Milestone
XProtect®

Analytics 2.2
Generic VA Interface
Developer's Manual

The Open Platform Company

Target Audience for this Document

This document is intended for video content analysis system developers wishing to integrate with Milestone XProtect Analytics. The content of this document is not relevant for regular surveillance system users.

Contents

Copyright, Trademarks and Important Information

Copyright

Trademarks

Disclaimer

Introduction

Video content analysis—VCA—has over the couple of years matured from research projects at labs and universities into products that can add substantial help to companies in their day-to-day surveillance tasks.

Currently, completely omitting the human monitoring part is not really realistic since most systems rely heavily on having a human to validate the results produced from the VCA systems. However, the VCA systems make it lot easier to secure large areas since the software is far better on keeping an eye on the details than a human would be after only a very short amount of time.

The functionality of the VCA systems spans from face recognition over advanced motion detection to complex behavioral analysis, where various types of abnormal behavior both of humans and vehicles can be detected.

Currently, there are roughly 30 important players within the area of delivering VCA systems. This number keeps growing, and the increased competition will keep improving the functionality and performance of the systems.

False Alarms

One of the messages we often hear from the market in relation to the use of VCA systems is the problem referred to as *false positives*. False positives are situations where the VCA system reports some kind of observed behavior or motion, but where the reports turn out to be caused by different types of distortions in the input video or observed scenery, and are insignificant seen from a security perspective.

In certain cases up to several thousand false alarms are reported during a day, which surely devaluates not only the functionality and success of the concrete installations, but also the reputation and general perception of the whole area within digital video analysis.

One could of course argue that such high numbers of false alarms are results of bad system decisions and poor configuration, and that may to some extent be true in many cases. However, this is the reality for many customers and users, so—regardless of the reason behind the problem—users will end up with a system that really does not solve their problem.

XProtect Analytics 2.2

One of the cornerstones in the Milestone surveillance product suite has always been the openness and flexibility that allows for the customer to use more or less any IP camera he wants. Replacing one camera with another—possibly even from a different manufacturer—should be an easy task.

The same basic idea can be applied to the *Milestone XProtect Analytics 2.2* (XPA 2.2) framework. Milestone XProtect Analytics 2.2 is an open, plugin-based platform in which VCA systems can be integrated and hook in their functionality to Milestone.

Multiple VCA systems can co-exist at the same time within the same framework, and may even work together to create an optimal solution for a given customer in a given situation.

The framework is an add-on product, and it builds on top of several of the existing Milestone components:

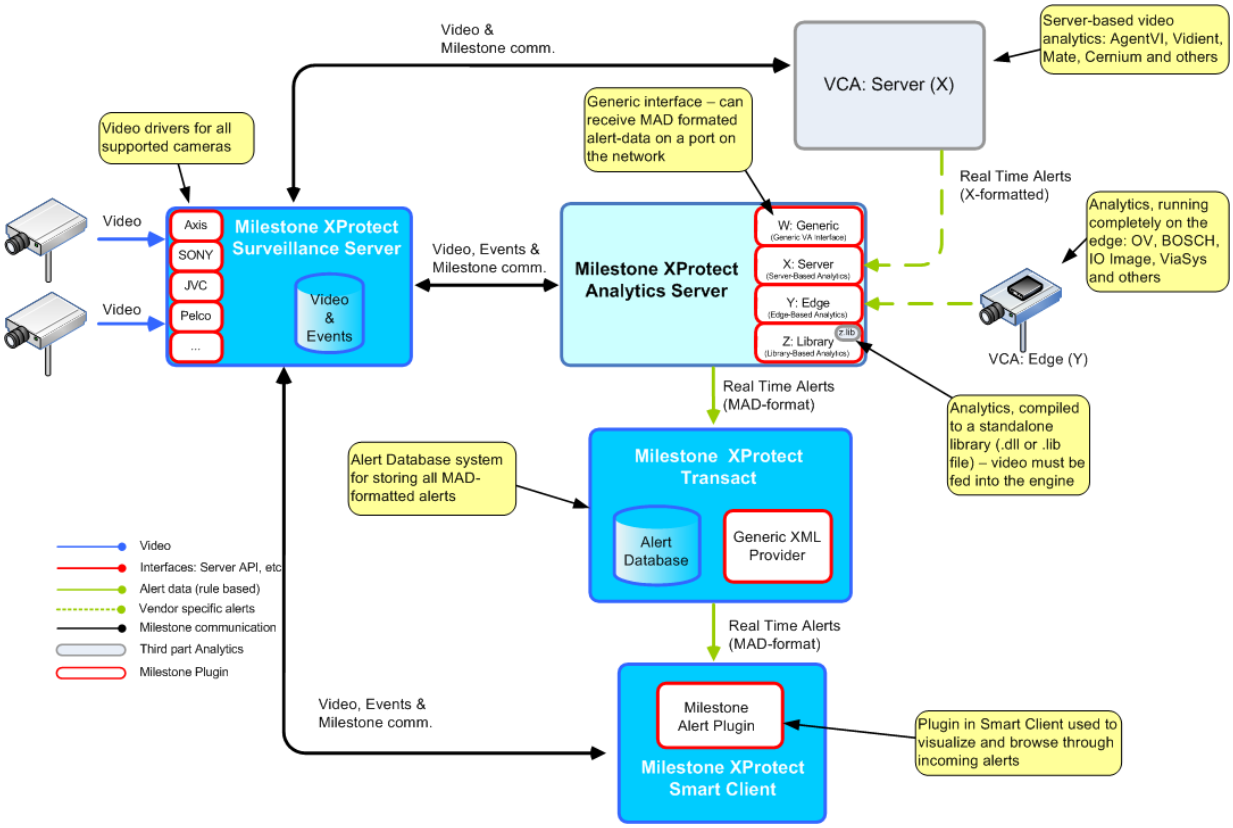
- Milestone Surveillance Server (XProtect Enterprise or XProtect Corporate)

