## Revision History

<table>
<thead>
<tr>
<th>Revision Version</th>
<th>Author</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Christopher Newman, Andrew Hurren</td>
<td>May 13, 2015</td>
<td>Initial release</td>
</tr>
<tr>
<td>B</td>
<td>Christopher Newman</td>
<td>June 23, 2015</td>
<td>Added Glossary and other updates</td>
</tr>
<tr>
<td>C</td>
<td>Ted McDonald</td>
<td>September 17, 2015</td>
<td>Updated Fabric Design information</td>
</tr>
<tr>
<td>D</td>
<td>Ted McDonald, Chris Smith</td>
<td>April 7, 2016</td>
<td>Updated information, added deployment examples</td>
</tr>
<tr>
<td>E</td>
<td>Julie Byrd</td>
<td>July 11, 2018</td>
<td>Updated to McAfee branding</td>
</tr>
</tbody>
</table>
**Introduction**

The purpose of this document is to provide recommendations and best practices for building a resilient and highly available Data Exchange Layer (DXL) communication fabric. The Data Exchange Layer enables an adaptive security ecosystem. It is a low footprint, near real-time, bi-directional communications fabric allowing connected security components to share relevant data between endpoint, network, and other security systems. It allows for automated response, greatly reduced response time, and better containment.

Traditionally communication to endpoints and between network products has been API driven and dependent on call backs or bidirectional access. As threats have grown more sophisticated these models have become increasingly untenable. Minutes, if not hours, from detection to reaction to containment is simply no longer acceptable. To deal with this, security architectures must evolve too. Shared threat information and synchronized real-time enforcement are becoming necessities, not luxuries. Until now, this has been seen only for specific products or single point-to-point integrations. The DXL supplies a simple solution to this real-time problem.

McAfee created a new standardized communication mechanism, as no existing tools were a good fit for the requirements. This mechanism does require new infrastructure, however, that infrastructure can support a multitude of solutions and use cases. Our philosophy: build it right from the beginning, plug in products as needed.

**Benefits**

The Data Exchange Layer has several advantages over traditional communication mechanisms:

- **Product integration simplicity** — A single API implementation allows integration with any product already part of the data exchange layer (e.g. no more point-to-point integration).
- **An Open framework** — The Data Exchange Layer was designed with openness to support the integration of any third-party product.
- **Scalability** — The Data Exchange Layer was designed to understands present and future needs (e.g. ubiquitous computing), scaling for an enterprise (millions of concurrently connected clients) with a low network footprint. It is also built to communicate with clients regardless of location (on network or off network).
- **Easy to deploy** — Supports multiple flexible deployment options starting with embedding the different Data Exchange Layer components with current and future solution sets. For example, customers deploying the McAfee Threat Intelligence Exchange DXL service are not required to deploy additional components to see the value of the Data Exchange Layer.
- **Secure** — Ensuring proper authentication, authorization, non-repudiation, confidentiality and other required security related features and capabilities.

Data Exchange Layer powers the evolution and delivers on the Security Connected vision. Customers benefit from significant operational cost savings, reduced complexity with unmatched operational effectiveness.
High Level Topology

This diagram provides an overview of a DXL topology with multiple instances of McAfee ePO, TIE services, DXL brokers, and DXL service zones. We will review several example topologies, including a complex environment such as the one shown below.

*LATAM and EMEA will have the same topology layout as APAC*
Glossary

**DXL** — Data Exchange Layer. The DXL is the communication protocol used for real-time communications.

**MQTT** — from [http://en.wikipedia.org/wiki/MQTT](http://en.wikipedia.org/wiki/MQTT): MQTT is a publish-subscribe based "light weight" messaging protocol for use on top of the TCP/IP protocol. It is designed for connections with remote locations where a "small code footprint" is required and/or network bandwidth is limited.

DXL was originally based on MQTT, but some MQTT features have been altered for use with DXL, such as additional security (TLS), access controls, and a few new methods.

**DXL Broker** — Clients connect to a DXL broker in order to get on the DXL. All DXL communications go through the DXL brokers. Think of them like DXL routers.

