WHITE PAPER

sureview

Integrating your Security Operation

A ROADMAP FOR CONNECTING TECHNOLOGY TO DELIVER OPTIMUM OPERATIONAL VALUE

TABLE OF CONTENTS

INTEGRATING YOUR SECURITY OPERATION

A ROADMAP FOR CONNECTING TECHNOLOGY TO DELIVER OPERATIONAL VALUE

WHAT DRIVES THE NEED TO INTEGRATE SYSTEMS?

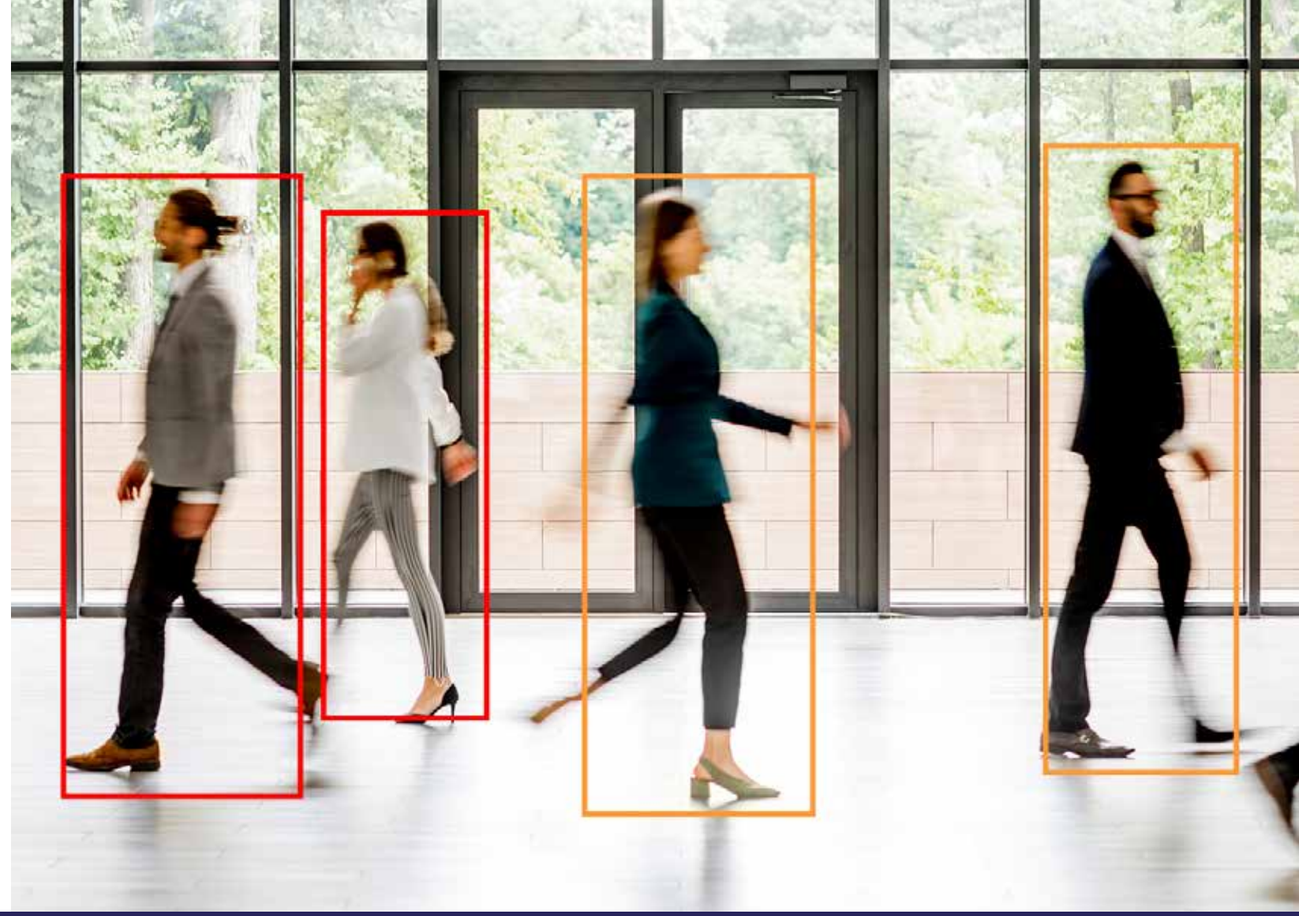
For most organizations the pace of change continues to accelerate. For security teams this change has profound impacts: they are responsible for protecting more locations, assets, and people while using an ever-expanding list of systems and devices. To make it even more challenging, the systems and devices they rely on to monitor their assets were often installed by different teams at different times. When an incident occurs, security response teams are often left with an inconsistent and disjointed process that involves them jumping back-and-forth between one system and another, trying to piece together what happened. It can be time-consuming, is prone to human error, and rarely delivers the security goals of the organization. In order to meet these challenges and provide a consistent level of security response, organizations typically look to integrate their systems to achieve a single operational view.

DELIVERING BETTER SECURITY OUTCOMES

Traditionally security integration has meant connecting hardware, i.e. connecting a door alarm to a nearby camera. However, today's integrations also involve connecting software and networks across a vast array of different systems and deployments. In addition to the traditional security systems (access control, burglar alarms, CCTV, etc.), organizations today are leveraging technologies such as shooter detection systems, business risk alerts, video analytics, mass notification, mobile tracking tools, and geospatial technologies to provide their teams with real-time awareness of their security threats. So, how does an organization deliver the simple goal of a responsive security operation when change and complexity are expanding all the time? This white paper will answer that question by providing a roadmap for security professionals to plan and execute any integration project.

