White Paper

OpenDXL Idea Book

Fifteen Easy Ways to Integrate, Orchestrate, and Expedite Security Operations

intel Security

Table of Contents

Overview . . . 3
What Is OpenDXL? . . . 4
What Makes OpenDXL Unique? . . . 4
How Does It Work? . . . 4
OpenDXL Use Cases: A World of Possibility . . . 4
- Master example: Six products, four vendors, total automation . . . 5
- Use Case 1: Extracting more value from existing security investments . . . 6
- Use Case 2: Integration with a SIEM leveraging a TAXII bus . . . 8

Intel Security Innovation Alliance Integration . . . 9
- Use Case 3: Broadcasting TrapX honeypot data to McAfee Advanced Threat Defense and other subscribers . . . 9
- Use Case 4: Using McAfee ePO automation to act on high-confidence detection information from other security products . . . 10

And 10 More Ideas . . . 11
Summary . . . 12
APPENDIX . . . 12
- Build Your Own OpenDXL Integration . . . 12
- Example of an OpenDXL Integration Scenario . . . 13
- Reality Check: OpenDXL versus Other Integration Models . . . 14

Overview

The rapid pace of attacks and shortage of expert security talent have driven the security industry to embrace the notion of pooling data resources and orchestrating actions across vendors, open source projects, and internal development efforts. Much like a neighborhood watch, sharing threat information and codifying procedures will make the community as a whole—vendors, industry leaders, and enterprises—stronger and better able to fend off advanced, evolving threats with greater speed and accuracy.

Goals of the OpenDXL Initiative

- Increase integration ease and flexibility.
- Provide a simple set of tools to speed development.
- Improve security operations' ability to create cross-product, cross-vendor security automation.
- Protect organizations' integration efforts by immunizing them from API changes by product vendors.

Better intelligence also improves security detection and response. And, beyond sharing data, multiple systems need to collaborate to create a concerted response, from prioritization to containment to remediation, along with an improvement in security controls and protections.

While there are clear advantages to sharing threat data and organizing heterogeneous applications and security operations into systems, there are also some significant challenges:

- Security and IT infrastructures, which have been built up over many years from disparate technologies, vendors, and in-house applications, are complex, with fragile connections.
- At times, integrations between products from different vendors don't do exactly what an enterprise has in mind, or the organization's needs change or expand over time and vendor offerings don't keep up.
- Point-to-point, API-led product integrations are time-consuming to build and difficult to maintain as you upgrade product versions and data formats.
- To integrate security products, vendors have to go through a process of negotiation, agreement, and implementation, which takes a great deal of time and leaves enterprises at the mercy of vendor priorities and market forces.
- Traditional REST APIs do not efficiently support real-time notifications. The traditional approach is to implement polling and scheduled data publishing models that incur processing overhead and a time delay in responding to critical events.
- Without easy access to data—such as context or emerging threat intelligence—or a simple way to invoke another service, applications can't apply the insights that could make them more effective against emerging threats.

In response to this set of hurdles, Intel Security developed the McAfee® Data Exchange Layer (DXL), which overcomes much of the complexity of integration and enables high-speed communication between applications. Over the past several years, this technology matured as it was being used to connect Intel Security products and Intel® Security Innovation Alliance partner products.

Having proven that the software is ready, Intel Security launched the OpenDXL initiative to put the power of integration and automation in your hands by providing open source tools, expertise, and a supportive community. Now, any application, whether homegrown or vendor supplied, can tap into the real-time capabilities of the DXL communications fabric, extending the reach of every other integrated system. The goals of the OpenDXL initiative are to increase integration ease and flexibility; to provide a simple set of tools to speed development; and to offer creative opportunities for developers in order to improve security operations by leveraging applications from open source, in-house developers, and the commercial development community.

Seven Key Advantages of OpenDXL

OpenDXL provides seven key values that make it the protocol of choice for security messaging:

- Real-time exchange of security-relevant data enables automated orchestration of cross-vendor security defenses.
- Multiple messaging patterns for comprehensive, integrated security, including publish/subscribe for one-to-many messaging and request/response for one-to-one messaging with a service.
- Services simplify access and expand availability of security and management technologies to all connected clients.
- Support for diverse use cases enables a single OpenDXL fabric to handle all of a company's security messaging requirements with no need for multiple solutions.
- Enhanced security built into DXL addresses one of the top gaps reported for commonly used transport protocols.
- The integration abstraction layer protects developers' investments by providing a single integration API across supported protocols, so integrations stay active longer and require less maintenance due to API changes
- Superior manageability through the McAfee ePO platform.