**Fabric** — A fabric is a connected network of DXL brokers used for connecting to the DXL and message relays.

**DXL Clients** — Any device that connects to the DXL that is not providing a service or routing messages.

**Service** — Services are the components that utilize the DXL for communications. Examples are things like TIE or Active Response. Different services can have different communication models and purposes. For example, TIE is a centralized clearing house for reputation information, whereas Active Response uses DXL for real-time communication with clients from the Active Response server.

**Topic** — Topics are like the URLs of the DXL. It is where a service publishes its specific methods. When a client connects to the DXL it notifies the broker of the topics it is interested in.

- Clients can subscribe to a topic to receive updates (one to many).
- Clients can perform a query on a topic (one to one).
- A server can publish on a topic.

**TIE Server** — A server that provides the TIE service.

**Hub** — Two DXL brokers combined into one virtual unit for HA and scalability reasons.

**Root Hub** — A hub that provides a communication link between all regions. This hub does not accept client connections and exists only to facilitate communication across the fabric. It is the universal parent.

**TIE client** — The Threat Intelligence client. Refers to the endpoint client that is managed via ePO. This is the component deployed to workstations and servers. The TIE client is both the enforcement point and decision maker when it comes to TIE rules. It parses information supplied by the TIE Server to alert, block, quarantine, etc.
**DXL Design Considerations**

The DXL provides a foundation transport for messages for many different McAfee and third-party solutions, and it is important to consider DXL fabric design when integrating products. At a small scale, DXL design is simple. Depending on the number of clients and the location of the clients/datacenters, a simple broker hub (two brokers connected to each other) may suffice. Once you start adding multiple products and services, as well as additional network security zones and geo-locations, further consideration must be taken to ensure a scalable and fault-tolerant architecture.

When designing a DXL fabric one must take into account:

1. Sizing/capability of DXL server components and DXL messages
2. The architecture of the underlying network, in particular choke points and client locations and Security Zoning
3. The Services to be provided and their unique needs
4. Availability and redundancy requirements

As different services take advantage of the DXL, the basic design of the DXL will become increasingly important. If you place brokers and hubs in sensible locations in advance, additional clients and services can be plugged in without additional work.

**DXL Brokers**

The basic building blocks of the DXL fabric are components known as DXL brokers. They combine the roles of a client terminator and a message router. All DXL clients connect directly to a broker and maintain a persistent connection while on the DXL. In addition, the brokers must connect to each other, discover, and maintain services via service zones. When a client requests a service, or when an update is broadcast, it is the broker's responsibility to relay these messages.

The number of brokers needed is determined by the geographic and network configuration of the customer. From a scalability standpoint, they will need, at minimum, one broker for every 50,000 end points.

If there are multiple ePO's or multiple datacenters to coordinate between two additional brokers, you will most likely need to construct a more robust DXL fabric, including multiple DXL brokers connected through a Root Hub.

If there are multiple regions or datacenters, additional brokers can always be added to service those areas, minimizing WAN communication and network latency.

**DXL Messages**

DXL messages come in two forms.

- **1-1** — This is a direct message on a specific service. Examples include a file reputation query or a database update.
- **1-Many** — This is a message broadcast to all interested parties. An example is a file reputation change or a request for all relevant clients to take an action.

**Topics**

DXL topics are like channels, or resource identifiers, associated with a DXL service. DXL topics are analogous to a URL that is associated with a resource supplied by a web server. They are named hierarchically and designed to be friendly with multiple providers. Clients subscribed to a topic receive notifications when messages for the topic are sent. At the same time, clients would use the topic for sending 1-1 queries. If there are multiple servers supplying a topic, the DXL brokers will round robin load or follow the service zones. An example would be a file reputation change from the TIE service. All clients 'subscribed’ to the topic would receive notifications when a file’s reputation changes.
DXL Clients

When clients connect to the DXL they instantiate a persistent connection to the closest broker and preserve this TCP/IP connection while they have network access. When a client receives the DXL client policy from ePO, it includes a certificate and key for authentication to the DXL and a list of brokers. The closest broker is then determined via ICMP hop counts. If ICMP is disabled in the environment, the client will simply round-robin the brokers, which may not be ideal. See the sections on Service Zones and Client Broker Connections for additional details about controlling DXL communication.