START WITH AN OPEN RESPONSE PLATFORM

So, you have all of these systems that you want to integrate into a single, holistic platform, but first, you need to select that platform. The security operations team is responsible for coordinating the response to real-time security incidents, and in order to do this effectively, they need a Next-Gen PSIM (Physical Security Information Platform) or response platform. In our white paper "Next-Gen PSIMs - the Top 10 Things Leading Commands Expect Today", we cover what organizations should look for in these platforms. However, here are a few characteristics of these systems that should be looked at specifically when embarking on security integration projects.



CHECKLIST FOR CHOOSING A RESPONSE PLATFORM

- Does it have a universal alarm queue?
  Can we bring alarms from various types of systems (i.e. physical security or business systems)?
- Does it provide a place to organize workflows?
  What do you do when event X happens at location Y?
- Universal plugins that support standard protocols?
  Does it provide support for industry-standard, simple onboarding that reduces maintenance and costs?
- Does it have an open API to integrate with back-office systems?
  Is there a published API that allows us the flexibility to integrate with our unique systems and processes?
- Does it provide databases for centralized reporting?
  Can the system provide databases from our entire security operation rather than siloed systems?

DEVELOPING YOUR INTEGRATION REQUIREMENTS

4 MINIMUM REQUIREMENTS EVERY SECURITY OPERATIONS TEAM NEEDS FROM INTEGRATIONS

The security operations team's primary goal is to efficiently respond to real-time events to protect the staff, visitors, and assets of the organization. To achieve this, every SOC has a minimum set of requirements that they need from integrated security systems, they are:

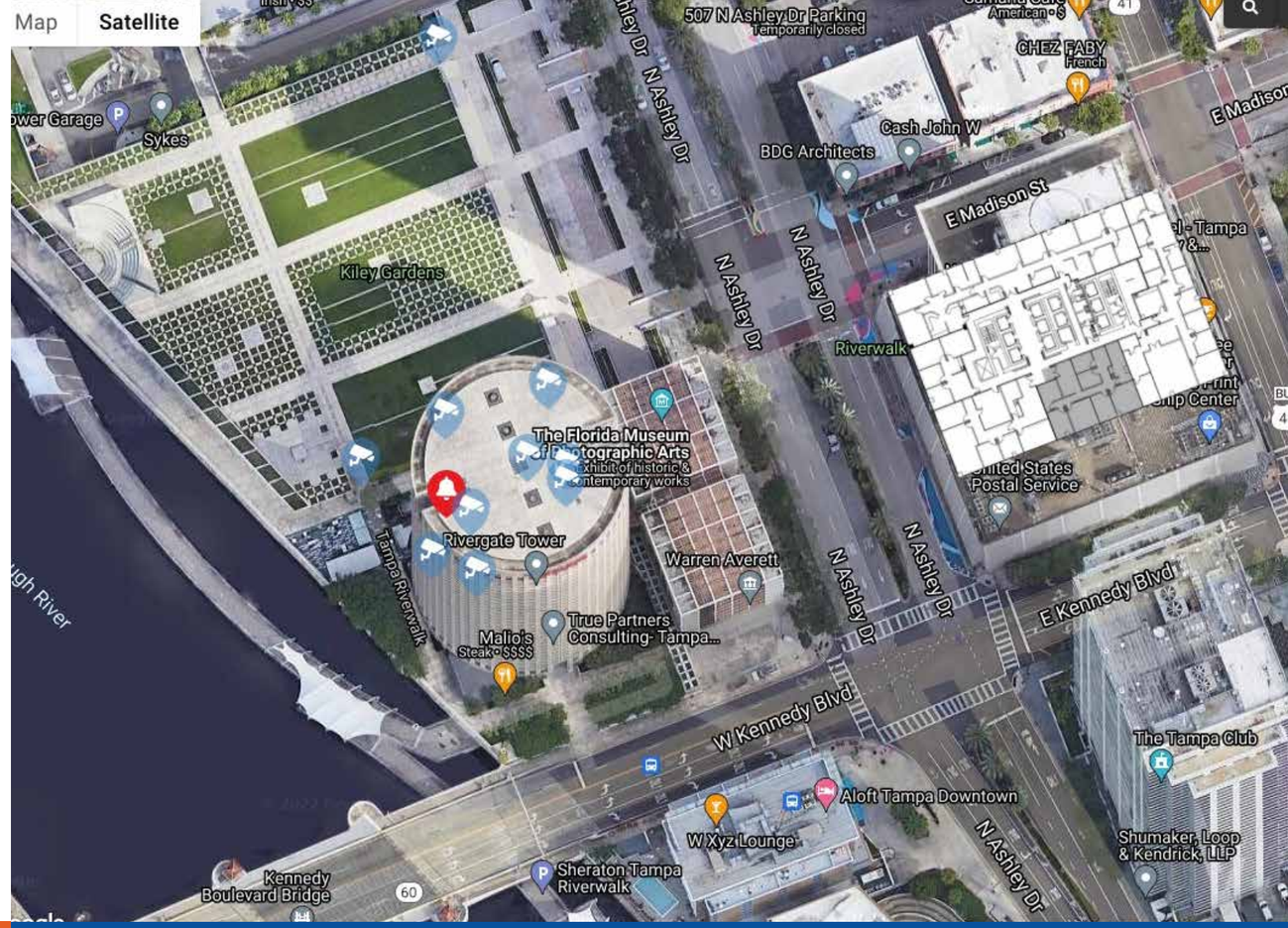
1

Real-time Alarms / Alerts

These alarms and alerts can be triggered for a wide variety of reasons including door events, video motion and analytics, I/O (input/output) and PIRs (Passive Infrared Sensor) connected to alarm systems, temperature sensors on IoT devices, intercom buttons, the list is almost endless.