What Is OpenDXL?

Open Data Exchange Layer (OpenDXL) allows developers and scripters to connect products throughout an organization's security infrastructure and accelerate the threat defense lifecycle. With access to data and triggers from multiple sources delivered in milliseconds over the McAfee DXL communications fabric, everyone benefits from more precise analysis and orchestration of security actions with IT systems. OpenDXL also provides opportunities for more rapid and flexible security infrastructure. Enterprises have a way to achieve integrations between competing products as well as niche applications and open source projects, connections that would typically not occur under standard market conditions.

What Makes OpenDXL Unique?

Commonly, security teams and vendors integrate applications through slow and tedious one-to-one integrations, manual scripts, and scheduled processes. With OpenDXL, these processes become a thing of the past. Instead of a series of individual integrations, all components connect to the DXL, a fast and simple messaging bus that allows for bi-directional sharing with other connected systems. Best of all, you only have to integrate once to the DXL fabric to obtain access to all of the applications that are using the fabric. Another key advantage of OpenDXL integration compared to other methods is high-speed sharing of vital security events. This improves time to detection, containment, and remediation—the key performance metrics for security operations.

How Does It Work?

Applications publish and subscribe to message topics or make calls to DXL services in a request/response invocation, which is similar to the RESTful APIs typically used for developing web services. The OpenDXL fabric delivers the messages and calls immediately, connecting an enterprise's security, IT, and in-house solutions into a coordinated and orchestrated ecosystem. If there are changes to the publishing or receiving applications, the DXL abstraction layer insulates the rest of the deployment from the change, reducing risk and the cost of integration maintenance.

OpenDXL Use Cases: A World of Possibility

There are already numerous DXL integrations developed by Intel Security, our partners, and our customers. Some data types applications publish over DXL today include:

- Deception threat events.
- File and application reputation changes.
- Mobile devices and assets discovered.
- Network and user behavior changes.
- High-fidelity alerts.

Vulnerability and indicator of compromise (IoC) data.

We hope that the open source tools available on GitHub.com/opendxl and the use cases presented below will inspire more developers and organizations to come up with new applications for this powerful and versatile technology.

To give you an idea of the types of integrations made possible by OpenDXL, we will walk through a detailed integration. Then we will explore more use cases that apply these concepts and others to solve various problems.

Master example: Six products, four vendors, total automation

In November 2016, we demonstrated an OpenDXL orchestration integration sharing intelligence across multiple vendors' products, then coordinating responses and policy enforcements. (Click here to view the demo.) Thanks to automation capabilities, the entire process described below occurs in just seconds. The basic idea is that the firewall's detection of a single command and control communication leads to several simultaneous containment and remediation actions that affect the original host (whose outbound communication triggered the alert), as well as other endpoints.

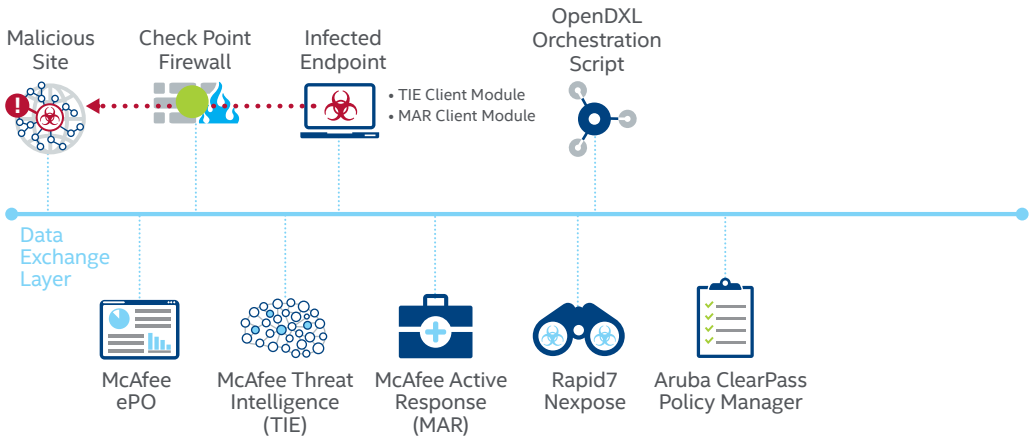

Figure 1. Orchestration with OpenDXL.

