню, отображаемом при щелчке правой кнопки мыши на канале в приложении **Macroscop Клиент**. Это позволяет изменять логику работы визуализатора в зависимости от предпочтений пользователя. Данная возможность предоставляется делегатом **RegisterChannelMenuItem**.

Для регистрации плагина-визуализатора в системе, необходимо на этапе загрузки сборки с плагином в методе инициализации класса, реализующего интерфейс **IPlugin**, вызвать метод **RegisterRTVisualiser** интерфейса хоста (см. Регистрация плагинов в Macroscop, стр. 6).

3.7 Плагин Элемент меню

Плагин данного типа позволяет создавать собственный графический интерфейс, открываемый через подпункт меню **Настройка** приложения **Macroscop Клиент**.

Типичное применение данного плагина заключается в организации клиент-серверного взаимодействия с другими плагинами. Например, плагин может обрабатывать результаты работы аналитических плагинов, работающих на сервере. Для создания плагина-элемента меню необходимо создать наследника класса **ClientMenuItem**:

```
/// <summary>
/// Класс элемент меню. Используется в клиенте для добавления
/// подпунтка в меню кнопки "Настройка".
/// </summary>
public abstract class ClientMenuItem : IDisposable
{
    private bool isEnabled = true;

    /// <summary>
    /// Инициализация плагина.
    /// </summary>
    /// <param name="toolSet"></param>
    public abstract void Initialize(IPluginToolSet toolSet);

    /// <summary>
    /// Название элемента меню.
    /// </summary>
    public abstract string Name
    {
        get;
    }

    /// <summary>
    /// Включен ли в данный момент элемент меню.
    /// </summary>
    public bool IsEnabled
    {
        get
        {
            return isEnabled;
        }
        set
        {
            isEnabled = value;
        }
    }

    /// <summary>
    /// Метод обработки щелчка мыши (нажатия клавиатуры) пользователем
    /// </summary>
    public abstract void OnClicked();
```

```
    /// <summary>
    /// Метод освобождения ресурсов.
    /// </summary>
    public abstract void Dispose();
}
```

В классе-наследнике необходимо определить метод инициализации, в который со стороны хоста (приложения **Macroscop Клиент**) передается интерфейс с сервисными функциями. Эти функции позволяют получить идентификаторы каналов и их имена в текущей конфигурации. Кроме того, они предоставляют доступ к архиву **Macroscop**, возможность отправлять команды аналитическим плагинам на сервере и получать результат их исполнения. Имеется возможность подписываться на любые события, возникающие в системе. Интерфейс представляется следующим образом:

```
/// <summary>
/// Интерфейс предоставляемый хостом для
/// доступа в архив, для отправки команд, для
/// подписки на события в системе.
/// </summary>
public interface IPluginToolSet
{
    /// <summary>
    /// Получает интерфейс для работы с архивом
    /// </summary>
    /// <returns></returns>
    IArchiveEventsReader GetArchiveReader();

    /// <summary>
    /// Отправляет команду плагину,
    /// работающего на сервере.
    /// </summary>
    /// <param name="pluginId">Идентификатор плагина</param>
    /// <param name="channelsId">Идентификаторы каналов</param>
    /// <param name="cmdObj">Команда</param>
    /// <returns>Возвращает результат от каждого канала.
      /// Ключ - идентификатор канала.
    /// Значение - результат выполнения команды.</returns>
    Dictionary<Guid, object> SendChannelsCommand(Guid pluginId,
             List<Guid> channelsId, object cmdObj);

    /// <summary>
    /// Устанавливает/удаляет обработчик событий, возникающих в канале.
    /// </summary>
    /// <param name="subscrId">Идентификатор подписчика</param>
    /// <param name="channelId">Идентификатор канала</param>
    /// <param name="eventsHandler">Обработчик. Если null, то
      /// ранее установленный обработчик удаляется.</param>
    void SetEventsHandler(Guid subscrId,
             Guid channelId, EventsHandler eventsHandler);

    /// <summary>
    /// Получает настройки плагина с указанным идентификатором на данном канале.
    /// </summary>
    /// <param name="channelId">Идентификатор канала</param>
    /// <param name="pluginId">Идентификатор плагина</param>
    /// <returns></returns>
    PluginSettings GetPluginSettings(Guid channelId, Guid pluginId);
}
```

Метод **GetArchiveReader** предоставляет интерфейс доступа к архиву и к информации о текущих каналах в конфигурации. Подробные сведение о данном интерфейсе размещены в исходных кодах **Macroscop SDK**.

Метод **SendChannelsCommand** отправляет команду разным экземплярам плагина одного и того же типа по указанному списку каналов и получает результаты исполнения команды. Метод **SetEventsHandler** устанавливает или удаляет обработчик событий на заданном канале.

В методе **OnClicked** должна быть реализована логика работы с пользователем через графический интерфейс. Данный метод вызывается в момент щелчка клавиши мыши (клавиатуры) по пункту меню, связанному с плагином.

Для регистрации плагина **Элемент меню** необходимо на этапе загрузки сборки с плагином в методе инициализации класса, реализующего интерфейс **IPlugin**, вызвать метод **RegisterMenuItem** интерфейса хоста (см. Регистрация плагинов в Macroscop, стр. 6).

3.8 Плагин Процессор событий

Плагин **Процессор событий** позволяет получать и обрабатывать сигналы от сторонних систем. В процессе обработки сигналов данный тип плагина может как выполнять команды в канале, в котором он существует, так и генерировать в этом канале события. Данный плагин создается путем наследования от базового класса **EventProcessor**:

```
/// <summary>
/// Класс плагина для обработки событий системы, генерации команд и своих событий
/// </summary>
public abstract class EventProcessor
{
   /// <summary>
   /// Инициализация. Вызывается хостом.
   /// </summary>
   /// <param name="archiveReader">Интерфейс доступа к архиву</param>
   /// <param name="channelSpecificSettings">Настройки плагина</param>
   public abstract void Initialize(IArchiveEventsReader archiveReader,
         PluginSettings settings);

   /// <summary>
   /// Генерирует в каналее ранее зарегистрированное внешнее событие. Заполняется
   /// хостом. Вызывается плагином.
   /// </summary>
   public GenerateEventDelegate GenerateEvent;

   /// <summary>
   /// Запускает заданную команду на выполнение в канале. Заполняется хостом.
    /// Вызывается плагином.
   /// </summary>
   public ExecuteCommandExDelegate ExecuteCommand;

   /// <summary>
   /// Позволяет подписываться на события, происходящие в канале. Заполняется
    /// плагином при необходимости на этапе инициализации.
   /// </summary>
   public ReceiveEventDelegate OnChannelEventReceived;

   /// <summary>
   /// Изменяет общие или специфические настройки, если это необходимо.
   /// Вызов метода должен приводить к появлению пользовательского окна настройки.
   /// Вызывается хостом (конфигуратором).
   /// </summary>
   /// <param name="settings">Текущие настройки плагина.</param>
   /// <returns>Новые настройки плагина.</returns>
   public abstract PluginSettings SetSettings(PluginSettings settings);
}
```

Производный класс должен иметь обязательный атрибут **PluginGUINameAttribute**, содержащего название аналитики, отображаемой в приложении **Macroscop Конфигуратор**. Если аналитика может быть настроена в конфигураторе, следует реализовать метод настройки **SetSettings** и указать атрибут **PluginHasSettingsAttribute**.

В метод инициализации **Initialize** со стороны хоста передается интерфейс для доступа в архив **Macroscop** и объект настроек плагина. Поскольку каждый плагин прикрепляется пользователем к тому или иному каналу, объект настроек **PluginSettings** состоит из 2 частей:

1) настройки, связанные с текущим каналом безопасности;
2) общие настройки аналитики, не зависящие от выбранного канала безопасности.

```
public struct PluginSettings
{
    /// <summary>
    /// Специфические настройки плагина, связанные с текущим каналом. Может быть null.
    /// Объект настроек должен быть сериализуемым.
    /// </summary>
    public object channelSpecificSettings;

    /// <summary>
    /// Общие настройки плагина. Не зависят от текущего канала. Может быть null.
    /// Объект настроек должен быть сериализуемым.
    /// </summary>
    public object generalSettings;
}
```

Если настройки едины для всех каналов, то достаточно заполнять только объект общих настроек. В противоположном случае, когда все настройки плагина привязываются к конкретному каналу, достаточно заполнять только объект специфических настроек канала. Объекты общих и специфических настроек являются пользовательскими; единственным требованием к ним является возможность сериализации, поскольку все настройки плагина хранятся в общей конфигурации **Macroscop**.

Если плагину в качестве результата своей работы необходимо выполнить определенную команду в канале (например, включить/выключить запись, установить пресет на камере, повернуть камеру и т.д.), требуется создать объект команды. На текущий момент реализованы следующие команды:

- **RawEnableRecordingCommand** — включает запись в канале, опционально указывается интервал записи;
- **RawDisableRecordingCommand** — выключает запись в канале;
- **RawGoToPresetPtzCommand** — устанавливает пресет на камере;
- **RawGoHomePtzCommand** — устанавливает домашнее положение камеры;
- **RawStopPtzCommand** — останавливает выполнение ptz команд на камере;
- **RawMovePtzCommand** — перемещение на шаг;
- **RawZoomPtzCommand** — относительное приближение (зум);
- **RawStartMovePtzCommand** — непрерывное (желательно плавное) движение;
- **RawMoveToPtzCommand** — поворачивает камеру таким образом, что указанная точка оказывается в центре области кадра;
- **RawSetOutputIOCommand** — устанавливает уровень сигнала на выходе камеры;
- **RawSetOutputPulsesIOCommand** — генерирует последовательность импульсов на выходе камеры;

Плагин **Процессор событий** может подписываться на события, возникающие в канале. Для этого на этапе своей инициализации плагин должен заполнить делегат **OnChannelEventReceived**. Кроме того, плагин может генерировать собственные события в канале.

Для регистрации плагина **Процессор событий** необходимо на этапе загрузки сборки с плагином в методе инициализации класса, реализующего интерфейс **IPlugin**, вызвать метод **RegisterEventProcessor** интерфейса хоста (см. Регистрация плагинов в Macroscop, стр. 6).