As the connection is client initiated, it is firewall and NAT friendly. Think of it like an SSL web connection. The client needs to be able to resolve and route to the web server, but the web server does not need to route directly back to the client. Any NATs and firewalls between the two are traversed as long as the destination is accessible. Once the connection is up, the broker can then direct messages back to the client through this already instantiated connection.

However, just because the clients can connect across the network (or off net) does not mean that it is advisable to just place a couple of brokers in the core, and consider your DXL fabric complete. Factors to consider when locating brokers include the network topology, latency, and scalability. It is recommended to locate brokers on the client side of firewalls, VPNs, WAN links, and other choke/enforcement points. If nothing else, permanent sessions through firewalls will consume resources if there are many of them (1000s-10,000s). In addition you can be more selective with your firewall rules if the broker is on the client side. If a broker is located on the client side, potentially thousands of connections can be compressed into one.

When you link DXL brokers together it is called a DXL fabric. DXL broker-to-broker traffic is directional and on the same port (8883) as client communications. Initially clients connect to the closest broker and supply a list of subscribed topics. Once communications are established, the broker is responsible for relaying messages back and forth from clients to services on the DXL.

Client Broker Connections

Client Broker Connections is a client-side ePO policy option that provides several options to further customize how DXL clients connect to the fabric. Clients can be restricted to connect only to a specific broker, hub, or branch of the fabric. Clients can also be configured to connect to specific branches of the fabric, and still retain the ability to connect elsewhere in the fabric if that branch is unavailable.

Enabling Client Broker Connections requires careful consideration before restricting how clients will communicate with the fabric. The default Client Broker Connections settings of no restrictions is the recommended configuration.

Services like TIE must have a policy that restricts them to one specific Hub in order to enforce the distribution of TIE services across the fabric.

End-points may want to be set up with a preference for a certain set of brokers to make sure they connect to those before any others.

For example, consider a user who primarily works in North America and their Client Broker Connections preference is set to only connect to North American brokers. If this user travels with their laptop to Asia, the DXL Client continues to connect to the North American brokers if they were visible. If the Client Broker Connection preference was not enabled, the DXL Client connects to the closest broker, allowing the user to travel and take advantage of any local DXL brokers.
DMZ Brokers

If you require endpoints that are outside the network to be able to communicate with the DXL fabric, then a broker must be established inside the DMZ to provide Internet-facing DXL connectivity. DMZ Brokers must have their publicly exposed System Name (Published System Name) and publicly exposed IP Address (Published IP Address) configured on the DXL Topology page. When both the Published System Name and Published IP Address have been set, all connections made to these brokers will use these values instead of the reported internal values. It is recommended that DMZ brokers not have any children in the topology unless it is intended that they connect through the external Published System Name and external Published IP Address.

No additional client configuration is necessary to take advantage of DMZ brokers. External endpoints will be unable to see any internal brokers aside from the DMZ brokers and will connect to the closest available DMZ broker. This works in conjunction with a Client Broker Connection preference, as the preferred branch will be unreachable for the endpoint once it leaves the network, thereby making only DMZ brokers available.

Hubs

Two DXL brokers can be combined together to create what is known as a ‘hub’. Hubs supply load balancing and high availability in an Active-Active configuration. If a hub is created, clients will randomly choose a specific DXL broker and switch on failure.

The general rule is that any broker that has children should instead be a hub to prevent a single point of failure caused by a split of the fabric. Additionally a Hub should be established anywhere there is a need to connect multiple branches of the fabric. For example, the Root Hub would be a hub established to communicate between all branches of the fabric. It is important that this be a hub and not a single broker in order to ensure a single broker failure does not cause segmentation of the fabric.

Service Zones

Service Zones exist to ensure that services are supplied by local resources. A Service Zone can be applied to either a broker or a hub. Service Zones contain the selected broker or hub and all children of that hub, even if they themselves also have a service zone. All requests start in the containing zone first then work their way up into higher service zones until an available service is found.