- **Step 1: Detection**—Malware is launched on an infected endpoint, connecting to a known Command and Control Server identified by Check Point Anti-Bot. Check Point's firewall notes the traffic to a known malicious command and control server. Check Point's firewall publishes an OpenDXL event.
- **Step 2: Identification**—The OpenDXL orchestration script has been listening over DXL. It receives the event, and then queries McAfee Active Response to identify, from the source system, which executable made the outbound connection. This allows us to move from just knowing network information (source, destination, port, and more) to file information. And, since McAfee Active Response maintains a historical record of network connections, this will succeed even if the process is no longer running. This provides additional data for Step 4: Scoping.
- **Step 3: Containment**—The OpenDXL orchestration script then sets the reputation of the file it has identified to "known malicious" in McAfee Threat Intelligence Exchange. McAfee Threat Intelligence Exchange then broadcasts this new information over DXL, directing McAfee Threat Intelligence Exchange clients to kill the malware processes, thus disconnecting the communications. Note that this will apply to any system running the convicted process.
- **Step 4: Scoping**—Although not shown in the demo, McAfee Active Response can be leveraged to find other processes that connected to the same command and control server or other dormant instances of the files that either have not been executed or are no longer running and then remove them.
- **Step 5: Tagging**—To complete the cleanup, the systems that McAfee® Active Response has identified as containing malware are tagged in McAfee ePO software by the orchestration script as a hook to allow further activity by the administrator.

- **Step 6: Assessment**—The orchestration script triggers a vulnerability scan reaction by Rapid7 Nexpose as a final step to identify all the weaknesses in the compromised systems that could also be exploited.
- **Step 7: Remediation**—The orchestration script sends a request to the Aruba ClearPass OpenDXL service to update attributes for systems exposed to malware, which triggers policy enforcement, which can quarantine the device from the network while detailed remediation occurs.

This example results in several positive outcomes:

- **More rapid integration of multiple products from different vendors:** Lightweight Python clients and the one-to-many OpenDXL service wrappers reduce the burden on in-house developers.
- **Richer threat intelligence from multiple sources:** The integration increases the ability of all products in a multivendor environment to detect, protect, and correct faster and more accurately.
- **Dramatic improvement in operational efficiency of the security infrastructure:** This results in stronger ROI on investment in security tools.

Behind the Scenes
To create this demo, the team used existing APIs to build simple connectors or "service wrappers," to use OpenDXL terminology, for each of the third-party applications, Check Point, Rapid 7, and HP Aruba, as well as three Intel Security products: McAfee Threat Intelligence Exchange, McAfee Active Response, and McAfee ePO software. Then, the team scripted these connectors together into a logic-driven flow that ordered the appropriate action based on the data being sent over DXL.

Service wrappers are used to enable integration of data and processes across multiple tools by exposing existing APIs as DXL services. They provide a lightweight alternative to native Python, C++, or Java integrations. These service wrappers enable non-DXL-integrated APIs of Rapid 7 and HP Aruba functionality to be called via DXL. The Python clients simplify the process of querying endpoints in the enterprise environment, reducing the integration effort from 120 lines of code (with pure OpenDXL) to 20 lines of code (with the McAfee Active Response open source Python client).

We have published these service wrappers on GitHub.com/opendxl for use as building blocks for your integrations. Just as we took advantage of simple McAfee Active Response searches without having to focus on lower-level details, such as McAfee Active Response-specific DXL topics and message formats, other applications can do so as well. Several of the examples below illustrate the power of integrating with McAfee Threat Intelligence Exchange reputation services and the McAfee ePO management APIs to simplify access to endpoint policy and enforcements.

Let's look at other examples that use these and other DXL-based software elements to expedite security operations.

Use Case 1: Extracting more value from existing security investments
Description: A major insurance company with 70,000 endpoints located in various locations across the global was looking to derive greater value from their Palo Alto Networks firewall and Proofpoint email gateways. At the time, these solutions were compartmentalized—they had little or no interaction with the organization's Intel Security endpoint and data defenses. Prior to the release of OpenDXL, the organization was unable to find a well-supported, proven, and widely adopted standard that was able to measurably improve its risk profile out of the box.

Committed to maturing its security infrastructure and evolving from reactive, firefighting mode to a big-picture, proactive approach, the company wanted to leverage threat intelligence and detections generated by their network and email gateway products and distribute this data to the rest of their security toolsets. An immediate desire was to have their endpoints react to this threat intelligence in order to improve their protective and investigative capabilities. The endpoints were protected by Intel Security endpoint and data loss prevention technologies and by McAfee Active Response. The OpenDXL announcement prompted the company to adopt McAfee Threat Intelligence Exchange.