To properly respond, command center operators need these alarms to include:

- *Date and time of the alarm*
- *Type of alarm: what triggered the event*



2

Location of the alarm

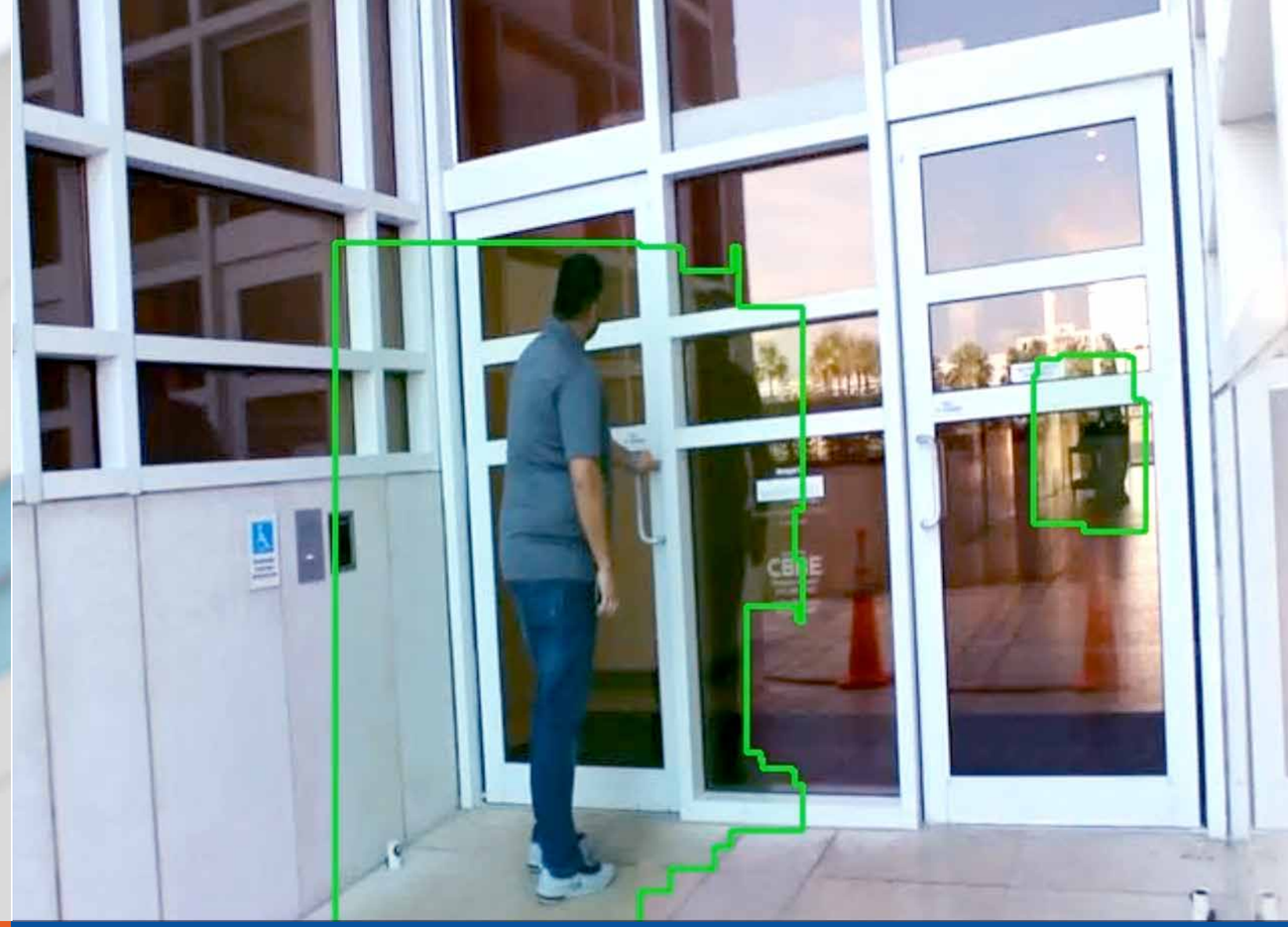
To quickly coordinate a response, operators need to know exactly where the alarm occurred. To achieve this the alarm should include:

- *The building name, floor, area/zone the alarm occurred in*
- *Ideally, the geo-location of the alarm so it can automatically be plotted on a map*

3 Stream nearest live cameras

To get real-time situational awareness, the operator needs to be able to immediately see live camera feeds close to the triggered event

- *Use geospatial association to quickly show the live camera feeds nearest the incident*



4 Video clip of the triggered event

Wherever and whenever possible, provide a recorded clip of the event

- *A short clip from the camera that triggered the event (i.e. analytics) or from a nearby camera provides extra context for quick response*

CHECKLIST BEFORE YOU START AN INTEGRATION PROJECT

Before the security technology team embarks on an integration project, it's important to determine the types of alarms that operations teams need to respond to. Today's systems and devices can generate a large volume of alarm types, many of which create noise and provide no operational value to responding operators. Anyone who's run an operations center knows how detrimental unnecessary, or false, alarms can be. In the worst-case scenario, real events are missed in the clutter. More commonly, response times increase as operators wade through the sea of alarms, impacting delivery and lowering morale.