3.9 Плагин Получатель кадров

Плагин данного типа позволяет получать видео и звук с IP-устройств (камер), данные о детекции движения, управлять поворотными камерами, управлять входами/выходами (I/O) IP-устройств. Для решения подзадачи получения кадров необходимо реализовать интерфейс **IRealTimeFrameReceiver**:

```
/// <summary>
/// Интерфейс получения кадров реального времени.
/// Используется для создания плагинов, получающих кадры
/// с IP-камер.
/// </summary>
public interface IRealTimeFrameReceiver
{
    /// <summary>
    /// Обработчик события о приходе нового кадра.
    /// Вызывается реализующей интерфейс стороной.
    /// На событие подписывается хост.
    /// </summary>
    event NewRawFrameEventHandler NewRawFrame;

    /// <summary>
    /// Обработчик событий. Вызывается реализующей интерфейс стороной.
    /// На обработчик подписывается хост.
    /// </summary>
    event NewRawEventHandler NewEvent;

    /// <summary>
    /// Обработчик добавления записей в лог. Уведомляет хоста, о том, что
    /// поле ConnectionLog (см. ниже) изменилось.
    /// Лог просматривается в конфигураторе при нажатии на кнопку "Журнал"
    /// Вызывается реализующей интерфейс стороной.
    /// На обработчик подписывается хост.
    /// </summary>
      event EventHandler NewLogRecord;

    /// <summary>
    /// Флаг, указывающий нужно ли писать в лог.
    /// Изменяется хостом.
    /// </summary>
      bool IsWritingConnectionLog
      {
        get;
        set;
      }

    /// <summary>
    /// Текущее содержимое лога подключения.
    /// </summary>
      string ConnectionLog
      {
        get;
      }

    /// <summary>
    /// Является ли поток активным
    /// </summary>
    /// <param name="streamType">Тип потока</param>
    /// <returns></returns>
    bool IsStreamActive(ChannelStreamTypes streamType);
```

```
    /// <summary>
    /// Запускает поток указанного типа для получения кадров
    /// </summary>
    /// <param name="streamType"></param>
    void StartStream(ChannelStreamTypes streamType);

    /// <summary>
    /// Останавливает поток указаного типа
    /// </summary>
    /// <param name="streamType"></param>
    void StopStream(ChannelStreamTypes streamType);

    /// <summary>
    /// Отправляет звук на устройство (случай реализации дуплексного звука).
    /// </summary>
    /// <param name="soundData"></param>
    void SendSound(byte[] soundData);

    /// <summary>
    /// Освобождает все ресурсы. Закрывает все потоки.
    /// </summary>
    void Release();
}
```

При реализации данного интерфейса необходимо создать механизм работы с потоками данных. Потоки данных представляют собой или последовательность кадров определенного типа, или последовательность событий. В текущей версии **Macroscop SDK** имеются следующие типы потоков данных:

```
/// <summary>
/// Типы потоков канала.
/// </summary>
public enum ChannelStreamTypes : ulong
{
    /// <summary>
    /// Главный поток видео
    /// </summary>
    MainVideo = 1,

       /// <summary>
       /// Альтернативный поток видео
       /// </summary>
       AlternativeVideo = 2,

    /// <summary>
    /// Поток звука, идущий с камеры.
    /// </summary>
    MainSound = 4,

       /// <summary>
       /// Альтернативный звуковой поток
       /// </summary>
       AlternativeSound = 8,

    /// <summary>
    /// Обратный поток звука.
    /// </summary>
    OutputSound = 16,

    /// <summary>
    /// Поток данных детекции движения.
    /// </summary>
    MotionDetection = 32,
```

```
            /// <summary>
            /// Поток данных от системы ввода-вывода камеры
            /// </summary>
            IO = 64,
}
```

В зависимости от возможностей подключаемого устройства, а также от настроек канала в конфигураторе, необходимо генерировать те или иные потоки данных.

Потоки данных **MainVideo** и **AlterntaiveVideo** состоят из видеокадров **RawVideoFrame**, получаемых с камеры.

Потоки данных **MainSound** и **AlternativeSound**, **OutputSound** состоят из последовательности звуковых кадров **RawSoundFrame**. Поток данных **MotionDetection** — из последовательности событий **RawChEv_MDresults**, **RawChEv_NoDetection**.

Поток I/O — состоит из событий **RawChEv_InputSignalLevelChanged**.

Запуск потоков данных происходит со стороны хоста с помощью метода **StartStream**, в котором производится необходимая инициализация очередного потока. Метод должен сразу же возвращать управление хосту, а длительные операции (операции ввода/вывода) выполнять в отдельных потоках. Остановка потоков и проверка их активности методами **StopStream** и **IsStreamActive** также вызывается со стороны хоста и должны выполняться немедленно без выполнения долгосрочных операций. Результаты своей работы потоки должны возвращать хосту через вызовы обработчиков кадров (**NewRawFrame**) и событий (**NewEvent**).

Например, если IP-устройство шлет MJPEG-кадры, то поток данных **MainVideo** (или **AlternativeVideo**) должен вызывать **NewRawFrame** и в качестве одного из аргументов передавать видеокадр **RawMJPEGFrame**:

```
/// <summary>
/// MJPEG кадр
/// </summary>
 [Serializable]
public class RawMJPEGFrame : RawVideoFrame
{
      public RawMJPEGFrame() { }

      /// <summary>
      /// Данные кадра
      /// </summary>
      /// <param name="data"></param>
      public RawMJPEGFrame(byte[] data)
      {
            Data = data;
      }
}
```

Аналогично для потока данных **MainSound** (или **AlterntaiveSound**) и звука в формате G.711U, необходимо передавать кадр **RawG711UFrame**:

```
/// <summary>
/// Кадр стандарта G.711U
/// </summary>
[Serializable]
public class RawG711UFrame : RawSoundFrame
{

     /// <summary>
     /// Данные кадра
     /// </summary>
     /// <param name="data"></param>
        public RawG711UFrame(byte[] data)
        {
             this.Data = data;
             this.samplesRate = 8000;
             this.bitsPerSample = 16;
             this.channels = 1;
             this.bitrate = 64000;
       }
}
```

После создания кадра подходящего типа следует обязательно заполнить следующие поля базового класса **RawFrame**:

- **Id** — идентификатор кадра, состоящий из двух частей. Первая часть: идентификатор последовательности (генерируется случайным образом перед началом работы потока получения данных); вторая часть: порядковый номер кадра.
- **Timestamp** — временная метка кадра, задается в формате UTC.

В кодеках MPEG-4 и H.264:

- для P-кадров обязательно требуется заполнять поле **Dependencies**, содержащее идентификаторы всех кадров, от которых данный кадр зависит;
- в каждом I-кадре требуется указывать инициализирующую информацию для декодера в поле **SpecInitData**.

Пример создания MJPEG-кадра:

```
RawMJPEGFrame jpegFrame = new RawMJPEGFrame(frameData);
jpegFrame.Id.SeqId = videoSeqId;
jpegFrame.Id.NumInSeq = videoNumInSeq++;
jpegFrame.TimeStamp = DateTime.UtcNow;
```

Где изначально были заданы:

```
//Идентификатор последовательности видео кадров
Guid videoSeqId = Guid.NewGuid();
//Номер очередного видео кадра в последовательности
long videoNumInSeq = 0;
```

Пример H.264 I-кадра:

```
RawH264_I_Frame iFrame = new RawH264_I_Frame(frameData);
iFrame.Id.SeqId = videoSeqId;
iFrame.Id.NumInSeq = videoNumInSeq++;
iFrame.TimeStamp = DateTime.UtcNow;
iFrame.SpecInitData = initData;
```

Где **initData** — инициализирующая информация для декодера.

Пример H.264 P-кадра:

```
RawH264_P_Frame pFrame = new RawH264_P_Frame(frameData);
pFrame.Id.SeqId = videoSeqId;
pFrame.Id.NumInSeq = videoNumInSeq++;
pFrame.TimeStamp = DateTime.UtcNow;
pFrame.Dependencies = frameDependencies;
```

Где **frameDependencies** — массив идентификаторов кадров, от которых зависит данный кадр. Иными словами совокупность кадров, которая должна быть продекодирована перед декодированием данного кадра.

Пример G.711U кадра:

```
RawG711UFrame g711Frame = new RawG711UFrame(frameData);
g711Frame.Id.SeqId = audioSeqId;
g711Frame.Id.NumInSeq = audioNumInSeq++;
g711Frame.TimeStamp = DateTime.UtcNow;
```

В случае если в процессе получения потоков данных произошел обрыв связи с IP- устройством, плагин **Получатель кадров** должен уведомить об этом хост путем генерации события **RawChEv_NoDataConnection** с обязательно заполненным полем **StreamTypes**, показывающим, в каких потоках данных произошел обрыв соединения.

Для регистрации получателя кадров в системе необходимо заполнить регистрационную информацию устройства **DevType_RegInfo** и регистрационную информацию плагина **RTFR_RegInfo**. Ниже приведено описание класса **DevType_RegInfo**:

```
/// <summary>
/// Регистрационная информация устройства.
/// используется при регистрации плагина,
/// получающего кадры.
/// </summary>
public class DevType_RegInfo
{
    /// <summary>
    /// Идентификатор устройства.
    /// </summary>
    public Guid DeviceTypeGuid;

    /// <summary>
    /// Имя производителя.
    /// </summary>
    public string DevTypeBrandName;

    /// <summary>
    /// Имя устройства.
    /// </summary>
    public string DevTypeModelName;

    /// <summary>
    /// Список возможностей устройства.
    /// </summary>
    public DevType_Capabilities Capabilities;

    /// <summary>
    /// Список доступных разрешений для данного устройства.
    /// </summary>
    public List<VideoResolutions> AvailableResolutions = new
                                                    List<VideoResolutions>();

    /// <summary>
    /// Делегат, позволяющий хосту изменять настройки на камере/видеосервере.
      /// </summary>
      public SetDeviceParametersDelegate SetDeviceParameters;
```

```
        /// <summary>
        /// Получение интерфейса IRealTimeFrameReceiver.
        /// </summary>
        public GetRTFRDelegate GetRTFR;

    /// <summary>
    /// Получение PTZ интерфейса.
    /// </summary>
    public GetPtzControllerDelegate GetPtzController;

        /// <summary>
        /// Получение I/O интерфейса.
        /// </summary>
        public GetIOControllerDelegate GetIOController;
}
```

Конкретная реализация интерфейса **IRealTimeFramReceiver** должна возвращаться делегатом **GetRTFR**, вызываемого хостом. В качестве аргументов этому делегату передаются параметры подключения (**ConnectionParameters**) и настройки потоков (**SubStreamParameters**), которые были заданы пользователем в конфигурации канала.

Возможности IP-устройства, с которым работает плагин **Получатель кадров**, описываются полем **Capabilities**:

```
/// <summary>
/// Перечень возможностей устройства
/// </summary>
public enum DevType_Capabilities : ulong
{

       /// <summary>
    /// Устройство работает только с камерами.
    /// </summary>
    SupportsCameras = 1,
    /// <summary>
    /// Устройство поддерживает камеры и видеосерверы.
    /// </summary>
    SupportsCamerasAndServers = 2,
    /// <summary>
    /// Устройство поддерживает работу с альтернативным потоком.
    /// </summary>
    SupportsAlternativeVideoStream = 4,
    /// <summary>
    /// Параметры (разрешение, фпс, компрессия), описываемые массивами
        /// SupportedDeviceParameters/SupportedExtraParameters, не зависят от формата
        /// потока (одинаковы для mjpeg, mpeg4, h264).
    /// </summary>
    DeviceParametersFormatIndependent = 8,
}
```