Technical approach: The team of in-house developers were SOC analysts who were competent in coding, but not typically engaged in this activity on a day-to-day basis. They were familiar with the simple-to-code Python language. The company used the OpenDXL Python client and adapted the "openweather_service_wrapper.py" from OpenDXL.com/Github to create a connection to RESTful APIs for the Palo Alto Networks firewalls and Proofpoint TAP. For the connections to Intel Security products, they used the OpenDXL McAfee Threat Intelligence Exchange "file_rep_sample.py." These open source OpenDXL scripts required only minor modifications. This enabled fast implementation with a small amount of effort. An in-house study revealed that development time was less than 40 person hours.

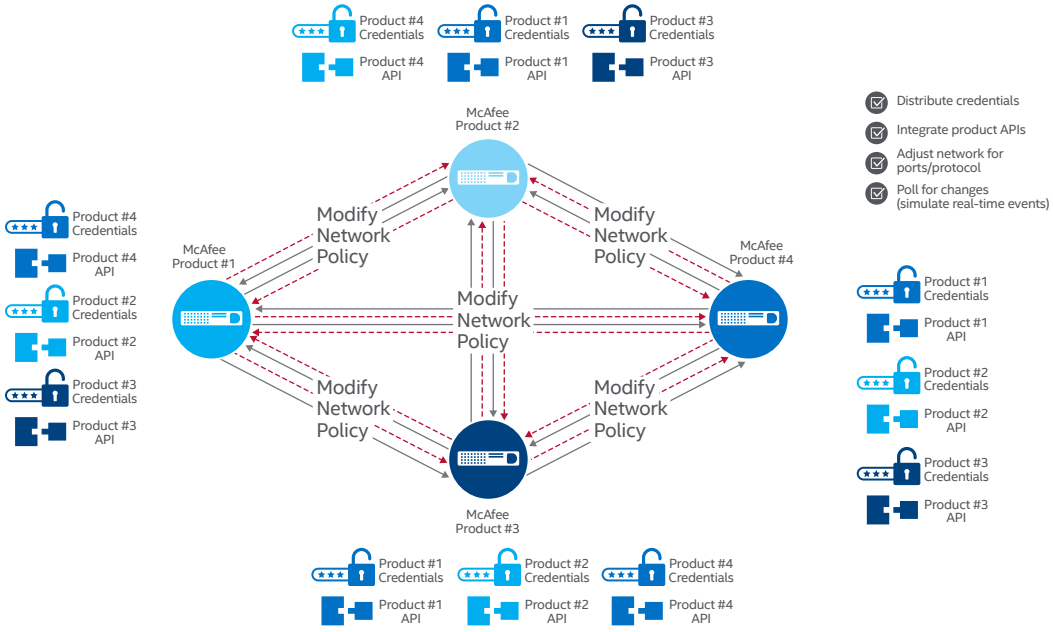

Figure 2. Pre-DXL security infrastructure (complex).

Technical Note

To consume threat information, the development team reused the file_rep_sample.py almost out the box, updating only the infrastructure details from their environment. Developers were quickly able to build composite enterprise reputations fed by various security devices (such as next-generation firewalls) in the McAfee Threat Intelligence Exchange database that the rest of the environment could immediately react on. For publishing threat information, they used the openweather_service_wrapper.py and updated it for every threat information source (next-generation firewall, email gateways, and others) with the relevant infrastructure details. The RESTful APIs from those devices wrapped the IoCs that were generated and subsequently parsed for relevant details (destination IP address, process SHA1 hash, and more to be published on the OpenDXL bus). See the DXL Open Topic Threat Event Format Specification, which details how we name topics and which fields we recommend in the exchange process.

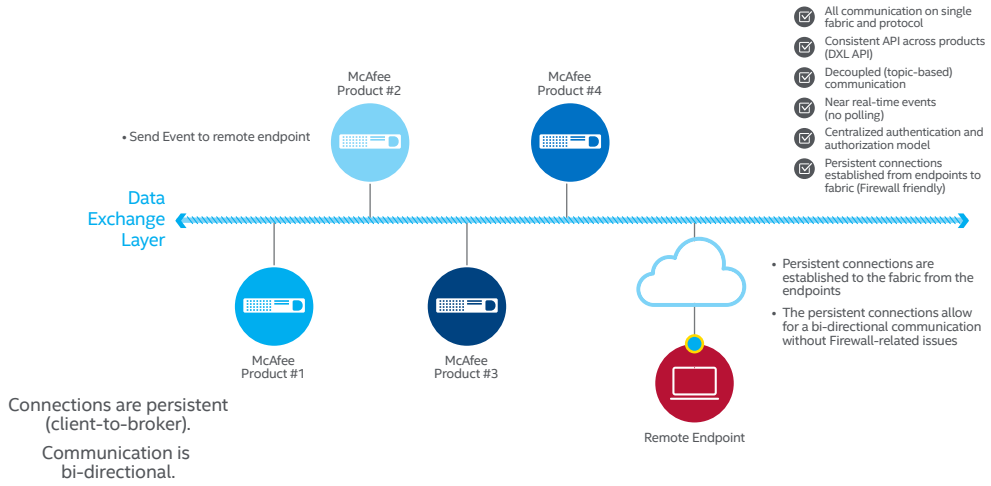

Figure 3. Security infrastructure with DXL integration (efficient).