For example, if a customer has a Europe region and an Asia region they should consider implementing one Europe Service Zone and one Asia Service Zone. If in Europe there are two major datacenters, then it most likely makes best sense to implement a separate Service Zone for each data center. The Europe Service Zone would contain the two-datacenter Service Zones.

A request coming into a Europe datacenter Service Zone will be routed to a service in that zone first. If a service cannot be found in the Europe datacenter Service Zone it will then look in the higher level Europe Region Service Zone. If a service cannot be found in the Europe Region Service Zone it will then search the entire fabric for a service to handle the request.
For a simple setup you do not need service zones. If there are only a couple of brokers and a single pair of TIE servers there is no need to limit where services are located. Use service zones only if you want to ensure that clients access services as close as possible. It is also important to note that not all services need to be located within zones as many will only have single instances that are global in nature.
Bridging DXL Fabrics

With the release of DXL 1.1 and later, you can link multiple DXL fabrics managed by separate ePO instances. When DXL fabrics are bridged, clients can connect to any broker in the bridged fabrics and share DXL services. A good example is a multinational corporation with ePO instances separated by geographic regions. In this case, bridged fabrics allow clients to connect to local brokers regardless of which ePO they are managed by and services like TIE can be shared globally. See the Data Exchange Layer Product Guide for information about how to bridge DXL fabrics.

When bridging fabrics it is recommended that there be a top-level (root) hub dedicated to bridge termination and message relay. See the Multinational Enterprise Architecture diagram for an example of what that would look like. Root Hub brokers should be configured not to accept client connections, but instead be dedicated to routing messages between the different bridged fabrics.

DXL Architecture Design Checklist

This section provides a basic set of questions and conditions useful when designing a DXL and TIE infrastructure for an enterprise.

Note: Most of this is guideline information, not hard rules. You should always consider how the guidelines apply to your own unique requirements.

Inputs

When designing a DXL fabric for a given enterprise, the following sets of data are important inputs.

- How many ePO servers does your enterprise have, and how many endpoints are managed by each?
- Is DXL already in place anywhere in the environment?
- What DXL-enabled solutions need to be integrated?
- Network Diagram of the enterprise environment, including as much detail as possible around:
  - Link Speeds
  - Node Counts at each major hub
  - Hop Counts
  - Link Utilization
  - Network security policies. Are there restrictions internally on network traffic that might interfere with DXL communication?
- What are your availability and disaster recovery requirements for your DXL-enabled solution?
- Do you have requirements to support DXL connectivity for off-premise?

Step 1: Design the DXL Topology

Given the basic inputs above, we can begin designing the DXL fabric. The minimum recommended production fabric will consist of two brokers, configured in a hub, for first ePO (redundancy). In addition to this initial block, consider adding brokers based on the following rules:

- Add one or more brokers for each additional ePO in your enterprise.
- If more than one ePO, implement a pair of brokers in a hub to create a Root Hub.
- The Root Hub should typically not have services directly connected. It should provide connectivity services across the fabric.
▪ The Root Hub should typically not be used for client connections in distributed deployments.

▪ Consider adding additional brokers to if spanning across network security zones. Example: one or more broker for DMZ (multiple for redundancy).

▪ Consider adding one or more brokers for each MAJOR site (e.g. major datacenter).

▪ Review node counts, and validate that you have enough brokers for performance. Guideline: maximum of 50k clients/broker.

▪ Consider broker loss -- will you still have enough brokers if one goes down? Often regional failover is acceptable.

Step 2: Define DXL Policy
▪ Once the general topology is defined, the next step is to define the broker policies in ePO. Specifically, set the connection count for the Root Hub (since the Root Hub is handling cross traffic for the fabric, limit to 1 connection so clients cannot connect).

Step 3: Define Service Zones
▪ Next set up Service Zones to reflect your topology and help contain service requests from clients.

▪ Service Zones are defined in the context of an individual ePO server.

▪ Define one or more Service Zones per ePO, to keep services from round-robining across the fabric.