HERE ARE A FEW QUESTIONS TO ASK YOUR OPERATIONS TEAM AS YOU DETERMINE WHICH ALARMS ARE MOST IMPORTANT FOR THE SOC

Alarm Types

Which alarms and events does the SOC need to know about?

Critical Points

What are the critical points within a building/area? Prioritize these points and alarm types for the SOC.

Schedule

When do they need to know about these alarms? Do they only need to know about these alarms at certain times of the day?

Compliance

Are there specific alarms or events that must be audited for compliance? Do all of the events require operator action or do they merely need to be logged?

Reporting

What reporting requirements does the SOC leadership have? This should focus on data that provides an operational view (not a system view) of the organization.

GROUP ALARMS INTO ONE OF THESE 3 CATEGORIES

With this information in hand, the security technology teams can then simply categorize their alarms into 3 categories: Alarms that—

1. REQUIRE OPERATOR RESPONSE

Operators need to take action and respond to these alarms in real-time.

2. ARE MASKED BUT LOGGED

Operators don't need to take action, however, for reporting or compliance these alarms need to be logged.

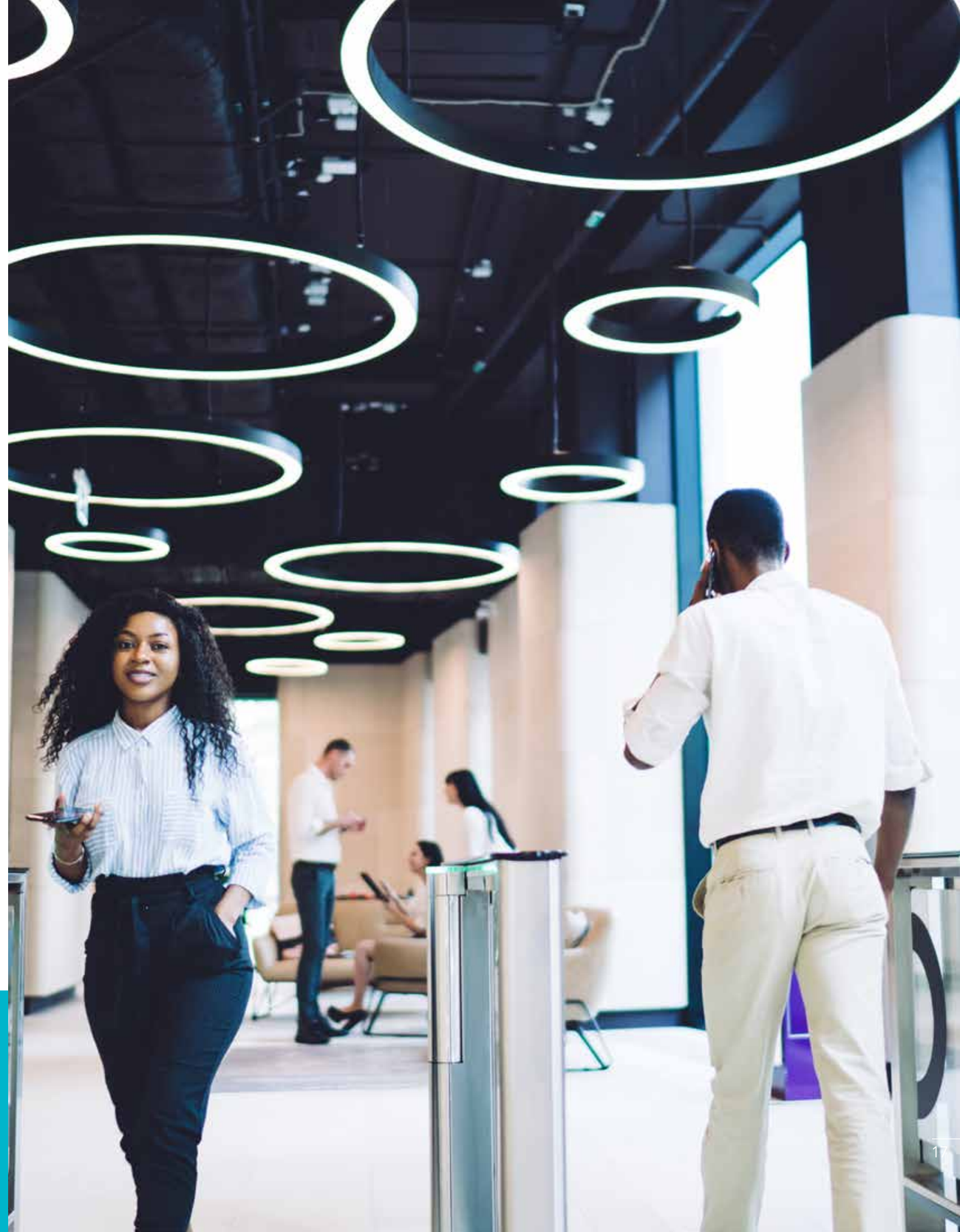
3. CAN BE IGNORED

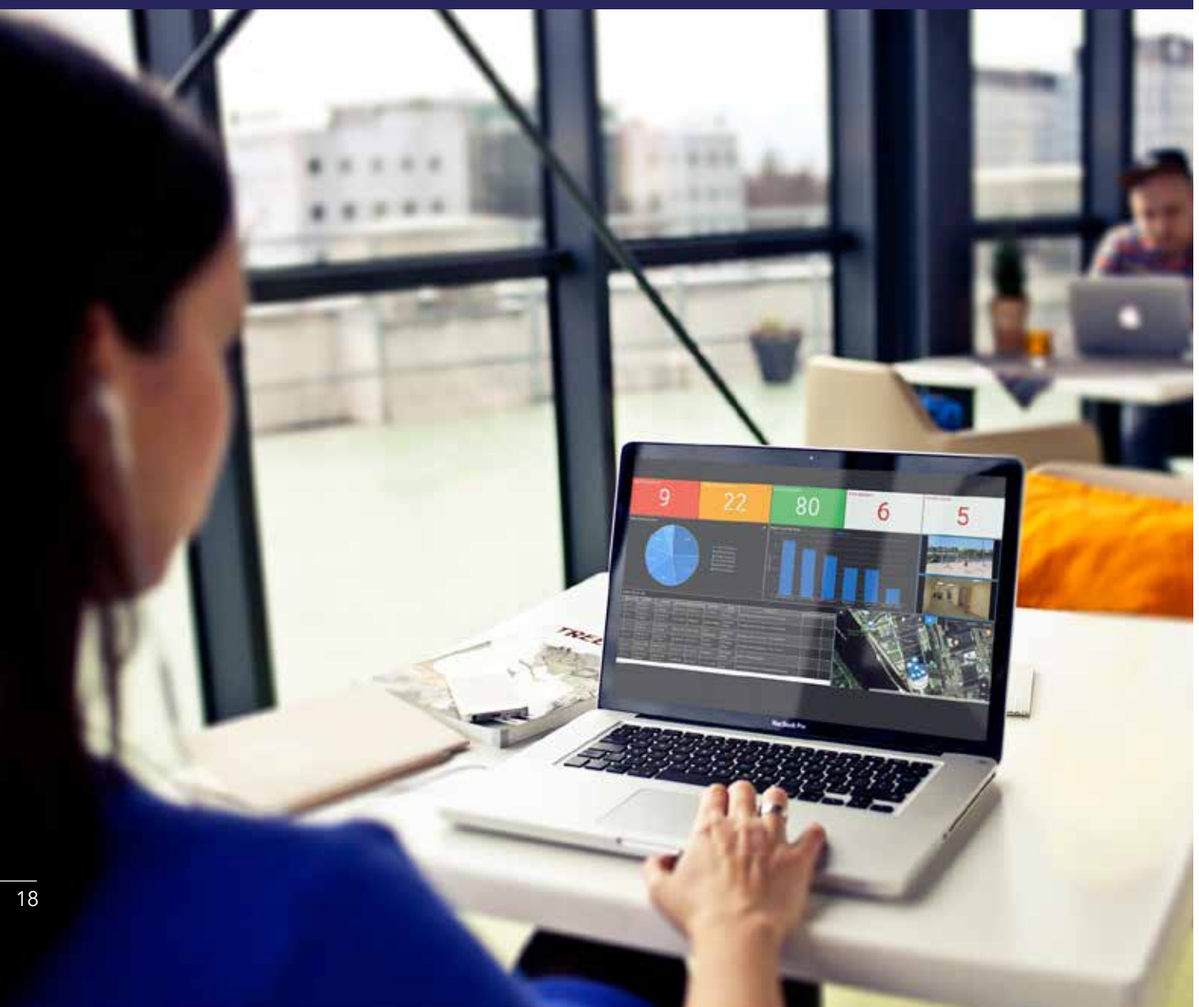
The SOC doesn't need to be notified of the alarms for action or logging. System messages are a good example of these types of alarms.

SCOPING THE INTEGRATION PROJECT—FOCUS ON THE USER STORY

Utilizing the minimum requirements and checklist discussed above, security technology teams can begin to develop the scope of the integration project. The scope is important to ensure that operations, security technology, IT, and the vendor are all clear on the goals of the project. Having spent over 20-years building integrations with security systems, we have developed a lot of scope documents over the years. The best piece of advice we can give when developing these scopes is to focus on the user story.

The user story is a simple, non-technical description of what the end-user, in this case, the operator in the command center, requires from this integration. It might seem simplistic and repetitive, but many of the people involved in these projects have never spent a day working in a command center. It's understandable that they cannot make assumptions about what is important. With highly technical teams it's very common for integration projects to focus on the technical implementation, i.e. opening ports, network configurations, etc. Though these are critical tasks on their own, they more often "become the project" but are not a part of delivering value to the end-user.





AN EXAMPLE OF A SIMPLE USER STORY MIGHT BE:

Operators in the command center are responsible for monitoring alerts from our business risk system that provide updates on outside events that may impact our offices. The type of alerts range from civil disturbances such as a protest, to weather alerts that might impact travel and deliveries. Today, these alarms come into a central email box that the operations teams have to monitor in addition to their other security events.

There is no way to properly track who responds to these alerts, what actions they need to take, or to enforce any internal SLA. The goal would be for these alerts to be sent to our security response system, where we could associate events with action plans and prioritize these based on the type of event and proximity to our primary campus.