Outcomes:

- Threat data gathered by the Palo Alto next-generation firewalls, as well as the intelligence collected by the sophisticated capabilities of Proofpoint with TAP, were passed on to endpoints, allowing McAfee Endpoint Security to do what it natively does (based on established policy)—namely, block, clean, or delete. This has resulted in more robust protection against a greater number of threats on devices that users interact with on a daily basis. Value already bought and paid for by the enterprise, while formerly only partially leveraged, now extends all the way down to every endpoint in the environment.
- The OpenDXL integration helps connect systems and threat data to more effectively and speedily block phishing and spear-phishing threats, along with malicious web downloads—all of which typically employ social engineering tactics to attack users, who are the weakest link in the security chain.
- The resource-constrained security department is experiencing increased efficiencies in workflows and the infrastructure.
- OpenDXL has maximized the return from the company's security investment and has acted as a force multiplier for the solutions in its multiple-vendor environment.

Use Case 2: Integration with a SIEM leveraging a TAXII bus

Description: Another large insurance company with approximately 50,000 globally deployed, Intel Security-protected endpoints also initiated an OpenDXL integration with the IBM QRadar SIEM solution, which employs STIX and TAXII to share threat intelligence information across this multivendor security infrastructure. The integration, modeled after existing closed loop automations of Intel Security products, enables sharing of indicators of compromise (IoCs) received from various sources, including other OpenDXL-integrated sources (next-generation firewalls, non-Intel Security intrusion prevention systems, and FireEye NX devices), as well as subscribed threat feeds.

Now, via a TAXII bus integration, this intelligence can be shared with IBM QRadar SIEM. When IBM QRadar SIEM receives the information, it treats those IoCs as "observables" and, following its policy, automatically does a historical search to determine whether anyone had come in contact with the sample prior to the moment in time when the sample was identified as malicious.

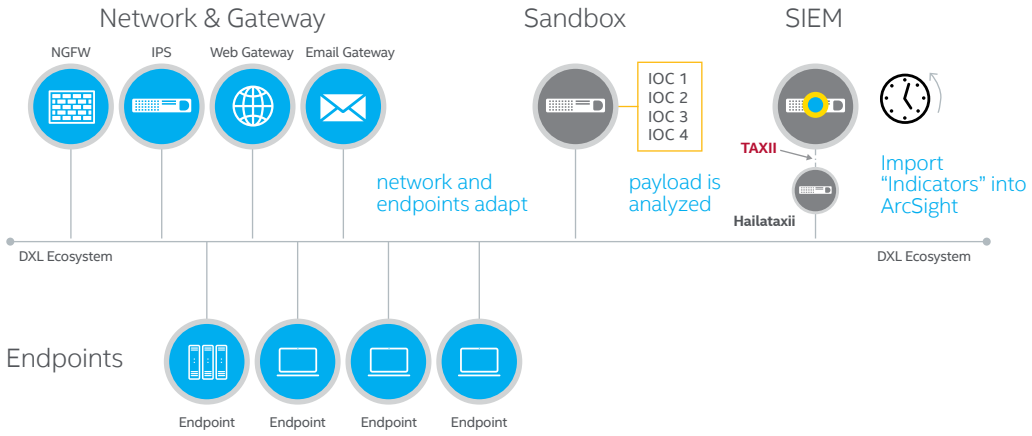

Figure 4. Integration with a non-Intel Security SIEM leveraging a TAXII bus.

Technical implementation and components: The development environment consisted of the OpenDXL Python client on a Linux system. An additional step involved the company's existing TAXII server, which has a RESTful API. The development team used the Python client to wrap the IoCs from McAfee Threat Intelligence Exchange in the STIX format, and the TAXII server then shares them with IBM QRadar SIEM.

Outcomes:

- Enables successful sharing of threat intelligence in STIX format over OpenDXL and TAXII environments.
- Examines the environment for dormant threats that have escaped detection.
- Ramps up the efficiency of the security infrastructure and security operations several notches by allowing the IBM QRadar SIEM to receive and act on to the rich data generated by varying threat/detection sources, including McAfee Threat Intelligence Exchange.