- Milestone XProtect Transact
- Milestone XProtect Smart Client

In the following, we describe the architecture and data flow of the framework. We will also present how to integrate analytics functionality and get alerts into the Milestone Smart Client through the common Milestone Alert Data format and the Generic VA Interface plugin.

Architecture and Data Flow

The overall architecture is as follows: for each concrete VCA system, a driver is developed which acts as the glue between the analytics system and the Milestone analytics framework. This driver must implement a certain interface, and can thereby be plugged into the analytics server, which is responsible for loading the plugin at runtime.



Milestone XPA 2.2 overall architecture and dataflow. Drivers, analog to the video camera drivers, connect the underlying VCA systems with the analytics server. For a larger version of the illustration, see Appendix A.

The plugin is completely responsible for controlling the communication with the underlying analytics system, and is the only runtime component in which vendor specific information is implemented.

In general, VCA systems are distributed in one of the following ways:

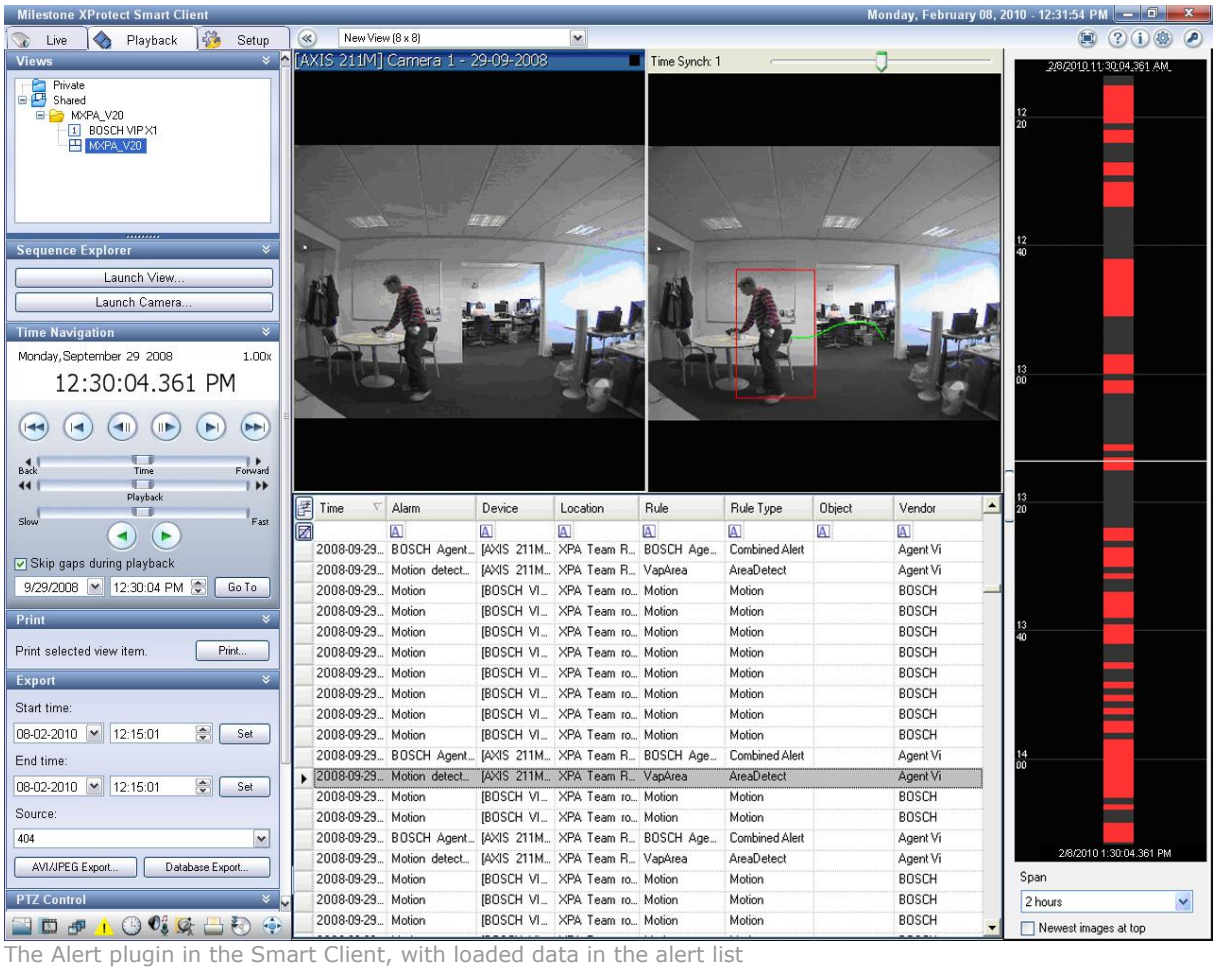
- As a complete, standalone server system (server-based)
- As a library which must be built into software (library-based)
- As part of the firmware on the device (edge-based)