The information required for the operator to respond properly is:

- *Time of the event*
- *Type of event - e.g. protest, weather warning, traffic incident, etc.*
- *Location impacted by this event - e.g. building name*
- *Specific location - we would like to see this on our map*
- *Extensive details about why this event was triggered*

Additionally, the manager of the SOC would like these alarms and their responses automatically added to the daily activity report that tracks all of the other security events on the campus.

Notice how this user story focuses on the problem and the desired outcomes, not on the technical implementation? How the security technology and IT team choose to solve this problem technically can then be developed with a clear understanding of the desired outcome.

TECHNICAL IMPLEMENTATION—STANDARDS-BASED AND NATIVE INTEGRATIONS

Once the user story has been established, the technology teams can determine the best way to integrate the technology for the operations team. There are two common ways to integrate systems, one utilizing standards-based protocols, and another using native integration using the SDK or API from the vendor. Let's explore both options and understand the pros and cons of each approach:

STANDARDS-BASED INTEGRATIONS

Standards-based integrations utilize common industry protocols and standards to connect systems. As the security industry has evolved and more systems have moved to Cloud technology, the adoption of common standards has greatly increased. By leveraging these protocols, security technology teams do not need to embark on complex development projects to connect software systems, they simply need to configure their existing systems to stream or send alerts.

LIVE STREAMING

There are two commonly adopted video-streaming protocols, RTSP (Real-Time Streaming Protocol) and ONVIF (Open Network Video Interface Form). You will find that the majority of IP camera and video surveillance systems will support one or both of these protocols. Both allow for secure streaming of live video feeds over the internet. No integration is required to access these cameras—as long as they are accessible on the network, operators can immediately connect and stream them.

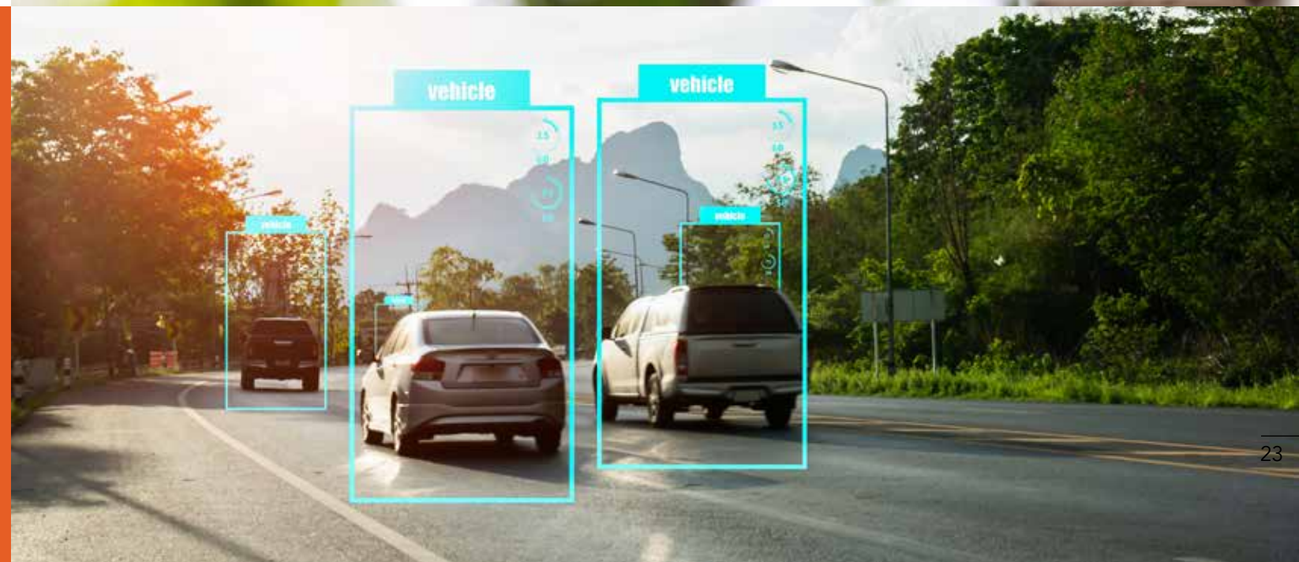
ALARMS

Nearly every security system in the market can send an email (SMTP) when an event occurs. These emails typically contain all the minimum information that SOC teams need to be able to respond to an event, including the time the event happened, what type of alarm was triggered, where it happened, and additional details about the alarm.

AUDIO

Connecting to remote intercoms or loudspeakers is a common task for operators in a command center. These can be used to communicate with an employee trying to gain access to a facility after hours, or to ward off trespassers with an audio command. The SIP protocol is the industry standard for IP telephony and is commonly adopted in both stand-alone intercoms/speakers and cameras that include audio capabilities.

By their very nature, standards-based integrations are limited in the types of functionality they support. More advanced and proprietary features of a system cannot be supported through common protocols. It's important to revisit your checklist and user story before deciding the value of these advanced features for front-line teams responding to real-time security incidents.



What is ONVIF?

ONVIF is an open industry group that develops standardized interfaces for the interoperability of IP-based physical security products. ONVIF provides various specifications for different IP security systems—ranging from cameras, to access control and analytics. At SureView, we support the most widely adopted specification, Profile S, for video streaming and configuration. Most IP camera systems support this specification, allowing users to connect and stream live video from a camera straight to SureView. It also allows for system discovery on a network, simplifying the onboarding process by finding your cameras and populating them directly into your SureView account.