Intel Security Innovation Alliance Integration

The next use case illustrates how independent software vendors in the Security Innovation Alliance program have created new capabilities using DXL. For complete and up-to-date descriptions of Intel Security Innovation Alliance partners that have fully integrated McAfee DXL products, visit the partner directory.

Use Case 3: Broadcasting TrapX honeypot data to McAfee Advanced Threat Defense and other subscribers

Description: TrapX is an Intel Security Innovation Alliance partner that specializes in deception-based cybersecurity defense—emulating enterprise systems to lure attacks and reveal tactics and intent. The objective of this integration was to broadcast the threat information gathered and analyzed by the TrapX DeceptionGrid solution across both Intel Security solutions and endpoints deploying other partner solutions, such as Rapid 7, Forcepoint, Avecto, and more. The existing integration between McAfee Advanced Threat Defense and McAfee Threat Intelligence Exchange set the stage for the OpenDXL integration.

The moment attackers come in contact with a DeceptionGrid emulation, an alert is triggered, and the injected malware is sent to McAfee Advanced Threat Defense, which swiftly and accurately analyzes the threat and updates its file reputation. Working with McAfee Threat Intelligence Exchange, McAfee Advanced Threat Defense broadcasts the data over DXL. The threat intelligence is then incorporated into enforcement processes across other Intel Security and third-party solutions. Through OpenDXL, TrapX achieves a one-to-many integration, enabling distribution of the data across a diverse security environment.

Technical implementation and components: The TrapX integration with McAfee Advanced Threat Defense was accomplished through the use of the McAfee DXL SDK, written in C++.

Outcomes:

- Broadcasting the results of this type of advanced deception technology enables IT security to find advanced threats that may bypass other security solutions.
- Time-to-detection is significantly reduced.
- Provides near real-time reputation information on new files that enter the environment. Through OpenDXL, the enterprise environment is constantly updated, enabling disparate security products to collaborate as they detect, protect, and correct with greater speed and accuracy.

Use Case 4: Using McAfee ePO automation to act on high-confidence detection information from other security products

Description: The new wrapper for McAfee ePO management APIs provides easy, fast options to use the industry-leading management console to apply policies, tag systems, move groups, and trigger other actions within McAfee ePO software. These capabilities are the most frequently used and valued integrations available within McAfee ePO management APIs and permit more applications to leverage centralized and efficient management. Possible sources of high-confidence detection information could include products from other vendors, open source applications, cloud services, in-house capabilities, and more. For example, McAfee ePO software could be used to more efficiently manage risk by tagging a compromised system and adjusting risk posture based on the incoming information, such as adding more restrictive data loss prevention policies or quarantining.

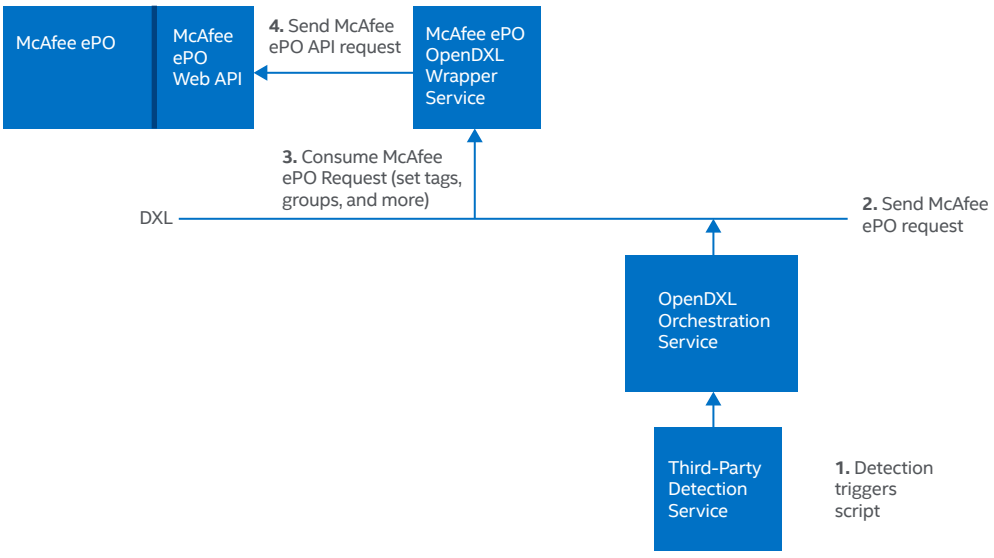

Figure 5. Non-DXL integrated third-party product.