Если требуется решить задачу управления поворотной камерой или ее входами/выходами (I/O), необходимо реализовать соответствующие интерфейсы **IPtzController** и/или **IIOController**:

```
/// <summary>
/// Унифицированный интерфейс реализации управления поворотной камерой.
/// </summary>
public interface IPtzController
{
    #region ----------------- БАЗОВАЯ ЧАСТЬ ------------------

    /// <summary>
    /// Инициализация камеры.
    /// </summary>
    /// <returns></returns>
    void Initialization();

    /// <summary>
    /// Возвращает возможности данной камеры.
    /// </summary>
    /// <returns></returns>
    PtzCapabilities GetCapabilities();

    /// <summary>
    /// Возвращает названия пресетов, установленных на камере.
    /// Количество элементов результирующего массива соответствует количеству
    /// пресетов.
    /// Каждому пресету соответствует номер, равняющийся индексу в массиве.
    /// </summary>
    /// <returns>Названия пресетов, установленных на камере.</returns>
    string[] GetPresetsNames();

    /// <summary>
    /// Устанавливает пресет по его номеру.
    /// </summary>
    /// <param name="presetIndex">Номер пресета.</param>
    void SetPresetPosition(int presetIndex);

    /// <summary>
    /// Устанавливает камеру в "домашнее" положение.
    /// </summary>
    void MoveToHome();

    /// <summary>
    /// Прекращает выполнение любой команды PTZ.
    /// </summary>
    void Stop();

    /// <summary>
    /// Перемещение на шаг.
    /// </summary>
    /// <param name="panSpeed">Скорость по горизонтали. Интервал от -100 до
    /// 100.</param>
    /// <param name="tiltSpeed">Скорость по вертикали. Интервал от -100 до
    /// 100.</param>
    void StepMove(int panSpeed, int tiltSpeed);
```

```
    /// <summary>
    /// Непрерывное(желательно плавное) движение.
    /// </summary>
    /// <param name="panSpeed">Скорость по горизонтали. Интервал от -100 до
    /// 100.</param>
    /// <param name="tiltSpeed">Скорость по вертикали. Интервал от -100 до
    /// 100.</param>
    void ContiniousMove(int panSpeed, int tiltSpeed);

    /// <summary>
    /// Относительное приближение.
    /// </summary>
    /// <param name="step">Шаг от 1 до 100.</param>
    void StepZoomIn(int step);

    /// <summary>
    /// Относительное отдаление.
    /// </summary>
    /// <param name="step">Шаг от 1 до 100.</param>
    void StepZoomOut(int step);

    /// <summary>
    /// Непрерывное (желательно плавное) приближение.
    /// </summary>
    /// <param name="speed">Скорость. Интервал от 1 до 100.</param>
    void ContiniousZoomIn(int speed);

    /// Непрерывное (желательно плавное) отдаление.
    /// </summary>
    /// <param name="speed">Скорость. Интервал от 1 до 100.</param>
    void ContiniousZoomOut(int speed);

    /// <summary>
    /// Возвращает максимальное увеличение камеры.
    /// </summary>
    /// <returns>Максимальное увеличение камеры. Если функция не
    ///поддерживается, возвращает отрицательное число.</returns>
    double GetMaxZoomFactor();

/// <summary>
    /// Возвращает текущее увеличение камеры.
    /// </summary>
    /// <returns>Текущее увеличение камеры. Если функция не поддерживается,
    /// возвращает отрицательное число.</returns>
    double GetCurrentZoomFactor();

    /// <summary>
    /// Устанавливает абсолютное увеличение. Ничего не делает, если камера это
    /// не поддерживает. См. PtzCapabilities.
    /// </summary>
    void SetZoomFactor(double factor);

    /// <summary>
    /// Устанавливает максимальное увеличение.
    /// </summary>
    void ZoomTele();
```

```
    /// <summary>
    /// Устанавливает минимальное увеличение.
    /// </summary>
    void ZoomWide();

    #endregion

    #region ----------------- ДОПОЛНИТЕЛЬНАЯ ЧАСТЬ ------------------

    /// <summary>
    /// Поворачивает камеру таким образом, что указанная точка оказывается в
    /// центре области кадра.
    /// </summary>
    /// <param name="point">Точка изображения, которую необходимо поместить в
    /// центр путём поворота камеры. Задаётся в пикселях.
    /// Начало отсчета(0,0) - левый верхний угол кадра о</param>
    /// <param name="frameSize">Размер кадра в пикселях</param>
    void MoveTo(System.Drawing.Point point, System.Drawing.Size frameSize);

    /// <summary>
    /// Поворачивает и масштабирует камеру таким образом,
    /// что указанный прямоугольник занимает всю область кадра.
    /// Если пропорции прямоугольника не соответсвуют пропорциям кадра, то
    /// масштабирование производится таким образом, чтобы прямоугольник весь
       ///вошёл в кадр.
    /// Центр прямоугольника помещается в центр кадра.
    /// </summary>
    /// <param name="rect">Прямоугольник, задается в пикселях</param>
    /// <param name="frameSize">Размер кадра в пикселях</param>
    void ShowRect(System.Drawing.Rectangle rect, System.Drawing.Size frameSize);

    #endregion
}

/// <summary>
/// Уинифицированный интерфейс реализации управления входами/выходами камеры
/// </summary>
public interface IIOController
{
   /// <summary>
   /// Инициализация
   /// </summary>
   /// <returns></returns>
   void Initialization();

  /// <summary>
  /// Возвращает возможности данной камеры.
  /// </summary>
  /// <returns></returns>
  IOCapabilities GetCapabilities();

  /// <summary>
  /// Устанавливает заданное значение на выходе
  /// </summary>
  /// <param name="portID">Номер выхода</param>
  /// <param name="value">1 или 0</param>
  void SetOutput(int portID, int value);
```

```
  /// <summary>
  /// Выдает последовательность импульсов (ШИМ) на указанном выходе.
  /// Поддерживается не всеми камерами.
  /// </summary>
  /// <param name="portID">Номер выхода</param>
  /// <param name="pulses">Массив импульсов</param>
  void SetOutput(int portID, IOPulse[] pulses)
}
```

Возможности поворотного устройства (I/O контроллера) должны возвращаться методом **GetCapabilities()** для информирования хоста о том, какие методы опциональных возможностей (если поддерживаются устройством) реализованы в интерфейсе.

Регистрационную информацию **DevType_RegInfo** устройства необходимо указывать на этапе загрузки сборки с плагином в методе инициализации класса, реализующего интерфейс **IPlugin**, вызывая метод **RegisterDevType** интерфейса хоста (см. Регистрация плагинов в Macroscop, стр. 6).

Помимо **DevType_RegInfo**, необходимо также заполнить информацию о получателе кадров **RTFR_RegInfo**:

```
/// <summary>
/// Регистрационная информация о получателе кадров
/// </summary>
public class RTFR_RegInfo
{
    /// <summary>
    /// Идентификатор устройства
    /// </summary>
    public Guid DeviceTypeGuid;

       /// <summary>
       /// Формат потока данных
       /// </summary>
    public VideoStreamFormats StreamFormat;

       /// <summary>
       /// Используемый протокол подключения
       /// </summary>
    public NetworkConnectionTypes ConnectionType;

       /// <summary>
       /// Возможности устройства при заданном формате StreamFormat
       /// </summary>
       public RTFR_Capabilities Capabilities;
}
```

Данную информацию нужно задать столько раз, сколько различных форматов потока поддерживает IP-устройство.

Аналогично **DevType_RegInfo**, информацию **RTFR_RegInfo** необходимо задавать на этапе загрузки сборки с плагином в методе инициализации класса, реализующего интерфейс **IPlugin**. Для этого достаточно вызвать метод **RegisterDevType** интерфейса хоста.

Пример заполнения данных классов приведен ниже:

```
public void Initialize(IPluginHost host)
{
   DevType_RegInfo KingNet_KS3002MJ_Device = new DevType_RegInfo();
   KingNet_KS3002MJ_Device.DeviceTypeGuid =
          new Guid("5C49C51D-B17B-40b0-9656-6DC71DD86D90");
   KingNet_KS3002MJ_Device.DevTypeModelName = "KS3002MJ";
   KingNet_KS3002MJ_Device.DevTypeBrandName = "KingNet";
   KingNet_KS3002MJ_Device.AvailableResolutions = GetConcreteResolutions();
   KingNet_KS3002MJ_Device.Capabilities = DevType_Capabilities.SupportsCameras;
   KingNet_KS3002MJ_Device.GetPtzController = null;
```

```
    KingNet_KS3002MJ_Device.GetRTFR = (conParam, subParams) =>
    {
         RealTimeFrameReceiver rtfr = new RealTimeFrameReceiver(conParam, subParams);
         return rtfr;
    };
    KingNet_KS3002MJ_Device.SetDeviceParameters = (conParams, subParams) =>
    {
         return true;
    };

    host.RegisterDevType(KingNet_KS3002MJ_Device);

    RTFR_RegInfo rtfrKingNet_KS3002MJ_Mjpeg = new RTFR_RegInfo();
    rtfrKingNet_KS3002MJ_Mjpeg.DeviceTypeGuid =
           new Guid("5C49C51D-B17B-40b0-9656-6DC71DD86D90");
    rtfrKingNet_KS3002MJ_Mjpeg.StreamFormat = VideoStreamFormats.MJPEG;
    rtfrKingNet_KS3002MJ_Mjpeg.ConnectionType = NetworkConnectionTypes.UDP;
    rtfrKingNet_KS3002MJ_Mjpeg.Capabilities.SupportedStreamTypes =
           ChannelStreamTypes.MainVideo;
    rtfrKingNet_KS3002MJ_Mjpeg.Capabilities.SupportedExtraParameters =
        new DeviceParameters[] { DeviceParameters.Null, DeviceParameters.Null };
    rtfrKingNet_KS3002MJ_Mjpeg.Capabilities.SupportedDeviceParameters =
        new DeviceParameters[] { DeviceParameters.Null, DeviceParameters.Null };

    host.RegisterRTFR(rtfrKingNet_KS3002MJ_Mjpeg);
}
```

4. Macroscop API с интерфейсами HTTP и RTSP

Macroscop API позволяет обращаться к серверу по HTTP или RTSP интерфейсам для получения видеопотоков реального времени и из архива, а также по HTTP интерфейсу для получения информации о системе и отправки команд системе на выполнение определенных действий. Ниже описаны различные типы запросов.

Пароль пользователя (параметр **password**) передается в виде MD5-хэша в верхнем регистре.

4.1 HTTP-интерфейс для получения видео

4.1.1 Получение видео реального времени и архива

Наиболее простым способом получения потоков данных из **Macroscop** является HTTP интерфейс. Получение видео реального времени по HTTP осуществляется следующим общего вида CGI-запросом на сервер:

<адрес и порт сервера macrosop>/video?channel=<название канала>
&login=<имя пользователя>&password=<хэш-строка MD5 пароля>[&sound=on]
[&streamtype=alternative]

или

{адрес и порт сервера macrosop}/video?channelid=<id канала>
&login={имя пользователя}&password=<хэш-строка MD5 пароля>

[&sound=on][&streamtype=alternative]

Если у пользователя нет пароля, то параметр password можно опустить или оставить его значение пустым. Опциональный параметр sound со значением on позволяет вместе с видео в том же соединении получать звук путем чередования кадров. Звуковые кадры всегда приходят в формате G.711U. Опционально можно запрашивать альтернативный поток, который, как правило, идет в меньшем разрешении и может быть использован для отображения.

В результате на указанный выше запрос сервер шлет «бесконечный» HTTP ответ, в котором идут видео (и аудио) кадры, разделенные заголовками. Типичный ответ сервера на запрос:

```
HTTP/1.1 200 OK
…
Content-Type: multipart/x-mixed-replace; boundary=myboundary

-- myboundary
Content-Type: image/jpeg
Content-Length: 63125

<тело JPEG кадра>

или

-- myboundary
Content-Type: audio, PCMU
Content-Length: 1000

<тело G711U кадра>
```

Если на канале установлен формат потока MJPEG, то в значении параметра **Content-Type** содержится строка **image/jpeg**. В случае MPEG-4 передается **Content-Type** равный **video, mpeg4, I-frame** для I-кадров и **video, mpeg4, P-frame** для P-кадров. В каждом опорном I-кадре имеется инициализирующая информация для декодера MPEG-4. Аналогично, в случае H.264, для I-кадров в поле **Content-Type** содержится значение **video, h264, I-frame** и **video, h264, P-frame**. Как и в случае MPEG4, перед каждым I-кадром встраивается инициализирующая информация для декодера H.264.