Some systems have analytics put on both the edge and on the server, where a pre-processing is done on the camera and the server handles the core part of the processing. However, in this context, the server would still be our point of entry, and the analytics framework would not do anything about the pre-processing on the edge.

In the above figure, the three different types of analytics are denoted by the X, Y and Z. In version 2.2 of the analytics framework, only real-time alerts are handled. This means that metadata in general is not collected and stored. This will be handled in later versions.

The overall data flow of the framework is as follows: the underlying analytics systems detects a predefined behavior, which is configured by setting up *rules*, and sends out an alert notification to its surroundings in a format specific to the vendor. Through configuration, the analytics driver is set up to catch the alerts.

All caught alerts are transformed into a common format defined by Milestone, namely the *Milestone Alert Data* (MAD) format, and stored in a common database. A plugin has been developed for the Smart Client, which pulls the alert information from the database and displays it. By switching to browse mode in the Smart Client, users can browse through recorded alerts, and for each alert see a marked up image of the alert situation.



The Alert plugin in the Smart Client, with loaded data in the alert list

To view the video clip that corresponds to the alert situation, the standard playback functionality in the Smart Client can be used.

The Common Alert Format (MAD)

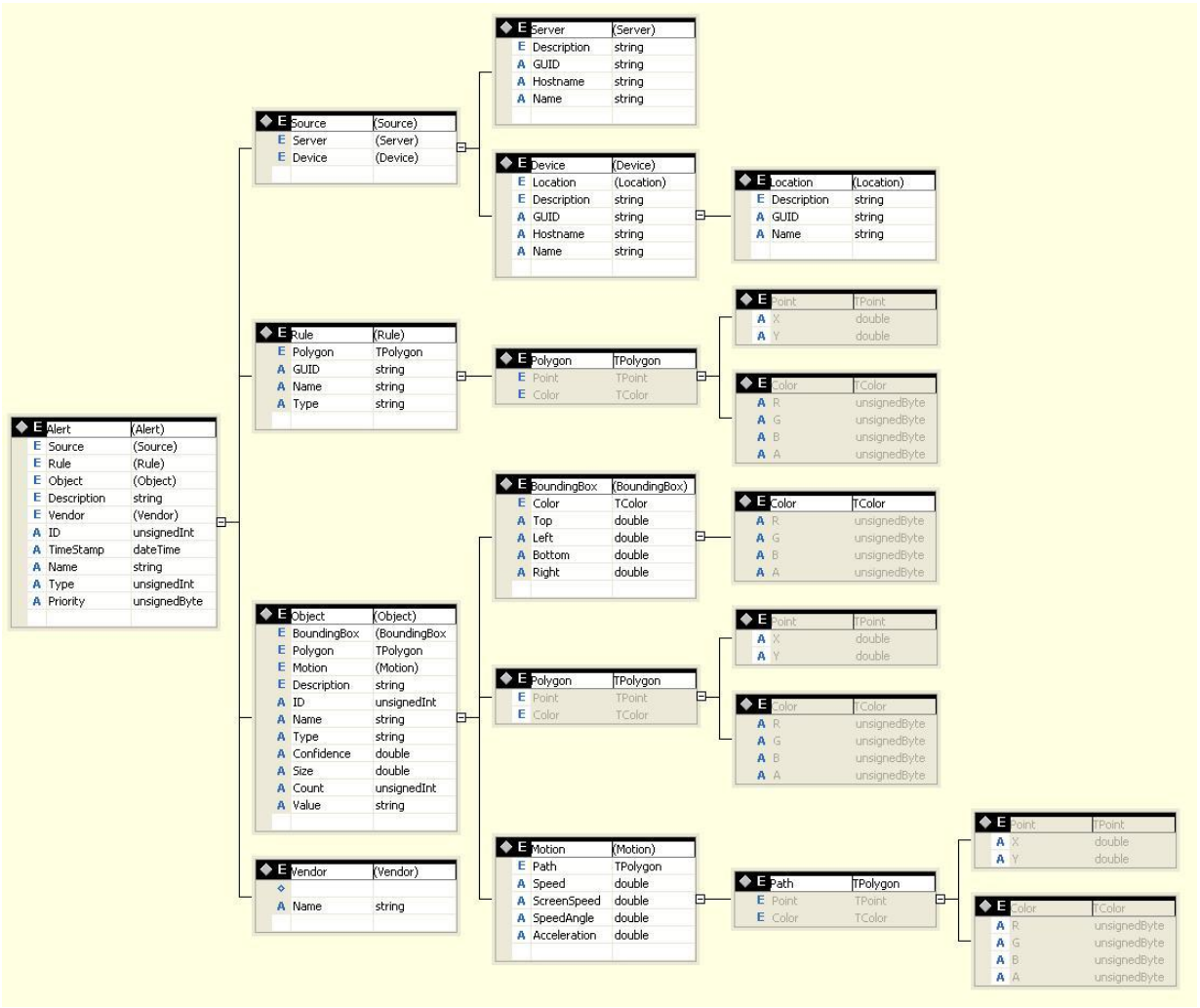
The common alert format is a very essential part of the analytics framework. All alerts, regardless of the actual vendor-specific system they originally came from, are stored side-by-side in the same database and can be displayed in the same user interface control since the format of data is known to the system.