Technical implementation and components: The external software detects a security event and triggers the automation script with a set of remediation actions. There is no registration required to receive a triggering DXL event in this case, as remediation actions are generally activated via query/response to a service. The script then sets into motion the necessary McAfee ePO actions via the OpenDXL fabric. The McAfee ePO service wrapper enables McAfee ePO software to securely execute the necessary response via OpenDXL.

One such scenario is integration with Cuckoo, an open source automated malware analysis tool (similar to McAfee Advanced Threat Defense). By leveraging the Cuckoo API to gather results from various file analyses, an OpenDXL script can be used to trigger an action in McAfee ePO software. One example would be to apply a tag to that system and then apply restrictive security endpoint policies via McAfee ePO tag-based policies.

Outcomes:

- Increases the value of high-confidence detection information and distributes it across the entire infrastructure.
- Automatically triggers a response or policy change from McAfee ePO software to protect systems better and faster.
- Enables disparate security products to take advantage of the efficiencies of the McAfee ePO management console.

And 10 More Ideas

In addition to the above case studies, the OpenDXL community is actively pursuing a variety of other integrations. Here are just some of the ideas that may already have been published on github.com/opendxl.

- An IP reputation service to publish newly identified malicious or spammy IP addresses.
- A wrapper for intrusion prevention tools (including Snort) to receive and respond to OpenDXL events, taking action such as disconnecting communications, blocking an IP address, or quarantining a host or workgroup.
- DXL listeners for any countermeasure, varying the action to the enforcement available in the countermeasure (delete file, blacklist domain/IP address, set watchlist, escalate monitoring, or force scans for compliance, vulnerability, or malware). The master example shows some of these. You can map to the functionality of your existing software.
- A connector to open up SNMP and network management software to receive and publish and call and respond to DXL events, simplifying response to network-oriented threat data, such as command and control traffic.
- Send memory dumps to a sandbox for process analysis and IoC identification.
- Convert a threat intelligence feed (public, private, government, or industry) to a DXL topic to enrich threat analysis.
- Receive alerts via your Twitter account on DXL topics, which is a faster way of getting this information than through email or other methods. Additionally, private Twitter accounts can be used to share intelligence.
- A DXL connector for the open source Postfix message transfer agent to enable Postfix to block spam IP addresses and spam servers (needs other apps to publish the events)
- Sensor/equipment connector to pass environmental and state data (such as equipment temperature) to DXL and send alert when thresholds are reached, minimizing downtime and sensor wear or keeping devices within optimal ranges.
- Use DXL to create an alert in the SOC or other monitoring center when there is an urgent alarm, turning digital alerts (such as from SIEM) into audible or visual alerts to ensure human action.

Summary

The foregoing use cases—all with varying levels of complexity—are an overview of the creative development opportunities afforded by OpenDXL. We hope these have inspired you to start your own OpenDXL project. Bookmark and explore these pages for core resources.

- Build Your Own OpenDXL Automation: A PDF tutorial anyone can download.
- OpenDXL Python SDK: is located on GitHub, with software and tutorials for many different integrations. Customers can also access the SDK with their grant number from the Intel Security download site.
- C++ and Java DXL SDKs are available for Intel Security Innovation Alliance partners through the partner portal.

For more information, visit mcafee.com/dxl.

About Intel Security

Intel Security, with its McAfee product line, is dedicated to making the digital world safer and more secure for everyone. www.intelsecurity.com. Intel Security is a division of Intel.

APPENDIX

Build Your Own OpenDXL Integration

The use cases in this guide demonstrate how your security solutions can benefit from sharing of data and integration of actions within your infrastructure. Now, let's look at the approach to take. To plan your integration, you'll need two components:

"Ingredients": You create these by ensuring that security applications you'd like to leverage are appropriately enabled for OpenDXL. These ingredients are the things that can publish or subscribe to DXL content or that you can call or tell to take action over DXL as part of a script.

Native OpenDXL Python integrations: These lightweight scripts let you extend source code for your applications and open source applications to let them publish to, subscribe to, and communicate via the DXL fabric. On GitHub, you will find examples such as the weather station integration.

Application programming interface (API) service wrappers: You build an OpenDXL script to wrap an application API and expose it as a DXL service on a DXL fabric. This type of integration does not require access to the application's source code and can be used to open up key features of an application quickly. GitHub examples include our wrappers for McAfee Threat Intelligence Exchange, McAfee Active Response, and McAfee ePO software.