Пример запроса для получения видео реального времени:

http://127.0.0.1:8080/video?channel=Канал%201&login=root&password=

Во избежание коллизий, рекомендуется вместо имени канала передавать его идентификатор в параметре **channelid**:

http://127.0.0.1:8080/video?channelid=7f54efa4-e7d6-456e-ac1018215f2492d6&login=root&password=

Также для задания канала можно использовать его порядковый номер в конфигурации (начинается с нуля):

http://127.0.0.1:8080/video?channelnum=1&login=root&password=

Номер канала может измениться при изменении конфигурации (например, при перемещении каналов внутри объектов безопасности и изменении их порядка), поэтому из трех вышеперечисленных вариантов задания канала рекомендуется использовать его идентификатор (**channelid**).

Для того чтобы получить конфигурацию, содержащую идентификаторы каналов и их настройки, необходимо выполнить запрос:

{адрес и порт сервера macrosop}/configex?login=<имя пользователя>&password=<хэш-строка MD5 пароля>

Пример запроса:

http://127.0.0.1:8080/configex?login=root&password=

Подробнее запрос конфигурации описан в разделе Получение конфигурации системы (стр. 36).

Для доступа в архив по HTTP интерфейсу достаточно сформировать следующий CGI-запрос на сервер:

<адрес и порт сервера macrosop>/video?mode=archive&startTime=dd.mm.yyyy+hh:mm:ss[.fff][&speed=n]&channel=<название канала>[&channelid=id]&login=<имя пользователя>&password=<хэш-строка MD5 пароля>[&sound=on]

Параметр **startTime** является стартовой позицией, с которой начинается воспроизведение архива. Его значение представляется в виде комбинации даты и UTC-времени.

Опциональный параметр **speed** задает скорость воспроизведения архива. Диапазон принимаемых значений является непрерывным и изменяется от 0.1 до 20. Значение по умолчанию — 1.0.

Пример запроса:

http://127.0.0.1:8080/video?mode=archive&startTime=20.09.2011+11:49:12&speed=1&channel=Канал 1&login=root&password=

4.1.2 Получение перекодированного видео в формате MJPEG.

При обращении к интерфейсу **/video** сервер Macroscop будет возвращать видео в оригинальном (полученном от камеры) формате. Для некоторых приложений и непроизводительных устройств декодирование видео в формате H.264 или отображение MJPEG-видео в оригинальном разрешении может составить проблему. Для таких случаев в Macroscop есть CGI-обработчик **/mobile**. При запросе к нему с параметрами, аналогичными запросу **/video** (логин, пароль, имя/номер/идентификатор канала, воспроизведение звука, параметры архива), сервер Macroscop будет возвращать видео в формате MJPEG (в том числе и для потоков, которые транслируются с камер в формате H.264 и MGEG-4) в фиксированном разрешении, определенном в приложении **Macroscop Конфигуратор** (вкладка **Серверы**, блок настроек **Подключение мобильных устройств**).

Настройки перекодирования возвращаются в разделе **MobileServerInfo** XML-конфигурации, получаемой по запросу **/configex**).

Получение перекодированного видео может быть запрещено для группы, в которой состоит пользователь. В этом случае в XML-конфигурации (по запросу **/configex**) в разделе **UserGroup** параметр **CanGetTranscodedVideoFromMobileServer** будет иметь значение **false**.

По умолчанию возвращается самое низкое разрешение. Более высокое разрешение можно запросить, задав параметр **resolutionx** для ширины и **resolutiony** для высоты (целое положительное число пикселей). Например:

http://127.0.0.1:8080/mobile?channelnum=1&login=root&resolutionx=640&resolutiony=480

В этом случае будет возвращено наиболее подходящее из разрешений, определенных в настройках мобильных подключений сервера.

Каждому разрешению соответствует максимальная частота кадров (определяется в настройках). Клиент может запросить более низкую частоте кадров, задав в запросе параметр **fps** (целое положительное — число кадров в секунду). Если задать параметр **fps** больше максимального значения, или не задать вовсе, то видео сервера будет передаваться с максимально возможной частотой для запрошенного разрешения. Например:

http://127.0.0.1:8080/mobile?channelid=e9d42141-8f7f-4577-bb5e-4dd9f79331ab&login=root&resolutionx=640&resolutiony=480&fps=10

4.2 HTTP-интерфейс для получения данных

Запросы для получения данных в общем виде имеют следующий вид:

{адрес и порт сервера macrosop}/<команда>?login=<имя пользователя>&password=<хэш-строка MD5 пароля>&<Параметр 1>&<Параметр 2>...

По умолчанию, данные возвращаются в формате XML. Однако, для части запросов реализована возможность возвращать данные в формате JSON — для этого нужно указать параметр **responsetype=json**

. Если в описании запроса ниже явно не указана возможность возврата данных в JSON, подразумевается, что данные возвращаются только в XML.

4.2.1 Получение конфигурации системы

XML: http://127.0.0.1:8080/configex?login=root&password=

JSON: http://127.0.0.1:8080/configex?responsetype=json&login=root&password=

Параметры:

login – имя пользователя.

password – MD5-хэш от пароля.

Ответ на запрос в XML -формате:

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration  xmlns="http://www.macroscop.com"
                ServerVersion="2.0.15"
                XMLProtocolVersion="2"
                Timestamp="2015-05-20T12:19:21.8562773Z" Revision="71"
                SenderId="23424773-aa6e-4088-a0d3-97d50fe76c5f"
                Id="36b21916-1186-4ba0-8ad6-138f963140a6"
                xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
                xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <Servers>
        <ServerInfo  Id="23424773-aa6e-4088-a0d3-97d50fe76c5f"
                     Url="192.168.100.61:8080"
                     Name="Сервер 1" />
    </Servers>
```

```
<Channels>
    <ChannelInfo  Id="9bacefb4-4605-4813-940d-41c056248f8d"
                  Name="Канал 1"
                  ArchiveStreamType="Main"
                  ArchiveMode="MDandManual"
                  IsTransmitSoundOn="false"
                  IsPtzOn="false"
                  AllowedArchive="true"
                  AllowedRealtime="true"
                  IsSoundArchivingEnabled="false"
                  IsArchivingEnabled="true"
                  IsSoundOn="false"
                  IsDisabled="false"
                  AttachedToServer="23424773-aa6e-4088-a0d3-97d50fe76c5f"
                  DeviceInfo="LTV ICD(M,V)(x)-xxx"
                  Description="">
        <Streams>
            <StreamInfo  RotationMode="None"
                         StreamFormat="H264"
                         StreamType="Main"/>
            <StreamInfo  RotationMode="None"
                         StreamFormat="MJPEG"
                         StreamType="Alternative"/>
        </Streams>
    </ChannelInfo>
</Channels>
<RootSecurityObject Id="9c7d175a-9c84-455e-b104-57cc12cb9d47">
    <ChildSecurityObjects>
        <SecObjectInfo  Id="583f67a8-d173-4b59-8c49-5e774c99bcf9"
                        Name="Объект 1">
            <ChildSecurityObjects/>
            <ChildChannels>
                <ChannelId>9bacefb4-4605-4813-940d-41c056248f8d</ChannelId>
            </ChildChannels>
        </SecObjectInfo>
    </ChildSecurityObjects>
    <ChildChannels/>
</RootSecurityObject>
<UserGroup>
    <Id>464daa9e-755d-4491-a616-5fbda4423ac8</Id>
    <Name>Администраторы</Name>
    <CanConfigure>true</CanConfigure>
    <CanConfigureWorkplace>false</CanConfigureWorkplace>
    <CanShutdown>true</CanShutdown>
    <CanChangeChannelMode>true</CanChangeChannelMode>
    <CanManageRec>true</CanManageRec>
    <CanAccessExpertMode>true</CanAccessExpertMode>
    <CanPTZ>true</CanPTZ>
    <CanReceiveSound>true</CanReceiveSound>
    <CanTransmitSound>true</CanTransmitSound>
    <CanAccessNewCamera>true</CanAccessNewCamera>
    <CanGetTranscodedVideoFromMobileServer>
        true
    </CanGetTranscodedVideoFromMobileServer>
    <CanAccessEditingAnalystPluginsInClient>
        true
    </CanAccessEditingAnalystPluginsInClient>
    <CanAccessVideoViaWeb>true</CanAccessVideoViaWeb>
    <CanAccessVideoViaSmartTV>true</CanAccessVideoViaSmartTV>
    <CanExportVideoToAvi>true</CanExportVideoToAvi>
    <CanReceiveMainStream>true</CanReceiveMainStream>
    <AllowedArchiveDepth/>
```

```
        <IsAllForbidden>false</IsAllForbidden>
        <CanAccessUnifiedLog>true</CanAccessUnifiedLog>
        <CanAccessToAllUsersInUnifiedLog>true</CanAccessToAllUsersInUnifiedLog>
        <CanReceiveMobilePush>true</CanReceiveMobilePush>
    </UserGroup>
    <MobileServerInfo HighResolution="800 x 480"
                      MiddleResolution="240 x 180"
                      LowResolution="120 x 90"
                      FpsLimit="0"
                      UsePFrames="false"
                      Port="8089"
                      IsMobilePushEnabled="false"
                      IsProxyEnabled="true"
                      IsEnabled="true">
        <Resolutions>
            <ResolutionInfo  FpsLimit="15"
                             UsePFrames="true"
                             IsEnabled="true"
                             Type="High"
                             Height="480"
                             Width="800"/>
            <ResolutionInfo  FpsLimit="4"
                             UsePFrames="false"
                             IsEnabled="true"
                             Type="Middle"
                             Height="180"
                             Width="240"/>
            <ResolutionInfo  FpsLimit="4"
                             UsePFrames="false"
                             IsEnabled="false"
                             Type="Low"
                             Height="90"
                             Width="120"/>
        </Resolutions>
    </MobileServerInfo>
    <RtspServerInfo  IsEnabled="true"
                     IsMjpegEnabled="false"
                     TcpPort="554"/>
</Configuration>
```

Ответ на запрос в JSON -формате:

```
{
 "Id": "36b21916-1186-4ba0-8ad6-138f963140a6",
 "SenderId": "23424773-aa6e-4088-a0d3-97d50fe76c5f",
 "RevNum": 70,
 "Timestamp": "2015-05-20T12:13:37.6889429Z",
 "XmlProtocolVersion": 2,
 "ServerVersion": "2.0.15",
 "Servers": [
  {
   "Id": "23424773-aa6e-4088-a0d3-97d50fe76c5f",
   "Name": "Сервер 1",
   "Url": "192.168.100.61:8080",
   "ConnectionUrl": null
  }
 ],
 "Channels": [
  {
   "Id": "9bacefb4-4605-4813-940d-41c056248f8d",
   "Name": "Канал 1",
   "Description": "",
   "DeviceInfo": "LTV ICD(M,V)(x)-xxx",
```

```
    "AttachedToServer": "23424773-aa6e-4088-a0d3-97d50fe76c5f",
    "IsDisabled": false,
    "IsSoundOn": false,
    "IsArchivingEnabled": true,
    "IsSoundArchivingEnabled": false,
    "AllowedRealtime": true,
    "AllowedArchive": true,
    "IsPtzOn": false,
    "IsTransmitSoundOn": false,
    "ArchiveMode": "MDandManual",
    "Streams": [
     {
      "StreamType": 0,
      "StreamFormat": 3,
      "RotationMode": 0
     },
     {
      "StreamType": 1,
      "StreamFormat": 1,
      "RotationMode": 0
     }
    ],
    "ArchiveStreamType": "Main"
  }
 ],
 "RootSecObject": {
  "ChildSecObjects": [
   {
    "ChildSecObjects": [],
    "ChildChannels": [
     "9bacefb4-4605-4813-940d-41c056248f8d"
    ],
    "Id": "583f67a8-d173-4b59-8c49-5e774c99bcf9",
    "Name": "Объект 1"
   }
  ],
  "ChildChannels": [],
  "Id": "9c7d175a-9c84-455e-b104-57cc12cb9d47",
  "Name": null
 },
 "UserGroup": {
  "Id": "464daa9e-755d-4491-a616-5fbda4423ac8",
  "Name": "Администраторы",
  "CanConfigure": true,
  "CanConfigureWorkplace": false,
  "CanShutdown": true,
  "CanChangeChannelMode": true,
  "CanManageRec": true,
  "CanAccessExpertMode": true,
  "CanPTZ": true,
  "CanReceiveSound": true,
  "CanTransmitSound": true,
  "CanAccessNewCamera": true,
  "CanGetTranscodedVideoFromMobileServer": true,
  "CanAccessEditingAnalystPluginsInClient": true,
  "CanAccessVideoViaWeb": true,
  "CanAccessVideoViaSmartTV": true,
  "CanExportVideoToAvi": true,
  "CanReceiveMainStream": true,
```