The alert format has been elaborated based on alert samples collected from a wide range of VCA vendors, and the intention with the format is that it supports all the common fields that usually are stored in alerts. Having this "super-format," it will be possible to convert any alert from any system into this. Since we do not wish to throw away data, we have designed a special place-holder where

the original alert XML can be stored. In the future, if vendors would like to create their own interface, special and highly vendor-related information can still be pulled out.

Surely, there are various attempts to create industry standards for this particular purpose, and one could argue in favor of using such industry standards instead of creating our own. Furthermore, there are several companies pushing out their version of the alert format. However, as long as all companies do not agree on using one common format for handling real time alert information and analytics data in general, the Milestone Alert Data is a very pragmatic way of handling and eliminating the differences between the VCA systems and their formats respectively.

The MAD-format is XML based, like most of the existing alert formats, and is defined by the XSD (XML Schema Definition) file *Alert.xsd* that can be found in the *Xsd* folder in the distribution. The following figure shows the structure of the alert format in a more visual way:



Graphical representation of the XSD structure; for a larger version of the illustration, see Appendix B

Alert API

To create MAD-formatted alerts, we have implemented a .NET-based API where alerts can be created programmatically. The key-advantage gained by using the API instead of just building the XML as concatenation of substring is of course to guarantee the integrity and validity of the XML. All the relevant classes in the API have support for COM interfaces which means that the API can be used from native C++ code as well. In the samples folder, a Visual Studio 2005 project