What is RTSP?

The Real-Time Streaming Protocol (RTSP) is a very well-established protocol used for streaming video over the internet. It was first established in 1996/97 and the vast majority of IP-based video and CCTV systems support this protocol. Like ONVIF this common protocol allows users to connect and stream live video from almost any IP camera to SureView.

What is SMTP?

Simple Mail Transfer Protocol (SMTP) is the standard for all email communication. The protocol is used by mail servers to send, receive, and/or relay outgoing mail between email senders and receivers. It was first developed in 1981 and has become an incredibly robust system of authentication, encryption, and message handling. It ensures messages are delivered to the right recipient across the globe—almost instantly. Nearly every security system in the world can send notification emails when events and alarms occur in their systems. SureView simply becomes an address to receive these emails, and then our system converts these messages into an alarm, so operators can respond immediately.

The beauty of email is that it contains a lot of valuable information—when it was sent, what device sent it, what triggered the alarm, additional details in the body of the alarm that might be relevant, and even attached images and video. Using this common messaging protocol eliminates complex alarm integrations and immediately allows users to realize extra value from their existing system.

What Is SIP?

The Session Initiation Protocol (SIP) is the common industry protocol for telephony over the internet, commonly known as Voice-Over-IP (VOIP). It was designed to be a general-purpose way to set up real-time multimedia sessions between groups of participants. SIP is very common in both VOIP telephone systems and also in security systems such as mass notification and intercoms. This common protocol allows users to conduct two-way talk downs to remote locations to provide security assistance and deliver an improved level of service.

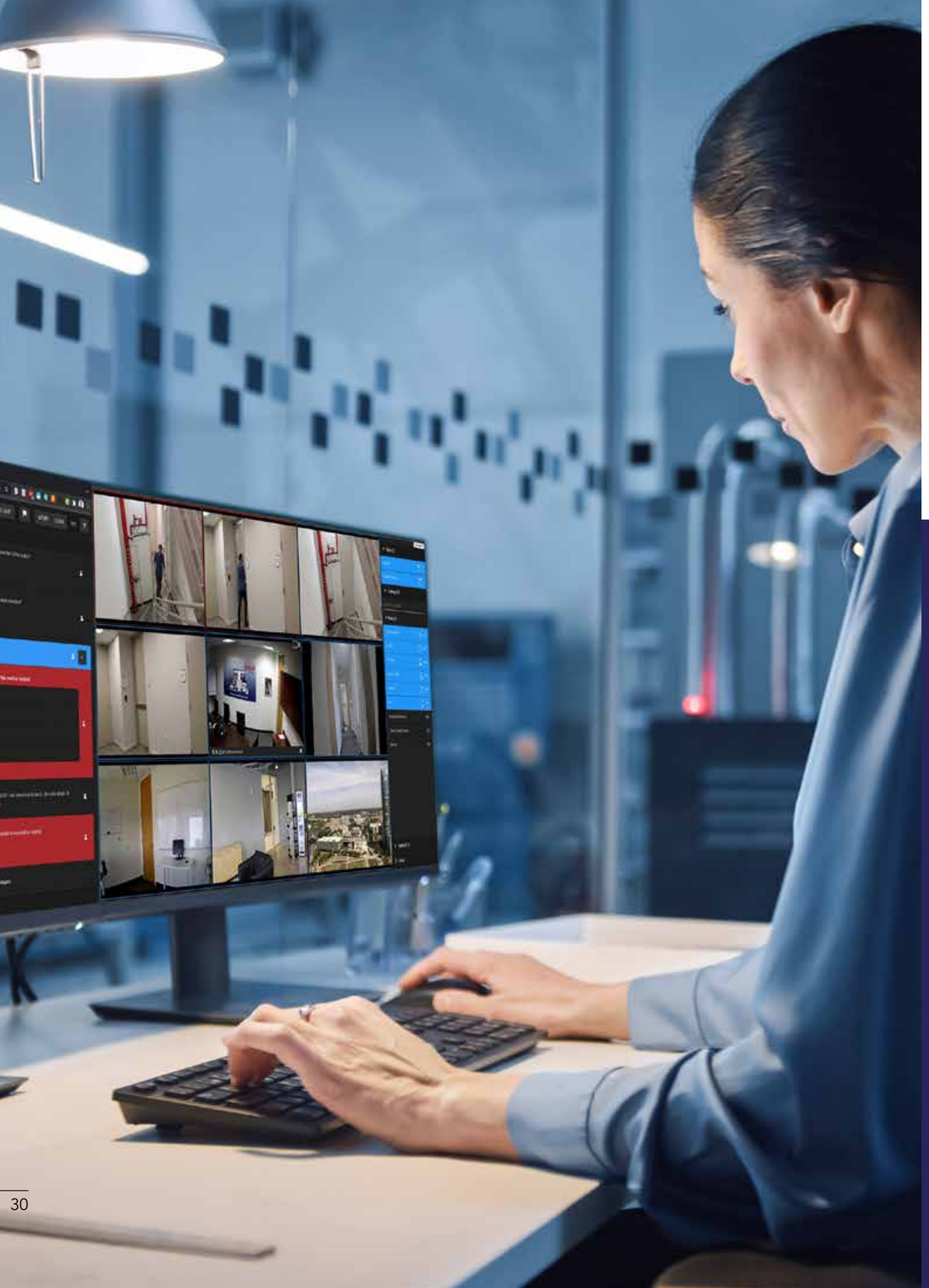
NATIVE INTEGRATIONS

INTEGRATION FOR LARGE GLOBAL DEPLOYMENTS

Native integrations utilize the published API (Application Programming Interface) or SDK (Software Development Kit) of the manufacturer to develop integration between 2 systems. These published API/SDKs provide a way for third-party developers to properly interact with their systems.