Native DXL C++ and Java integrations: These integrate your source code with Intel Security source code and require Intel Security Innovation Alliance partnership. Examples include most of the Intel Security Innovation Alliance partners listed here.

"Recipe": Next, you'll need to create the orchestration script—that is the recipe—to send commands to your ingredients via OpenDXL, expediting your desired outcome. The script, which can be launched by another Python or OpenDXL script or OpenDXL connector, triggers one or more actions by your ingredients. This script may be run on demand (manually), or automatically in response to a specific OpenDXL event. It delivers security automation in the form of "if-then" logic.

Creating the Ingredients

To create the ingredients, you need to determine whether the desired product functionality is either natively exposed via OpenDXL or is accessible via an existing, non-OpenDXL integrated API. If the product provides the desired functionality via a native DXL integration, then you're ready to begin. If the product provides the desired functionality via a non-OpenDXL API, then you need to download or create an API service wrapper for the product. The API service wrapper exposes the desired functionality as an OpenDXL service. After you take care of this step, you're ready to begin.

In the event the product does not provide the desired functionality, either via OpenDXL or an API, you cannot automate a particular product's functionality via OpenDXL. At this point, you may want to request that the vendor natively support OpenDXL to expose the desired functionality in a future release.

Creating the Recipe

Now that you have prepared the ingredients, you're ready to create the recipe for the integration. To do so, you need to create an OpenDXL script that can either be invoked on demand or automatically in response to a specific OpenDXL message.

Example of an OpenDXL Integration Scenario

To give you an idea of how to go about an OpenDXL integration, let's explore an example where we create security automation that leverages two common security products: a firewall and a network access control (NAC) solution. Our objective is to use security automation to command the firewall to block all connections to a device inside our network at the IP address we supply. The NAC solution would then isolate the system on a remediation network with limited access to internal resources.

Start by creating the ingredients:

Check the firewall and confirm it has a native OpenDXL integration that does exactly what we're looking for. In this scenario, we discover that it does indeed provide an OpenDXL service that supports requests sent to the topic "Company.Firewall.BlockIP" with an IP address parameter.

Check the NAC solution. In this case, we discover that the NAC solution does not natively support OpenDXL, but it does have an existing API that provides the desired functionality. There is an API method, "TransferSystem," that takes an IP address (of the device that needs to be moved) and a network identifier (as the target network to where the system needs to be moved) as parameters.

Check the OpenDXL community to see if there is an API service wrapper available. We find that there is none, so this becomes our first OpenDXL scripting task. We need to create an API service wrapper that exposes the "TransferSystem" API method as a DXL service topic. We check the OpenDXL SDK and create the API service wrapper.

Now we're ready to move on to creating the recipe, that is, building the automation script. First, we check the OpenDXL SDK for information on how to build an automation script, and then we copy the template.

In the automation script, we invoke the two Open DXL services by sending two requests. The first request is sent to the "Company.Firewall.BlockIP" topic to instruct the firewall to block communication to/from the specified IP. The second request is sent to the "TransferSystem" topic we registered in our API service wrapper, which, in turn, calls the NAC solution API to isolate the system at the specified IP on the specified remediation network.

We want to trigger our automation script manually on demand to ensure we've successfully built the ingredients and recipe for our security automation use case.

Reality Check: OpenDXL versus Other Integration Models

While we think OpenDXL fills a critical need for the industry, it is not a panacea. OpenDXL is all about rapidly creating security automation that leverages disparate security tools from different vendors to achieve consistent, beneficial outcomes. Any product/component that has an API (DXL-connected or not) can be leveraged within this automation. OpenDXL is helpful in integrating components that support programmatic usage and/or programmatic integration but have not yet integrated with McAfee DXL. In this case, service wrappers can be used to connect these types of applications with OpenDXL.

The chart below summarizes possible integration scenarios, applications, and recommended tools, starting with the best uses of DXL.

Integration Type	Who Does the Work	Value Add	Recommended Tools
OpenDXL Python	Scripters	Easy and fast way to create an OpenDXL ingredient or automate ingredients in scripts	Open Source package of your choice OpenDXL Python Client
OpenDXL Service Wrapper	Scripters	Exposes existing APIs as DXL services for other recipes to easily use	Example code of github.com/opendxl
DXL C++ or Java	Security application owner (must be Intel Security Innovation Alliance partner)	Always-on connectivity to the messaging fabric	C++ and Java rather than Python for primary workflows, (example: McAfee Threat Intelligence Exchange server to its client)

McAfee. Part of Intel Security.
2821 Mission College Boulevard
Santa Clara, CA 95054
888 847 8766
www.intelsecurity.com