demonstrates how to use the Alert API from C# code. The following XML snippet is an example of an alert:

```
<?xml version="1.0" encoding="utf-8"?>
<Alert xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" ID="0" TimeStamp="2008-09-
29T20:06:02.0720778+02:00" Name="Hallway_Tripwire" xmlns="http://tempuri.org/Alert.xsd">
  <Source>
    <Server GUID="CCCF8C51-94DB-49b2-9718-0800A151D76B" Name="10.10.11.26">
      <Description>Milestone Surveillance Server</Description>
    </Server>
    <Device GUID="D23414E5-B03F-4C78-8AB9-A36D1488F9C2" Hostname="10.10.50.28" Name="[Axis 221]
Camera 1">
      <Location Name="Hallway Corner" />
    </Device>
  </Source>
  <Rule Name="Hallway Tripwire" Type="Tripwire">
    <Polygon>
      <Point X="241" Y="63" />
      <Point X="65" Y="140" />
      <Point X="75" Y="284" />
      <Point X="115" Y="402" />
      <Point X="238" Y="448" />
      <Point X="398" Y="462" />
      <Point X="521" Y="374" />
      <Point X="494" Y="235" />
      <Point X="486" Y="63" />
      <Point X="303" Y="37" />
      <Color R="0" G="0" B="255" A="255" />
    </Polygon>
  </Rule>
  <Object ID="42" Name="John Doe" Type="Person" Size="2.39">
    <BoundingBox Top="194" Left="286" Bottom="429" Right="408">
      <Color R="255" G="0" B="0" A="255" />
    </BoundingBox>
    <Motion Speed="1.816466" ScreenSpeed="0.2389245" SpeedAngle="1.546658">
      <Path>
        <Point X="88" Y="288" />
        <Point X="129" Y="300" />
        <Point X="150" Y="304" />
        <Point X="169" Y="307" />
        <Point X="186" Y="311" />
        <Point X="201" Y="313" />
        <Point X="215" Y="315" />
        <Point X="229" Y="316" />
        <Point X="242" Y="316" />
        <Point X="255" Y="316" />
        <Point X="262" Y="316" />
        <Point X="272" Y="315" />
        <Point X="284" Y="314" />
        <Point X="296" Y="312" />
        <Point X="307" Y="310" />
        <Point X="318" Y="309" />
        <Point X="327" Y="308" />
        <Point X="332" Y="308" />
        <Color R="0" G="255" B="0" A="255" />
      </Path>
    </Motion>
  </Object>
  <Description>Tripwire at main door</Description>
  <Vendor Name="VCA Vendor" />
</Alert>
```

For a detailed description of the alert format, please refer to the Xsd file and the Alert API reference guide found in the *doc* (documentation) folder.

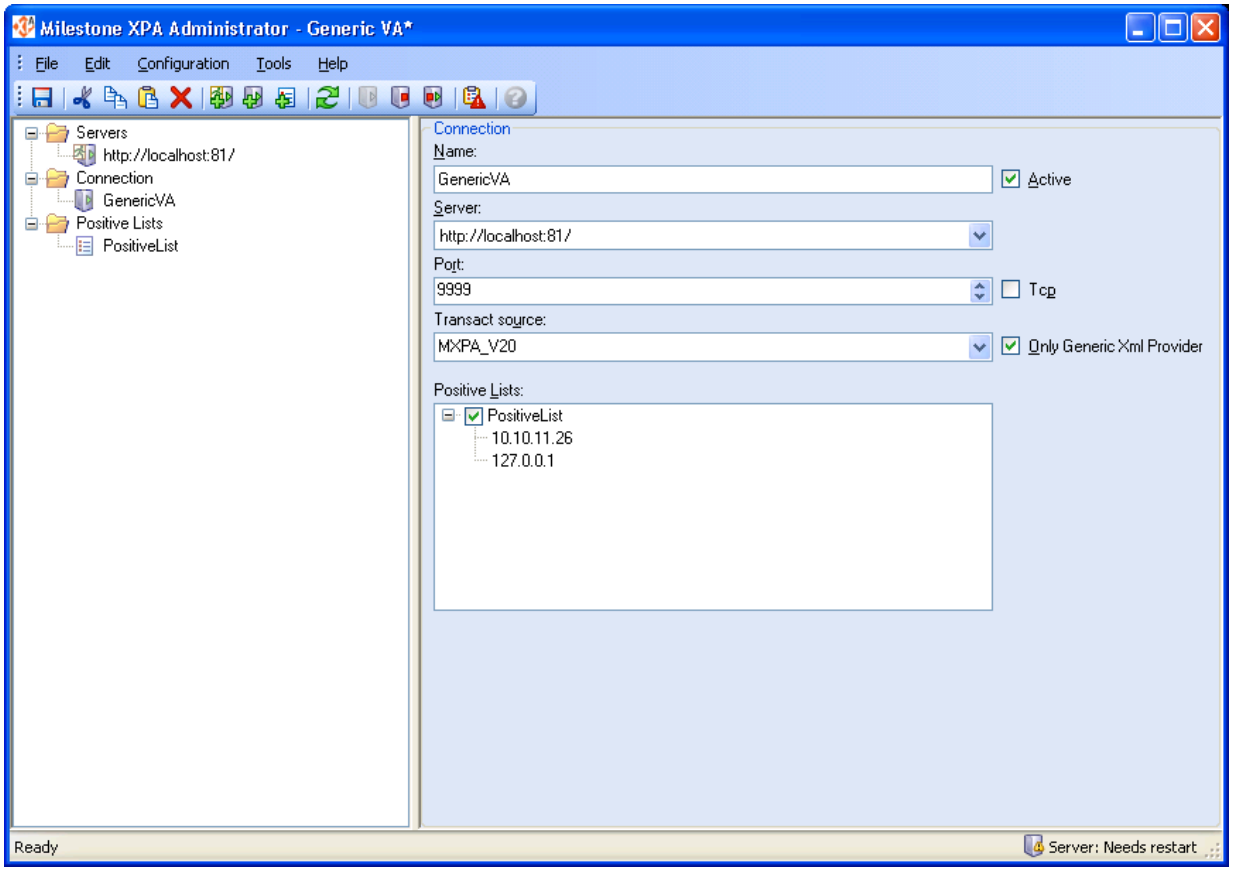
Integration through Generic VA Interface