The benefit of utilizing a native integration is it often provides the ability to access a richer set of functions on the integrated system. At SureView, we have found that syncing data between two systems is one of the most powerful capabilities of native integrations. This is especially important in large organizations where there is constant change—new builds and upgrades mean that the cameras, doors, and security devices are constantly being added or changed. Managing these changes is time-consuming and can be fraught with human error. Automating these processes through integration delivers a large number of operational benefits.

The downside to native integrations is that there are no defined standards or approaches to developing or maintaining APIs or SDKs. It is entirely dependent on the openness and commitment of the manufacturer. As such, some API/SDKs are well architected, documented, maintained, and supported, while others are not. A common way to address this is a certification process, whereby the manufacturer certifies an integration with the third-party system. This provides customers with confidence that both manufacturers have a relationship and have jointly reviewed the integration to ensure it meets their collective best practices. Not every manufacturer has a certification process, so check with both vendors to establish the status and supported features of the integration.



NATIVE INTEGRATION—LOOK FOR A MODERN API

As mentioned above the native integration can use an API or SDK. But what's the difference?

An API is an acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other.

*"*Think of an API like a menu in a restaurant. The menu provides a list of dishes you can order, along with a description of each dish. When you specify what menu items you want, the restaurant's kitchen does the work and provides you with the finished dishes. You don't know exactly how the restaurant prepares that food, and you don't really need to.*

Similarly, an API lists a bunch of operations that developers can use, along with a description of what they do. The developer doesn't necessarily need to know how, for example, an operating system builds and presents a "Save As" dialog box. They just need to know that it's available for use in their app.

*Sourced from, How-To-Geek: What is an API, and how do developers use them?

COMMON CHARACTERISTICS OF APIs

Rather than just a generic connectivity interface, today's modern APIs have some common characteristics that make their interaction development simpler, faster, and more reliable. Similar to standards-based protocols...

- Modern APIs adhere to standards (typically HTTP and REST), that are developer-friendly, easily accessible, and broadly understood

- They are treated more like products than code. They are designed for consumption for specific audiences (e.g., security integrations, mobile developers), they are documented, and they are versioned in a way that users can have certain expectations of their maintenance and lifecycle

- Because they are much more standardized, they have a much stronger discipline for security (often are part of third-party software penetration testing) and governance

- As they are an integral part of the product they are closely monitored and managed for performance and scale

An SDK (Software Development Kit) is a set of software tools and programs used by developers to create applications for specific platforms. SDK tools will include a range of things, including libraries, documentation, code samples, processes, and guides that developers can use and integrate into their apps. SDKs are designed to be used for specific platforms or programming languages.

Although most SDKs often include an API as part of the “devkit”, SDKs don’t have the same common characteristics as modern APIs. This lack of standardization means that developing integrations with a manufacturer’s SDK may be more complex as developers interact with different libraries, functions, and programming languages.

QUESTIONS TO ASK THE MANUFACTURER WHEN NATIVE INTEGRATION IS REQUIRED:

• Do they offer a SaaS? If they do, this is a very good indicator that they use a modern API.

• Does the vendor already have integration in their library?

• Does the manufacturer certify integration?

• Is extra licensing required to utilize the manufacturer's API?

• Are there additional requirements necessary to support deployment architecture?

DEPLOYING A NEW INTEGRATION—SET UP A TEST ENVIRONMENT

The best advice for any security technology team is to have a good test environment in which to try out new integrations and update existing ones. This is the time to revisit your initial user story and run some basic tests to ensure that the items in the user story are addressed. Then bring in a representative of the operations team for User Acceptance Testing (UAT) to establish that the integration delivers their needs. Once you are confident the functionality looks good, for new integrations, we recommend staged rollouts. These provide an opportunity to check for load and performance, before deploying system-wide. Having a pragmatic and simple process for deploying new integrations will pay dividends in the long-term.

sureview

sureviewsystems.com

Florida Office

400 N Tampa St
Suite #1750
Tampa, FL 33602

Phone
+1 (888) 387.2860

California Office

101 Jefferson Drive
Menlo Park, CA 94025

Phone
+1 (888) 387.2860

UK Office

Hawthorne House,
Tawe Business Village,
Phoenix Way,
Enterprise Park,
Swansea, SA79LA, UK

Phone
+44 (0) 1792 278 110

WHITE PAPER

Integrating your Security Operation

A ROADMAP FOR CONNECTING TECHNOLOGY TO DELIVER OPERATIONAL VALUE

sureview

sureviewsystems.com

Florida Office

400 N Tampa St
Suite #1750
Tampa, FL 33602

Phone
+1 (888) 387.2860

California Office

101 Jefferson Drive
Menlo Park, CA, 94025

Phone
+1 (888) 387.2860

UK Office

Hawthorne House,
Tawe Business Village,
Phoenix Way,
Enterprise Park,
Swansea, SA79LA, UK

Phone
+44 (0) 1792 278 110

sureview