▪ Define one or more Service Zones for each geographical location hosting services, to keep traffic local where possible.

Step 4: Design Client Broker Affinity
▪ Create a Client Policy for each service zone to restrict connections to regional brokers, and apply the policy to service servers (TIE Servers).

▪ Create a Client Policy to prefer connection to local brokers, apply to clients (not needed, but if you want to have additional protection to help clients from going to other geos for services).
**DXL Services**

Applications that utilize the DXL fabric are referred to as "services". Specific methods to the services are called topics, and specific components may have the ability to broadcast on topics or query specific topics. It is important to note that all services have different design characteristics and architecture models depending on the needs of the application and clients. DXL was designed to easily add new services without massive architecture changes, but that is not to say that the services offered will not impact the design. For example, some services will necessitate global infrastructure distributed across the DXL while others will be centrally located.

**Threat Intelligence Exchange (TIE) Service**

**Overview**

McAfee’s Threat Intelligence Exchange (TIE) service utilizes the DXL fabric for the collection and distribution of threat intelligence information from and to connected DXL clients. The TIE solution consists of TIE servers, TIE clients, and the ePolicy Orchestrator (ePO) management platform. TIE clients can be both endpoints and network components.

From a design perspective, key considerations for TIE include:

1. TIE client behavior
2. Sizing of TIE client components (maximum throughputs/connections)
3. TIE server types
4. Service availability
5. Reputation replication traffic
6. McAfee Advanced Threat Detection (ATD) integration

**NOTE:** Security zoning considerations are addressed by the DXL Fabric design.

**TIE Server Types**

All TIE server appliances can also include a DXL broker. This is configured during the initial configuration wizard. Most of the time it is advisable to configure the TIE server to also include a DXL broker, unless there are firewall or other limitations that make it unadvisable. Having a DXL broker on the same machine as the TIE server should not affect performance for either up to the TIE maximums.

The types of TIE servers are:

- **Master** or **Write-Only Master** — there can be only one Master TIE server at a time. A Master server functions like any other TIE server and serves queries at the same time it is aggregating updates from Slaves and pushes out database changes. A Write-Only Master only serves to aggregate changes and push out updates. In a large network, it is recommended to have a TIE Write-Only Master server dedicated to processing updates and setting policy.

- **Slave** — TIE slaves can serve requests but relay all updates from the clients up to the Master TIE server. The best way to think of them is as a local cache. They ensure response time is quicker but do not necessarily lower bandwidth costs. They can also ensure high availability and scalability. Slaves can be promoted to Master manually at any time.

- **Report Only Slave** — The Report Only Slave is a single purpose server only for providing data to ePO. The TIE server database is an additional data source for dashboards, reports and the TIE Reputation page specifically. The Report Only Slave does not serve queries or aggregate updates. However it has a complete copy of the database and can reduce load on the Master. This makes it a good candidate for a hot spare server.
**TIE Service High Availability**

TIE delivers a highly available service through the deployment of TIE servers in Master and Slave roles. You can have only one Master server in your environment, but you can have several optional slave servers to ensure resilience of the service.

If the Master TIE server becomes unavailable, Slave TIE servers will continue to respond to TIE client requests for known file and certificate reputations that have been cached or replicated to the Slave TIE servers. The TIE service will not be able to add any additional new file reputations until a Slave has been promoted to the Master role. TIE endpoint clients will continue to assess new file executions with local TIE rules that are independent of the TIE service.

**NOTE:** A TIE Slave server must be manually promoted to the Master role via script local to the Slave server. See the Threat Intelligence Exchange Product Guide for further detail.

In the case of a failure, keep in mind that a newly provisioned server will have to sync the entire database before it will become operational. Depending on database size and LAN speed this could take some time. It is better to have a system with a synced database (such as a slave), already available. It is for this reason that you should typically never deploy only a single TIE server in a production scenario.

While other methods may exist for database replication and disaster recovery, the officially tested and supported mechanism is utilizing the built in database replication.