As mentioned previously in the document, the communication between the analytics framework and the underlying VCA system is done through a driver which is implemented specifically for that vendor. However, there is another way of delivering alerts into the system, and that is through the *Generic VA Interface*.

The Generic VA Interface is essentially a driver, implemented as a "proxy" through which alerts are passed to the alert database. It listens on a specific port on the network (configured by the user), and if data arrives and has the correct MAD format, it will be serialized directly into the alert database. Using the Generic VA Interface, the only thing required in order for VCA system vendors to deliver their alerts, is to do the alert transformation themselves and send the transformed alert to the generic driver. The generic driver will validate the data, and if the data is correctly formatted, the data will show up in the Smart Client.

Configuring the Generic VA Interface

The generic driver is configured similarly to the other existing integrations. Using an administration user interface, a configuration is created. The configuration can then be loaded when the server starts the plugin.



Administration user interface for configuring the generic driver. Positive lists are used to control who are allowed to send data to the system. In this case the server will listen on port 9999 using the UDP protocol.

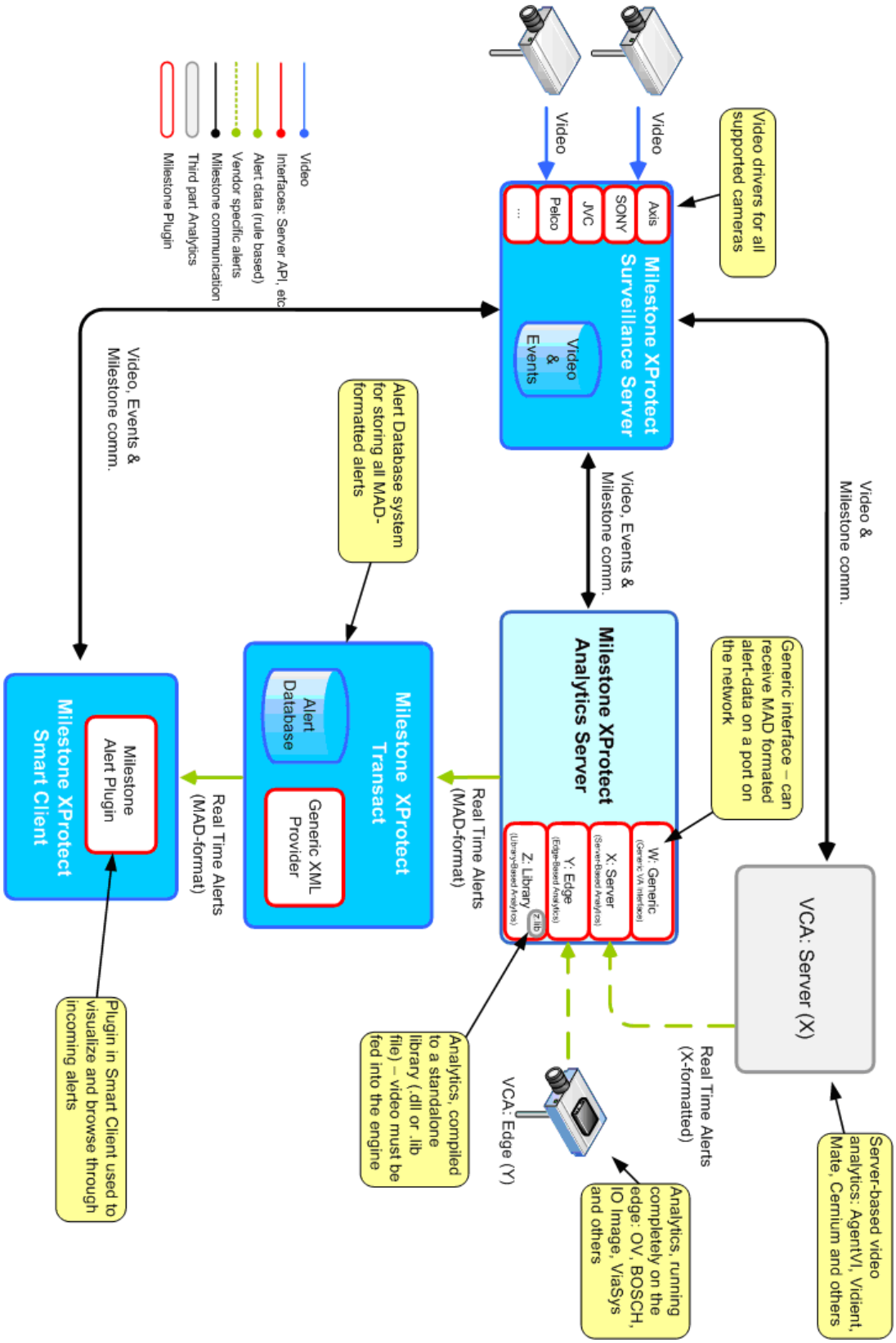
Multiple *connections* (or *channels*) can listen for multiple ports simultaneously, and users can specify which Transact source the data should be stored in. Using the concept of *positive lists*, it is possible to control which network components (specified by IP address) are allowed to send data to the database. In later versions, support for specifying sub-domains, so that any network component within a local subnet can be granted access, will be added.

Alert Sample

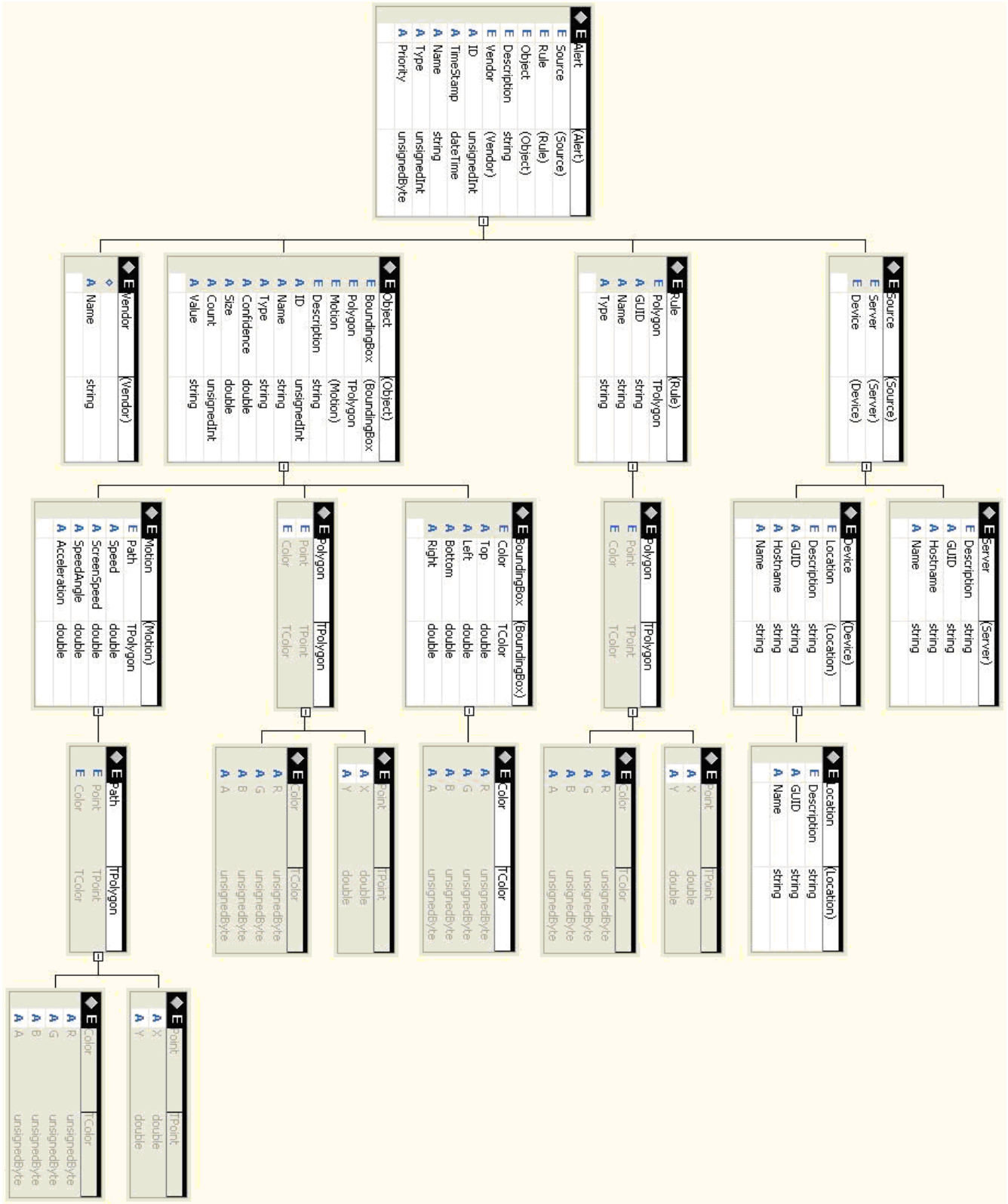
To see how the alert API can be used, refer to the sample located in the *samples* folder.