```
  "AllowedArchiveDepth": "416.16:00:00",
  "IsAllForbidden": false,
  "CanAccessUnifiedLog": true,
  "CanAccessToAllUsersInUnifiedLog": true,
  "CanReceiveMobilePush": true
 },
 "MobileServerInfo": {
  "IsEnabled": true,
  "IsProxyEnabled": true,
  "IsMobilePushEnabled": false,
  "Port": 8089,
  "UsePFrames": false,
  "FpsLimit": 0,
  "LowResolution": "120 x 90",
  "MiddleResolution": "240 x 180",
  "HighResolution": "800 x 480",
  "Resolutions": [
   {
    "Width": 800,
    "Height": 480,
    "IsEnabled": true,
    "FpsLimit": 15,
    "UsePFrames": true,
    "Type": 2
   },
   {
    "Width": 240,
    "Height": 180,
    "IsEnabled": true,
    "FpsLimit": 4,
    "UsePFrames": false,
    "Type": 1
   },
   {
    "Width": 120,
    "Height": 90,
    "IsEnabled": false,
    "FpsLimit": 4,
    "UsePFrames": false,
    "Type": 0
   }
  ]
 },
 "RtspServerInfo": {
  "IsEnabled": true,
  "TcpPort": 554,
  "IsMjpegEnabled": false
 }
}
```


В ответе в элементе **Configuration** содержится:

- Версия протокола **XMLProtocolVersion**. На данный момент номер протокола равняется 2, его смена в будущем предполагает появления новых элементов и атрибутов в xml-ответе.
- Временная метка последнего применения конфигурации **Timestamp**.
- Уникальный идентификатор **Id** текущей конфигурации и номер ее ревизии **Revision**. Номер ревизии увеличивается на единицу после каждого изменения конфигурации.
- Идентификатор сервера **SenderId**, отправившего данный xml-ответ.

В элементе **Servers** содержится описание серверов, входящих в текущую конфигурацию. Каждый сервер описывается элементом **ServerInfo**, в который входят следующие атрибуты:

- Уникальный идентификатор сервера **Id**.
- Его название **Name** в рамках текущей конфигурации.
- **Url** сервера/

В элементе **Channels** содержится описание настроек каналов текущей конфигурации. Настройки каждого канала описываются элементом **ChannelInfo**, в котором содержатся следующие атрибуты и элементы:

- Уникальный идентификатор **Id** канала, который может быть использован в запросах на получение видео при задании параметра **channelid**.
- Идентификатор сервера **AttachedToServer**, к которому прикреплен канал.
- Информация о выбранном устройстве **DeviceInfo**.
- Параметр **IsArchivingEnabled**, показывающий включено ли на канале архивирование видеоданных.
- Параметр **IsSoundArchivingEnabled**, показывающий включено ли архивирование звуковых данных.
- Параметр **ArchiveMode**, показывающий режим записи в архив на канале (если она включена). Возможные значения: **AlwaysOn** — всегда включена; **OnlyManual** — только ручное управление, **BySchedule** — запись по расписанию; **MDandManual** — автоматическое включение по детектору движения и ручное.
- Параметр **IsSoundOn**, показывающий включено ли получение звука на данном канале. Если данный параметр равен false, то параметр **IsSoundArchivingEnabled** может быть проигнорирован.
- Параметр **AllowedRealtime**, показывающий разрешено ли текущему пользователю просматривать видео в реальном времени на данном канале.
- Параметр **AllowedArchive**, показывающий разрешено ли текущему пользователю просматривать видео из архива на данном канале.
- Параметр **IsDisabled**, показывающий отключен ли канал в текущей конфигурации.
- Параметр **Name**, содержащий имя канала в конфигурации.
- Элемент **Streams**, содержащий настройки основного и альтернативного потоков видеоданных. В описании потока **StreamInfo** содержится один из трех возможных форматов потоков (H264, MPEG4, MJPEG) в поле **StreamFormat**; а также тип потока, принимающий значение **main** (основной) или **alternative** (альтернативный). Если альтернативный поток не включен на данном канале, то его описание будет отсутствовать.

В элементе **RootSecurityObject** содержится информация о структуре дерева объектов безопасности и принадлежности к ним каналов.

В элементе **UserGroup** содержится информация о правах группы, к которой принадлежит пользователь, который запрашивает конфигурацию.

В элементе **MobileServerInfo** содержится информация о параметрах перекодирования видео мобильным сервером.

4.2.2 Получение профилей (предустановленные сетки)

Примеры запроса:

XML: http://127.0.0.1:8080/command?type=getprofiles&login=root&password=

JSON:
http://127.0.0.1:8080/command?type=getprofiles&login=root&password=&responsetype=json

Пример ответа в JSON-формате в JSON:

```
[
    {
        "Id": "13851f3d-c7d3-4ec6-b0ff-2d66873bf118",
        "Name": "Новый профиль 1"
    }
]
```

4.2.3 Получение списка доступных сеток на клиенте Macroscop

XML:
http://127.0.0.1:8080/command?type=getgrids&clientip=127.0.0.1&monitor=0&login=root&password=

JSON:
http://127.0.0.1:8080/command?type=getgrids&clientip=127.0.0.1&monitor=0&login=root&password=&responsetype=json

В результате выполнения запроса возвращается список доступных сеток на конкретном клиенте. Пример:

```
[
    "1",
    "4",
    "6_1",
    "7",
    "8_1",
    "9",
    "10",
    "13",
    "16",
    "25"
]
```

Первая цифра (до знака "_") означает число ячеек в сетке, вторая (после "_") — номер конфигурации. Номер конфигурации введен только для сеток, обладающих одинаковым числом ячеек, но отличающихся размером и расположением ячеек.

4.2.4 Получение информации о текущей сетке в клиенте Macroscop

XML: http://127.0.0.1:8080/command?type=getcurrentgrid&login=root&clientip=127.0.0.1&monitor=0

JSON: http://127.0.0.1:8080/command?type=getcurrentgrid&login=root&clientip=127.0.0.1&monitor=0&responsetype=json

Параметры:

clientip — IP-адрес или URI клиента.

monitor — номер монитора на клиенте (начинается с 0).

В результате выполнения запроса возвращается тип текущей сетки и ее содержимое. Пример:

```
{
    "GridType": "4",
    "Cells":
    [
        {
            "Index": 0,
            "IsEmpty": false,
            "ChannelId": "483cd419-c03c-47b1-a3bb-ef5bce82d588",
            "Viewer": "Realtime"
        },
```

```
        {
            "Index": 1,
            "IsEmpty": false,
            "ChannelId": "fe5e37d5-6da8-403c-ace8-10c4ba4c4b64",
            "Viewer": "Realtime"
        },
        {
            "Index": 2,
            "IsEmpty": false,
            "ChannelId": "edc2f629-4565-49a1-8c70-663e16ab0104",
            "Viewer": "Realtime"
        },
        {
            "Index": 3,
            "IsEmpty": false,
            "ChannelId": "1b6204af-f3ae-49ab-935f-878cbb3a9139",
            "Viewer": "Realtime"
        }
    ]
}
```


Где:

index — порядковый номер ячейки (отсчет с нуля);

isempty — пуста ли ячейка;

channeled – идентификатор канала;

viewer — тип содержимого в ячейке, принимает значения:

- **none** — ячейка пуста;
- **realtime** — реальное время;
- **archive** — архив;
- **other** — другое, на данный момент может быть только план помещения.

Тип сетки совпадает с описанным в разделе Получение списка доступных сеток на клиенте Macroscop (стр. 42).

4.2.5 Получение времени компьютера, на котором работает сервер Macroscop

XML: http://127.0.0.1:8080/command?type=gettime&login=root&password=

JSON: http://127.0.0.1:8080/command?type=gettime&login=root&password=&responsetype=json

В результате выполнения запроса приходит время в UTC. Пример:


```
"22.09.2014 3:33:06"
```


4.2.6 Получение информации и о наличии архива в указанный момент времени

XML: http://127.0.0.1:8080/command?type=findarchive&login=root&password=&channelid=d96e8b67-3f13-44fd-b628-356d1f88a50c&searchTime=23.09.2014+06:10:00

JSON: http://127.0.0.1:8080/command?type=findarchive&login=root&password=&channelid=d96e8b67-3f13-44fd-b628-356d1f88a50c&searchTime=23.09.2014+06:10:00&responsetype=json

Время в параметре **searchTime** должно передаваться в UTC.

В результате выполнения запроса возвращается флаг наличия архива за указанное время и временная метка ближайшего видеокадра. Пример:


```
{
    "HasArchive": false,
    "NearestFrameTimestamp": null
}
```

4.2.7 Получение информации о состоянии каналов.

XML: http://127.0.0.1:8080/command?type=getchannelsstates&login=root&password=

JSON: http://127.0.0.1:8080/command?type=getchannelsstates&login=root&password=&responsetype=json

Пример ответа в JSON-формате:


```
[
    {
        "Id": "596ea82f-cf03-4e1f-9658-5aafbb1cb143",
        "IsRecordingOn": false,
        "StreamsStates":
         [
            {
                "Type": "MainVideo",
                "State": "Active"
            },
            {
                "Type": "AlternativeVideo",
                "State": "Stopped"
            },
            {
                "Type": "MainSound",
                "State": "Active"
            },
            {
                "Type": "AlternativeSound",
                "State": "Stopped"
            },
            {
                "Type": "OutputSound",
                "State": "Stopped"
            },
            {
                "Type": "MotionDetection",
                "State": "Stopped"
            },
            {
                "Type": "IO",
                "State": "Stopped"
            },
            {
                "Type": "ArchiveVideo",
                "State": "Stopped"
            },
            {
                "Type": "ArchiveSound",
                "State": "Stopped"
            }
        ]
    }
]
```


Где:

Id — идентификатор канала;

IsRecordingOn — включена ли запись на канале;

StreamStates — состояние потоков получения данных. У каждого потока есть тип **Type** и состояние **State**.

Type принимает значения:

- **MainVideo** — видео, основной поток, высокое разрешение;
- **AlternativeVideo** — видео, второй поток, среднее разрешение;
- **MainSound** — прием звука, основной поток;
- **AlternativeSound** — прием звука, альтернативный поток;
- **OutputSound** — передача звука;
- **MotionDetection** — встроенный детектор движения;
- **IO** — цифровые входы/выходы;
- **ArchiveVideo** — архивное видео;
- **ArchiveSound** — архивный звук.

State принимает значения:

- **Stopped** — поток остановлен, т.к. он не требуется системе;
- **Active** — поток находится в состоянии получения кадров и событий;
- **NoConnection** — в потоке произошел обрыв соединения с устройством.

4.2.8 Получение PTZ-возможностей устройства

XML: http://127.0.0.1:8080/ptz?command=getcapabilities&channelid=20d9884f-ae8c-45d3-ac5a-505ec258f01b&login=root&password=

JSON: http://127.0.0.1:8080/ptz?command=getcapabilities&channelid=20d9884f-ae8c-45d3-ac5a-505ec258f01b&login=root&password=&responsetype=json

Пример ответа в JSON-формате:

```
{
    "PtzCapabilities":
    {
        "HomePositionSupports": true,
        "MoveToSupports": false,
        "AreaZoomSupports": false,
        "ZoomSupports": true,
        "ContiniousZoomSupports": true,
        "AutoFocusSupports": true,
        "ManualFocusSupports": true,
        "ContiniousFocusSupports": true,
        "SupportedStepMoveDirections": 0,
        "SupportedContiniousMoveDirections": 255
    },
    "Resolution":
    {
        "Width": 1920,
        "Height": 1080
    }
}
```

Где

HomePositionSupports — поддерживается ли домашнее положение;

MoveToSupports — поддерживается ли центрирование (камера поворачивается к заданной точке);

AreaZoomSupports — поддерживается ли функция приближения выделенной области;

ZoomSupports — поддерживаются ли «шаговый» зум (при отправке одной команды однократное изменение зума);

ContiniousZoomSupports — поддерживается ли «непрерывный» зум (при отправке команды зум идет до тех пор, пока не придет команда на остановку);

AutoFocusSupports — поддерживается ли функция автофокусировки;

ManualFocusSupports — возможность ручного управления фокусом;

ContiniousFocusSupports — поддерживается ли «следящий» фокус;

SupportedStepMoveDirections — маска, описывающая доступные направления для «шагового перемещения»;

SupportedContiniousMoveDirections — маска, описывающая доступные направления для «непрерывного» движения;

Resolution — текущее разрешение кадра на основном потоке (необходимо для команд **moveto** и **areazoom**, см. ниже).

4.2.9 Получение пресетов с PTZ-устройства

XML: http://127.0.0.1:8080/ptz?command=getpresets&channelid=20d9884f-ae8c-45d3-ac5a-505ec258f01b&login=root&password=

JSON: http://127.0.0.1:8080/ptz?command=getpresets&channelid=20d9884f-ae8c-45d3-ac5a-505ec258f01b&login=root&password=&responsetype=json

Пример ответа в JSON-формате:

```
[
    "Предуст. 1",
    "Предуст. 2",
    "Предуст. 3",
    "Предуст. 4",
    "Предуст. 8"
]
```

4.2.10 Получение архива распознанных автомобильных номеров

Данные из архива распознанных автономеров всегда возвращаются в JSON-формате. Пример запроса:

http://127.0.0.1:8080/autovprs_export?login=root&password=&channelid=2ae58228-bb85-490b-8053-a3bde125462c&startTime=2015-05-20-15-36-00-000&finishTime=2015-05-20-15-45-59-999

С помощью параметров **startTime** и **finishTime** задается время нчала и конца временного интервала, за который требуется получить архив распознанных автономеров. Эти параметры необходимо указывать в **локальном времени** сервера, на который отправляется запрос.

Пример ответа в JSON-формате:

```
[
    {
        "TimeUtc" : "21.05.2015 05:53:17.537",
        "Numberplate" : "C896AX59",
        "LastName" : "",
        "FirstName" : "",
        "PatronymicName" : "",
        "Group" : ""
    } ,
    {
        "TimeUtc" : "21.05.2015 05:53:34.467",
        "Numberplate" : "X497KO59",
        "LastName" : "",
        "FirstName" : "",
        "PatronymicName" : "",
        "Group" : ""
    }
]
```

Где:

Timestamp — UTC-время, когда был распознан номер (для преобразования UTC в локальное время следует учитывать часовой пояс; подробнее о различиях между UTC и местным временем см. в Википедии);

Numberplate — распознанный автомобильный номер.

4.2.11 Получение списка всех зарегистрированных в системе событий

Реализовано в Macroscop версии 2.1 и более поздних.

http://127.0.0.1:8080/command?type=getallregisteredevents&login=root

Параметры:
login – имя пользователя;
password – MD5-хэш от пароля.

Формат ответа в XML-формате:

```
Timestamp: 27.07.2015 8:32:45
Error code: 0
Message: Success
Body-length: 3148
<?xml version="1.0" encoding="utf-8"?>
<ArrayOfEventInfo xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <EventInfo>
    <Id>00000000-0000-0000-0000-000000000033</Id>
    <GuiName>Движение</GuiName>
  </EventInfo>
</ArrayOfEventInfo>
```

4.2.12 Получение списка специальных архивных событий

Реализовано в Macroscop версии 2.1 и более поздних.

Ограничение на количество событий – 1000. Для получение последующих событий необходимо начинать поиск с времени последнего полученного события.

http://127.0.0.1:8080/specialarchiveevents?login=root&password=
&startTime=27.07.2015+07:39:05&endTime=27.07.2015+09:39:05
&eventid=9f5d8d14-e55d-4585-bb7f-c8c4bca1eeff

Параметры:
login — имя пользователя;
password — MD5-хэш пароля;
starttime — Начальное время архива (dd:mm:yyyy+HH:MM:SS);
endtime — Конечное время архива (dd:mm:yyyy+HH:MM:SS);
channelid — Guid канала для фильтрации (опционально);
filter — Guid события для фильтрации (опционально).

В данном запросе можно использовать только следующие значения фильтра:

```
"Id": "b19168ea-f3bb-473d-96b8-e82d278b3e7e", "GuiName": "Сигнал на вход камеры"
"Id": "5b5c2d65-c31a-4c2b-8a49-fa7d2a1f1e61", "GuiName": "Событие из внешней системы"
"Id": "9f5d8d14-e55d-4585-bb7f-c8c4bca1eeff", "GuiName": "Количество людей в очереди"
"Id": "427f1cc3-2c2f-4f50-8865-56ae99c3610d", "GuiName": "Обнаружено лицо (Модуль распознавания лиц)"
"Id": "c9d6d086-c965-4cf8-aef6-85b3894e3a4a", "GuiName": "Обнаружен автономер"
"Id": "99fff35a-0902-4ab3-a6cf-74cea73dffd6", "GuiName": "Требование открыть шлагбаум"
"Id": "90e32fea-f091-467c-ac2e-9767a2f257e8", "GuiName": "Требование закрыть шлагбаум"
"Id": "463fb0af-4273-40a1-bd42-90ce564d847a", "GuiName": "Неактивная зона"
"Id": "d99a411f-d0a6-42c4-b320-3c2dd0aa0829", "GuiName": "Обнаружен оставленный предмет"
```

Формат ответа в JSON-формате:

```
{
    "EventId" : "00000000-0000-0000-0000-000000000033",
    "Timestamp" : "27.07.2015 7:37:01",
    "EventDescription" : "Движение",
    "IsAlarmEvent" : "False",
    "ChannelId" : "ed93a7d5-4e90-4ab0-bcb7-d0a4dad4782e",
    "ChannelName" : "Канал 3",
    "Zoneid" : "5d295125-cac6-448b-8cc9-efee4fe21cc1"
}
{
    "EventId" : "00000000-0000-0000-0000-000000000033",
    "Timestamp" : "27.07.2015 7:37:03",
    "EventDescription" : "Движение",
    "IsAlarmEvent" : "False",
    "ChannelId" : "ed93a7d5-4e90-4ab0-bcb7-d0a4dad4782e",
    "ChannelName" : "Канал 3",
    "Zoneid" : "5d295125-cac6-448b-8cc9-efee4fe21cc1"
}
```

4.3 HTTP-интерфейс для получения событий в реальном времени

События можно получать в JSON-формате в «бесконечном» HTTP-соединении. Для этого необходимо выполнить следующий запрос:

http://127.0.0.1:8080/event?login=root&password=&responsetype=json

Где:
login — имя пользователя;
password — MD5-хэш пароля;
responsetype — ответ будет прислан в JSON;
channelid — GUID канала для фильтрации (опционально);
filter — GUID события для фильтрации (опционально).

Если необходимо получать события только для одного канала и/или только одного известного типа, то в строку запроса добавляется соответствующий параметр (**filter** с указанием идентификатора события и/или **channelid** с указанием идентификатора канала):

http://127.0.0.1:8080/event?login=root&password=
&filter=00000000-0000-0000-0000-000000000033&responsetype=json

Для отладки может быть добавлен параметр **mode=demo**, который укажет системе генерировать виртуальные события с типом, задаваемого идентификатором события в параметре **filter**:

http://127.0.0.1:8080/event?login=root
&filter=00000000-0000-0000-0000-000000000033&responsetype=json&mode=demo

Пример ответа в JSON-формате:

```
{
    "EventId" : "00000000-0000-0000-0000-000000000033",
    "Timestamp" : "27.07.2015 7:37:01",
    "EventDescription" : "Движение",
    "IsAlarmEvent" : "False",
    "ChannelId" : "ed93a7d5-4e90-4ab0-bcb7-d0a4dad4782e",
    "ChannelName" : "Канал 3",
    "Zoneid" : "5d295125-cac6-448b-8cc9-efee4fe21cc1"
}
```

```
{
    "EventId" : "00000000-0000-0000-0000-000000000033",
    "Timestamp" : "27.07.2015 7:37:03",
    "EventDescription" : "Движение",
    "IsAlarmEvent" : "False",
    "ChannelId" : "ed93a7d5-4e90-4ab0-bcb7-d0a4dad4782e",
    "ChannelName" : "Канал 3",
    "Zoneid" : "5d295125-cac6-448b-8cc9-efee4fe21cc1"
}
```

Идентификаторы событий:

- **00000000-0000-0000-0000-000000000033** — детекция движения;
- **8ee14b08-fc12-4b0f-a11b-3c859d4f4848** — установление связи с устройством;
- **e37ac864-824f-4848-bc25-7dc87fb145c7** — кратковременный обрыв связи с устройством;
- **ec94ad4b-6df0-4cd7-b2f7-9b4eff7e6413** — длительное отсутствия соединения с устройством;
- **0261ebee-fe03-4a13-8327-1b62d3b6f9e5** — изменение сигнала на тревожном входе устройства;
- **d117a331-7aca-4202-ac47-02d33402f7b6** — обнаружение лица;
- **e019878b-b965-4e21-950b-45ed99f7369e** — обнаружение или распознавание лица во внешнем модуле распознавания;
- **c9d6d086-c965-4cf8-aef6-85b3894e3a4a** — обнаружение или распознавание автомобильного номера во внешнем модуле распознавания;
- **8b5110f3-57d9-48f5-b3a2-ec698c9cff8d** — пользовательская тревога.

4.4 HTTP-интерфейс для отправки команд на сервер Macroscop

Для отправки команд на сервер Macroscop используются CGI-запросы, описанные ниже.

4.4.1 Включение/выключение записи на канале

Для включения записи с обязательным указанием времени записи в минутах нужно выполнить запрос:

http://127.0.0.1:8080/command?type=recording&mode=start&channelid=34512d50-c87e-4c75-a5a5-9b3a5aaa7d13&Interval=20&login=root&password=

Где:

channelid — GUID канала;

interval — время записи в минутах;

mode — режим, принимает значения **start** и **stop**.

Для выключения записи необходимо послать запрос:

http://127.0.0.1:8080/command?type=recording&mode=stop&channelid=34512d50-c87e-4c75-a5a5-9b3a5aaa7d13&login=root&password=

4.4.2 Синхронизация времени с другим компьютером в сети

Данный запрос установит время на сервере Macroscop согласно времени в параметре **time**.

http://127.0.0.1:8080/command?type=settime&time=”02.02.2014+08:11:00”&login=root&password=

4.4.3 Получение профилей (предустановленные сетки)

http://127.0.0.1:8080/command?type=getprofiles&login=root&password=&responsetype=json

Пример ответа в JSON-формате:

```
[
 {
  "Id": "13851f3d-c7d3-4ec6-b0ff-2d66873bf118",
  "Name": "Новый профиль 1"
 }
]
```

4.4.4 Установка профиля на клиенте

http://127.0.0.1:8080/command?type=setprofile&clientip=127.0.0.1&monitor=0&profileid=13851f3d-c7d3-4ec6-b0ff-2d66873bf118&login=root&password=

Где:

clientip — IP-адрес компьютера с установленным Macroscop клиентом;

monitor — номер монитора (отсчет от нуля);

pofileid — идентификатор профиля, который может быть получен из ответа на запрос получения профилей.

4.4.5 Смена сетки на клиенте

http://127.0.0.1:8080/command?type=setgrid&clientip=127.0.0.1&monitor=0&Cells=25&login=root&password=

Устанавливает требуемую сетку, передаваемую в параметре **cells**. Сетка должна быть разрешена для использования на указанном клиенте.

4.4.6 Очистка сетки

http://127.0.0.1:8080/command?type=cleargrid&clientip=127.0.0.1&Monitor=0&login=root&password=

Закрывает все каналы, открытые в ячейках текущей сетки

4.4.7 Установка канала в ячейку сетки

http://127.0.0.1:8080/command?type=setcell&clientip=127.0.0.1&monitor=0&mode=archive&cell=0&channelid=34512d50-c87e-4c75-a5a5-9b3a5aaa7d13&login=root&password=

Где:

mode – принимает значения **realtime** (для просмотра реального времени) или **archive** (для доступа в архив);

cell – номер ячейки, в которую необходимо поместить канал (отсчет от 0).

В зависимости от параметра **mode** открывает либо канал реального времени, либо архивный канал в указанной ячейке. Опционально возможно указать время **startTime**, начиная с которого должно начинаться воспроизведение архива и скорость проигрывания с помощью параметра **speed** (доступные скорости для воспроизведения: 0,1; 0,2; 0,5; 1; 2; 5; 10; 20; 60; 120).

4.4.8 Удаление канала из ячейки сетки

http://127.0.0.1:8080/command?type=clearcell&clientip=127.0.0.1&Monitor=0&cell=0&login=root&password=

Где **cell** – номер ячейки в сетке

4.4.9 Команда PTZ для «непрерывного» движения

http://127.0.0.1:8080/ptz?command=startmove&panspeed=2&tiltspeed=2
&channelid=20d9884f-ae8c-45d3-ac5a-505ec258f01b&login=root&password=

Где:

panspeed — скорость по горизонтали от -100 до 100;
tiltspeed — скорость по вертикали от -100 до 100.

4.4.10 Команда PTZ для «непрерывного» изменения фокуса

http://127.0.0.1:8080/ptz?command=startchangefocus&speed=1
&channelid=20d9884f-ae8c-45d3-ac5a-505ec258f01b&login=root&password=

Где **speed** — скорость фокусировки от -100 до 100.

4.4.11 Команда PTZ для «непрерывного» зума

http://127.0.0.1:8080/ptz?command=startzoom&speed=1
&channelid=20d9884f-ae8c-45d3-ac5a-505ec258f01b&login=root&password=

Где **speed** — скорость -100 до 100.

4.4.12 Команда PTZ остановки для «непрерывных» команд

http://127.0.0.1:8080/ptz?command=stop
&channelid=20d9884f-ae8c-45d3-ac5a-505ec258f01b&login=root&password=

4.4.13 Команда PTZ установки пресета

http://127.0.0.1:8080/ptz?command=gotopreset&index=1
&channelid=20d9884f-ae8c-45d3-ac5a-505ec258f01b&login=root&password=

Где **index** — порядковый номер пресета, отсчет с единицы.

4.4.14 Команда PTZ автофокусировки

http://127.0.0.1:8080/ptz?command=setautofocus
&channelid=20d9884f-ae8c-45d3-ac5a-505ec258f01b&login=root&password=

4.4.15 Команда PTZ для выполнения центрирования

http://127.0.0.1:8080/ptz?command=moveto&x=10&y=50&width=640&height=480
&channelid=20d9884f-ae8c-45d3-ac5a-505ec258f01b&login=root&password=

Где **x** и **y** — координаты на кадре размером **width** и **height**.

4.4.16 Команда PTZ для «шагового» движения

http://127.0.0.1:8080/ptz?command=move&tiltstep=1&panstep=1
&channelid=20d9884f-ae8c-45d3-ac5a-505ec258f01b&login=root&password=

Где:

tiltstep — шаг по вертикали от -100 до 100;
panstep — шаг по горизонтали от -100 до 100.

4.4.17 Команда PTZ для «шагового» зума

http://127.0.0.1:8080/ptz?command=zoom&zoomstep=10
&channelid=20d9884f-ae8c-45d3-ac5a-505ec258f01b&login=root&password=

Где **zoomstep** — шаг от -100 до 100.

4.4.18 Команда PTZ приближения выделенной области (AreaZoom)

http://127.0.0.1:8080/ptz?command=showrect&x=10&y=50&width=640&height=480&frameWidth=1920&frameHeight=1080&channelid=20d9884f-ae8c-45d3-ac5a-505ec258f01b&login=root&password=

Где **x**, **y**, **width**, **height** – задают в кадре прямоугольник, который будет увеличен;

frameWidth — ширина кадра;

frameHeight — высота кадра.

4.4.19 Постановка канала на охрану

Реализовано в Macroscop версии 2.1 и более поздних.

http://127.0.0.1:8080/command?type=setguardian&clientip=127.0.0.1&monitor=0&mode=realtime&login=root&password=&channelid=f67b3044-2fe2-451f-885a-7d570f8d2e01&isguardianmodeenabled=false

Параметры:

clientip — IP-адрес клиентского компьютера, для которого выполняется команда;

monitor — номер монитора клиентского компьютера;

isguardianmodenabled — **true** – включение / **false** – выключение охраны.

4.4.20 Отправка звука на камеру

Реализовано в Macroscop версии 2.1 и более поздних.

Для отправки звука на камеру используется POST-запрос. В заголовке указываются параметры запроса, в теле запроса передаются аудиоданные.

Заголовок:

http://127.0.0.1:8080/sendsound?login=root&password=&channelid=66abc0c4-d4b7-4d71-8ed1-e7beadf0dc46&clientid=66abc0c4-d4b7-4d71-8ed1-e7beadf0dc46

Параметры:

clientid — GUID сеанса передачи.

Для тела запроса **ContentType = "multipart/form-data;"**

Описание:

Запрос предназначен для использования в сторонних приложениях — эти приложения называются **клиентами**.

В рамках одного запроса передается одна порция аудиоданных; т.е. запрос не является постоянным подключением и сервер передает порцию звука на камеру только после того, как завершит прием соответствующего запроса от клиента.

Для формирования порции аудиоданных (кодирования звука) следует использовать сторонние библиотеки (например, NAudio: https://github.com/naudio/NAudio, параметры кодирования — Samplesrate = 8000; Bitspersample = 16; Количетсво каналов = 1).

Сеансом передачи считается серия запросов с определенного клиентского компьютера на определённую камеру. Для каждого сеанса передачи должен использоваться уникальный **clientID**. Для снижения серверных издержек рекомендуется для серии запросов в рамках одного сеанса использовать один и тот же **clientID**. Клиент самостоятельно формирует **clientid**.

4.4.21 Генерация события из внешней системы

Реализовано в Macroscop версии 2.1 и более поздних.

Данный запрос генерирует системное событие Событие из внешней системы. Пример:

http://127.0.0.1:8080/command?type=generateexternalevent&login=root&password=&channelid=af820905-3641-4448-a73a-eb5e91da73db&systemname=TestSystem&information=record

Параметры:

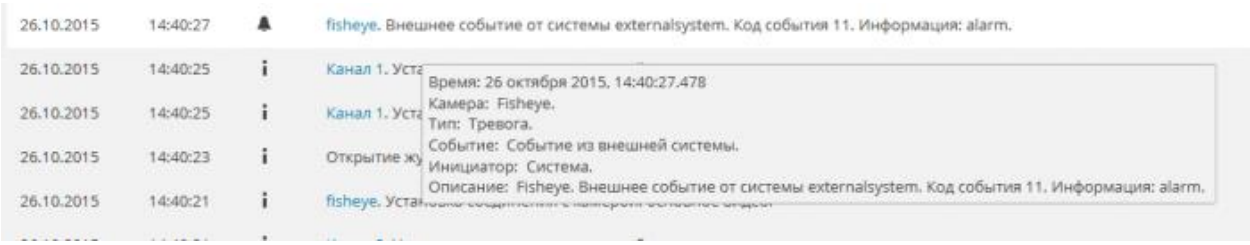
systemname — название внешней системы;

information — строка с информацией о событии;

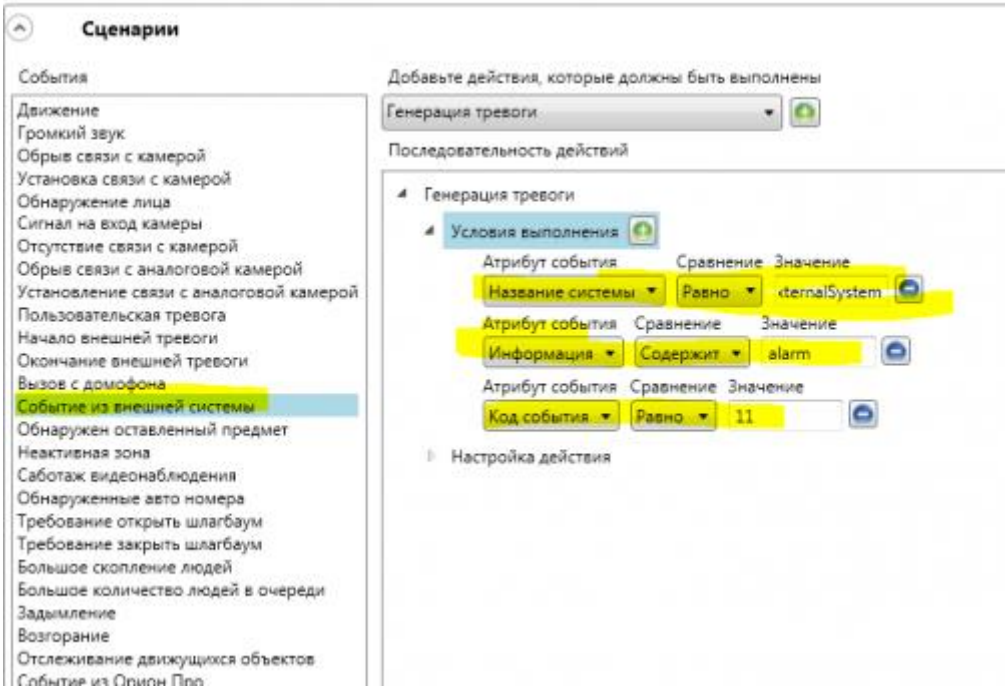
eventcode — код события (опционально).

В запросе обязательно должно быть указано либо название системы (**systemname**), либо информация о событии (**information**).

В **Журнале событий** приложения **Macroscop Клиент** данное событие будет выглядеть следующим образом:



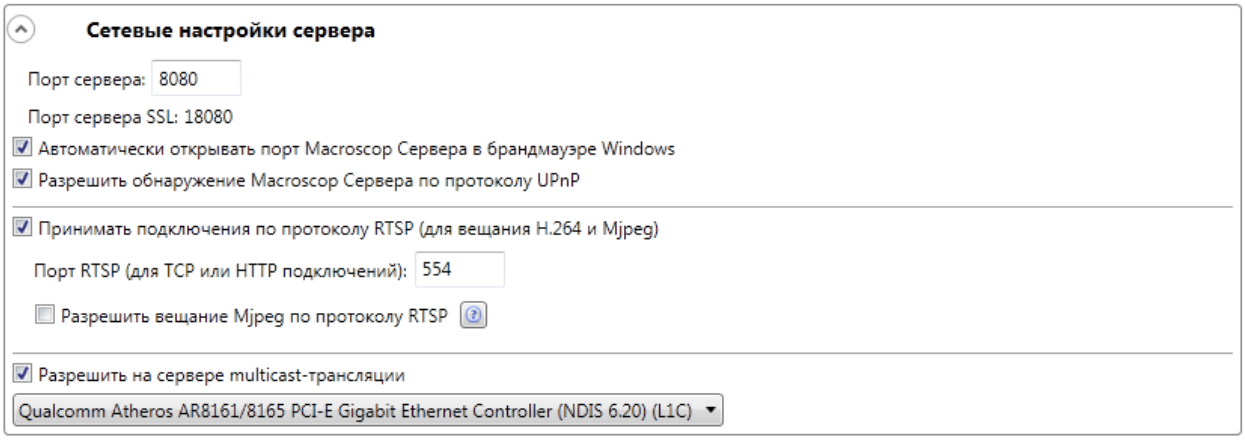
Также на эти события можно назначать действия в сценариях (посредством приложения **Macroscop Конфигуратор**):



4.5 RTSP-интерфейс для получения видео и звука

RTSP-интерфейс используется для получения видео и звука клиентами, работающими по протоколу RTSP. Данный интерфейс поддерживает кодек H.264 и, опционально, MJPEG (MJPEG по умолчанию выключен).

Перед началом использования следует удостовериться, что RTSP-интерфейс включен. Для этого нужно запустить **Macroscop Конфигуратор** и на странице серверных настроек в разделе **Сетевые настройки сервера** убедиться, что установлен флажок **Принимать подключения по протоколу RTSP (для вещания H.264 и MJPEG)**. В том же блоке настроек указан порт для RTSP подключений.



Для подключения по протоколу RTSP можно использовать TCP-подключения (RTSP over TCP) или HTTP-подключения (RTSP over HTTP). UDP-подключения (RTSP over UDP) не поддерживаются.

По умолчанию вещание формата MJPEG по протоколу RTSP отключено, поскольку протокол RTSP поддерживает только MJPEG-кадры, закодированные в базовом (Baseline) режиме кодирования. Таким образом, для передачи видеопотоков, закодированных в других режимах MJPEG, потребуется перекодирование; что, в свою очередь повысит нагрузку на сервер. Кроме того, при перекодировании MJPEG может быть понижена частота кадров (по сравнению с частотой кадров, передаваемой непосредственно камерой).

Подключение к серверу осуществляется RTSP-клиентом, например VLC, с помощью строки подключения следующего вида:

rtsp://<ip адрес сервера macroscop и порт rtsp>/rtsp?channelid=<id канала>
&login=<имя пользователя>&password=<хэш-строка MD5 пароля>[&sound=on]
[&streamtype=alternative]

Обязательный параметр **channelid** задает идентификатор канала, по которому необходимо получать видео. Также для задания канала можно использовать порядковый номер канала (начинается с нуля) в конфигурации — для этого вместо параметра **channelid** следует использовать параметр **channelnum** (см. Получение видео реального времени и архива, стр. 34).

Обязательный параметр **login** задает имя пользователя. У пользователя должны быть права на запрашиваемый канал.

Параметр **password** является обязательным, если у пользователя установлен пароль. Если у пользователя нет пароля (пустой пароль), то параметр **password** является опциональным и может быть опущен, либо оставлен с пустым значением.

Опциональный параметр **sound** со значением **on** позволяет вместе с видео получать звук. Звуковые кадры всегда приходят в формате G.711U.

Опциональный параметр **streamtype** со значением **alternative** позволяет запрашивать альтернативный (2-й) поток видео, который, как правило, имеет меньшее разрешение.

Пример запроса:

rtsp://127.0.0.1:554/rtsp?channelid=D3039B22-3350-47C6-85FE-40F29B1C7FBD&login=root

Для доступа в архив требуется задать параметры, аналогичные параметрам при подключении по HTTP (см. Получение видео реального времени и архива, стр. 34).

rtsp://<ip адрес сервера macroscop и порт rtsp>/rtsp?channelid=<id канала>
&login=<имя пользователя>&password=<хэш-строка MD5 пароля>&mode=archive
&starttime= <dd.mm.yyyy+ hh:mm:ss[.fff]>[&sound=on][&speed=<n>][&isforward=false]

Параметры **channelid**, **login**, **password** и **sound** аналогичны параметрам запроса для получения видео реального времени по протоколу RTSP.

Обязательный параметр **mode** со значением **archive** указывает на доступ в архив.

Обязательный параметр **starttime** указывает время записи, с которого начинается воспроизведение архива. Это значение задается в виде комбинации даты и UTC-времени.

Опциональный параметр **speed** задает скорость воспроизведения архива. Диапазон принимаемых значений является непрерывным и изменяется от **0.1** до **20**. Значение по умолчанию — **1.0**.

Опциональный параметр **isforward** со значением **false** указываетт, что воспроизводить архив следует назад.

Пример запроса:

rtsp://127.0.0.1:554/rtsp?channelid=D3039B22-3350-47C6-85FE-40F29B1C7FBD
&login=root&mode=archive&starttime=21.04.2015+ 12:05:01.125&sound=on&speed=1

5. Macroscop API с интерфейсом XML

XML интерфейс позволяет посылать на сервер **Macroscop** запросы в формате XML и получать в ответ данные в том же формате. Структура запроса должна быть следующей:

```
<?xml version="1.0" encoding="utf-8" ?>
<query>
  <server_login>root</server_login>
  <server_pass_hash></server_pass_hash>
  <query_name>get_people_counters</query_name>
  <query_params>
    …
  </query_params>
</query>
```

Ниже описано назначение параметров:

Параметр	Описание
server_login	Имя пользователя, под которым будет выполняться команда
server_pass_hash	MD5 хэш пароля пользователя
query_name	Строковое наименование типа запроса
query_params	Внутри данного тэга будут помещаться параметры, специфичные для типа запроса, указанного в параметре **query_name**

В ответ сервер возвращает ответ вида:

```
<?xml version="1.0" encoding="utf-8" ?>
<result>
  <query_name></query_name>
  <query_result>Ok</query_result>
  <query_msg>Запрос выполнен успешно.</query_msg>
  <query_time>20.09.2012 10:58:15</query_time>
  <query_time_local>20.09.2012 16:58:15</query_time_local>
</result>
```

Ниже описано назначение параметров:

Параметр	Описание
query_name	Строковое наименование типа запроса
query_result	**Ok** — если запрос выполнен успешно **Error** — если произошли какие-либо ошибки
query_msg	Строковый комментарий по результатам выполнения запроса
query_time	Время запроса UTC
query_time_local	Время запроса локальное

В ответе также могут быть тэги, специфичные для конкретного типа запроса.

5.1 Получение данных счётчика посетителей

Для получения данных счётчика посетителей используется запрос **get_people_counters**.

Параметрами данного запроса являются:

```
<channel_id>cacdd8e6-1c56-435c-86e3-6967d7494a50</channel_id>
<search_time>2012-09-17 09:50:00</search_time>
```

Параметр	Описание
channel_id	Идентификатор канала, на котором настроен счётчик посетителей
search_time	Момент времени, для которого необходимо выдать показания счётчика. Время указывается в формате **yyyy-MM-dd HH:mm:ss**. При этом указывается локальное время сервера, на который отправляется запрос

В ответ сервер возвращает следующие параметры:

```
<in>434</in>
<out>378</out>
```

Параметр	Описание
in	Количество вошедших людей для данного счётчика
out	Количество вышедших людей для данного счётчика

6. Организация вещания видео на сайт

Вещание видео на сайт может быть организовано с помощью службы мобильных подключений сервера Macroscop и компонентов на клиентской стороне (в браузере).

6.1 Вещание с помощью Flash

Вещание видео на сайт может быть организовано с помощью службы мобильных подключений сервера Macroscop и Flash-компонента на клиентской стороне. Пример использования компонента на HTML-странице размещен в папке **Examples\SiteFlash**. На HTML-странице (**index.html**) необходимо указать параметры подключения к серверу Macroscop, а также идентификатор (имя/номер) канала, с которого должно транслироваться видео и требуемый кодек (H.264 или MJPEG).

Пример настройки:

```
var flashvars = {
            server: "demo.macroscop.com", // адрес сервера
            port: "8080", // порт сервера
            login: "root", // имя пользователя
            password_hash: "", // md5-хэш пароля
            mode: "MJPEG," // предпочитаемый формат видео
            channel: "1" // имя, номер или идентификатор канала
        };
```

Идентификаторы всех каналов в системе могут быть получены с помощью соответствующего запроса (см. Получение конфигурации системы, стр. 36).

Параметр **Предпочитаемый формат видео** (**mode**) может принимать значения **MJPEG**, **H264** или вообще пропущен. Если предпочитаемый формат видео не задан, то автоматически будет выбран подходящий формат. Значение **H264** можно указать только для камер, транслирующих видеопотоки, закодированные в H.264. Значение **MJPEG** можно указать для всех камер, однако в случае перекодирования из H.264 это может привести к повышенной нагрузке на сервер.

6.2 Вещание видео на сайт с помощью JavaScript (устарело)

Данный способ является устаревшим, поскольку создает повышенную нагрузку на сервер Macroscop и предоставляет худшее качество по сравнению с вещанием на сайт с помощью Flash-компонента.

Вещание видео на сайт может быть организовано с помощью сервера Macroscop и JavaScript-компонента на клиентской стороне. Скрипт для клиентской стороны и пример его использования на HTML-странице размещен в папке **Examples\Site\frameReceiver.js**. В скрипте необходимо указать параметры подключения к серверу Macroscop, а также идентификатор (имя/номер) канала, с которого должно транслироваться видео и требуемый размер области, в которую будут выводиться видеокадры.

Пример настройки скрипта:

```
var serverUrl = "http://95.23.84.1:8080" /*URL сервера*/
var login = "root" /*пользователь, имеющий права на просмотр транслируемого канала*/
var password = ""; /*MD5-хэш пароля пользователя в верхнем регистре или пуста строка, если пароль пустой*/
var channelnum = 0; /*порядковый номер канала в общей конфигурации, счет с 0*/
var drawWidth = 577; /*ширина области отображения, в пикселях*/
var drawHeight = 432; /*высота области отображения, в пикселях*/
```

Пример скрипта, использующего идентификатор канала вместо его порядкового номера, размещен в папке **Examples\Site\frameReceiver_id.js**.

На самой HTML-странице должен быть размещен тег ****, в котором будет отображаться видеопоток, закодированный в MJPEG.

Не рекомендуется изменять размеры области отображения видео динамически, поскольку это приведет к существенному повышению потребляемых ресурсов службой мобильных подключений, так как эта служба перекодирует в MJPEG исходный видеопоток, после чего полученный поток кадров разделяет между многими клиентами (сессиями). Использование разных разрешений также приведет к дополнительной нагрузке на службу мобильных подключений.