**TIE Database Replication**

Replication of reputation data between the Master and Slave TIE servers is performed using a number of methods:

- **Initial Replication** — When a Slave TIE server is first installed and configured, the Slave TIE server connects to the Master TIE server and a copy of the Master database is initiated.
- **Operational Database Replication** — During standard operation the changes to the Master TIE database are replicated to the Slave TIE database(s) using native database (PostgreSQL) streaming replication. Additionally, the TIE Slave connects to the TIE Master database to perform write operations.

The impact of this replication is that the addition of Slave TIE servers will increase network traffic by a linear margin. For example, the addition of two (2) Slave TIE servers will require twice as much replication traffic as a single Slave TIE server. Each additional TIE server will mean a multiplication of traffic originating from the Master TIE server. Refer to the TIE Sizing and Performance Guide for further information.

**TIE Design**

The TIE service can be thought of like a heavy DNS. TIE queries are short but somewhat common. Each time a client executes an unknown file, the TIE Module for VSE must first query the TIE server for reputation data and prevalence. Once a decision is made, the TIE Module for VSE caches the result for up to 30 days. Other TIE clients may behave somewhat differently, however the basic flow is the same.

Running one PE file may result in up to four DXL transmissions

- Cert reputation req/resp (~2.2KB)
- File reputation req/resp (~2.2KB)
- File detection event (.7KB)
- First detection event (.7KB)
During the building of this cache, it is expected that the clients will need to perform dozens to hundreds of queries. Note that executables signed by known trusted issuers will not require a lookup. For example, if an executable is signed by a valid trusted McAfee certificate, we do not need to revalidate the file itself. If a large number of clients are deployed at the same time, or a large number of new files are deployed at the same time, you can expect a large number of simultaneous queries and request latency can increase once the TIE server reaches a threshold. Users could expect to see a delay upon initial execution.

See the McAfee TIE Sizing and Performance Guide for more information.

In addition to the latency added due to load, network transport can significantly increase the time for a TIE query. While DXL routing itself does not add very much latency after the first relay, slow links can. If a query must circumnavigate the world, queries may take an unacceptable amount of time and impact initial load time and user experience. For instance, if a client in East Asia must query a server in North America across a link that adds 500ms each direction, it is advised to locate a TIE server more locally and enforce that client use the local resource unless it is unavailable. Fortunately, TIE servers can be located in service zones in order to ensure they prefer local resources.

Adding additional TIE Slave servers does come with a cost. The burden of replication ensures that all database changes must be sent to all TIE server instances. In addition, the Slave instances must reach back to the Master in order to perform the writes in the first place. Depending on the quality of the links between the Master and the Slave instances and the number of hops, it is advisable to limit the number of TIE instances and only locate them in locations with good connectivity back to the Master. Essentially, there is a tradeoff between total bandwidth used and latency.

The chart to the right shows the order of replication for a TIE query. When a client requests the reputation of a file it could be initially served by a TIE slave. After that however the information (including the query itself) must be replicated up to the Master and back to all TIE slaves.
TIE Architecture Checklist

The following guidelines can help when designing a TIE implementation for an enterprise.

Design the TIE Topology

- Always begin with one TIE Master, deployed in the most central location.
- Place one Slave for redundancy. Bandwidth between the TIE Master and TIE Slave(s) should be sufficient to support data replication.
- Add additional TIE Slaves as needed for performance (50k clients/slave).
- If you have a large number of TIE Slaves (more than 3 Slaves) and they are loaded with a large number of clients (more than 30k clients per TIE server), then consider;
  - Converting the TIE Master to a TIE Write Only Master, and
  - Adding a TIE Slave to answer the queries that the Write Only Master is no longer handling, and
  - Adding a TIE Reporter to provide performance manageability via ePO.
- If your TIE Master is handling a large number of client connections (more than 30K client connections on the TIE Master), consider converting your TIE Master to a TIE Write Only Master and adding a TIE Slave Reporter.
- Consider implementing TIE Slaves in each geographical location for reduced latency; however, evaluate the TIE Slave to TIE Master bandwidth utilization. It’s very important to have good bandwidth between TIE Master and all TIE Slaves
  - 800 kbps for DB replication, JDBC writes for new data (unknown) - roughly 1 mbps
  - TIE Servers and DXL Brokers may be co-located on the same virtual instance. Review your TIE + DXL topology, and determine if brokers and services could be consolidated to reduce the overall number of virtual systems to deploy.