Note that the sample is a C#-based .NET program. COM interfaces are added to the API and can just as well be used through native C++.

Appendix A


Video
Interfaces: Server API, etc
Alert data (rule based)
Vendor specific alerts
Milestone communication
Third part Analytics
Milestone Plugin
Video
Video
Video drivers for all supported cameras
Axis
SONY
JVC
Pelco
...
Milestone XProtect Surveillance Server
Video & Events
Video, Events & Milestone comm.
Video & Milestone comm.
Alert Database system for storing all MAD-formatted alerts
Milestone XProtect Analytics Server
W: Generic (Generic VA Interface)
X: Server (Server-Based Analytics)
Y: Edge (Edge-Based Analytics)
Z: Library (Library-Based Analytics)
.lib
Generic interface – can receive MAD formated alert-data on a port on the network
VCA: Server (X)
Milestone XProtect Transact
Alert Database
Generic XML Provider
Real Time Alerts (MAD-format)
Milestone XProtect Smart Client
Milestone Alert Plugin
Real Time Alerts (MAD-format)
Analytics, compiled to a standalone library (.dll or .lib file) – video must be fed into the engine
VCA: Edge (Y)
Real Time Alerts (X-formatted)
Plugin in Smart Client used to visualize and browse through incoming alerts
Analytics, running completely on the edge: OV, BOSCH, IO Image, ViaSys and others
Server-based video analytics: AgentVI, Vidient, Mate, Cernium and others

Appendix B


Alert
Source (Source)
Rule (Rule)
Object (Object)
Description string
Vendor (Vendor)
ID unsignedInt
TimeStamp dateTime
Name string
Type unsignedInt
Priority unsignedByte
Source
Server (Server)
Device (Device)
Rule
Polygon TPolygon
GUID string
Name string
Type string
Object
BoundingBox (BoundingBox)
Polygon TPolygon
Motion (Motion)
Description string
ID unsignedInt
Name string
Type string
Confidence double
Size double
Count unsignedInt
Value string
Vendor
Name string
Server
Description string
GUID string
Hostname string
Name string
Device
Location (Location)
Description string
GUID string
Hostname string
Name string
Location
Description string
GUID string
Name string
Polygon
Point TPoint
Color TColor
Point
X double
Y double
BoundingBox
Color TColor
Top double
Left double
Bottom double
Right double
Color
R unsignedByte
G unsignedByte
B unsignedByte
A unsignedByte
Motion
Path TPolygon
Speed double
ScreenSpeed double
SpeedAngle double
Acceleration double
Path
Point TPoint
Color TColor

Milestone Systems offices are located across the world. For details about office addresses, phone and fax numbers, visit www.milestonesys.com.

milestone

The Open Platform Company