Disaster Recovery Scenarios

- If an ePO disaster recovery event occurs, administrators will be required to simply run the reconfig script on TIE Servers to allow the new ePO to connect to the TIE database.
- In the event of a TIE server failure, run the reconfig script on the TIE Slave to promote it to Master.

TIE Architecture Guidance

McAfee recommends locating TIE servers in a minimal number of locations that are located centrally as measured by network latency. Essentially you want to place TIE servers in core data centers that cover your network topology but want to limit replication across network links as much as possible. At least one TIE server is needed for every 50,000 nodes. If there are other applications that are expected to perform higher number of TIE queries (network devices, MOVE, etc.) it would be advised to increase the number of TIE servers accordingly.

The assumption is that you want to keep latency within a reasonable level while not replicating across expensive or oversubscribed links.
Integration – Advanced Threat Defense

McAfee Advanced Threat Defense (ATD) integrates with TIE to enable advanced malware analysis that is shared with all DXL connected security components. For example, the TIE Client can upload files to ATD for analysis. The results of the analysis can subsequently be shared with all connected TIE Clients.

NOTE: McAfee products such as McAfee Web Gateway currently integrate directly with ATD via REST API. The resulting analysis information from the ATD is shared with TIE via the DXL.

When integrating ATD, consider the following design constraints:

**Maximum File Objects and Network Impact** - An ATD appliance supports a maximum “throughput” of file objects per day. In the event this throughput is exceeded, new file submissions will be added to a queue for analysis. There will be no user impact, however file analysis will take longer to complete. Both ATD and TIE provide a number of mechanisms to minimize the potential of the throughput being exceeded.

- TIE Client policy can be configured such that only files above a defined level of suspicion are submitted to ATD for analysis. For example, the TIE Client can send all files, OR all files that are unknown, OR only files of a suspicious classification.
- Only files below a maximum file size are submitted to ATD. This also reduces the impact of file submissions on network traffic.
- The TIE service tracks all files executed on endpoints centrally. In the event of new file being executed, TIE ensures that that file is only submitted to ATD once. Repeated execution of the same file will not initiate a re-analysis.

ATD supports cluster deployment which enables load-balancing for greater throughput. Additionally TIE supports round-robin load-balancing of ATD servers.

**TIE Client File Submission Process** – When the TIE Client submits a file to ATD, it sends the file via HTTP to the TIE Master Server (TIE 1.0-1.2) or optionally TIE Slave Server (requires TIE 1.3 and greater). The TIE Server then submits the file to ATD via HTTPS.

NOTE: The TIE Client submits only Windows Portable Executable files to ATD. Although these files are initially sent via HTTP, it is unlikely that they will contain confidential or personally identifiable data. It is also assumed that this function is enabled only for internal network connectivity.

**DXL Client Management** – When the DXL client is enabled on an ATD appliance, an embedded McAfee Agent is enabled and all TIE/DXL policy is applied from ePO.

Ensure that routing from TIE endpoint clients to the TIE Server is enabled for file uploads.
Integration – Network Security Platform

McAfee Network Security Platform (NSP) NS-Series and Virtual IPS sensors perform lookups to the McAfee Threat Intelligence Exchange database as files traverse network inspection points. Files are carved from FTP, HTTP and SMTP traffic and file hashes are checked against reputations within the TIE database. If a file is found to have a bad reputation, the file can be immediately blocked.

**NOTE:** McAfee NSP currently integrates directly with ATD via SOFA. The resulting analysis information from the ATD is shared with TIE via the DXL.

Integration – Web Gateway

McAfee Web Gateway (MWG) supports messages in DXL format to retrieve and send web security information to and from connected McAfee products. In the first iteration, MWG can perform TIE queries and block based on the reputations provided by TIE. In the future, MWG will both consume and create other types of DXL services.

Deployment Examples

Now that we have reviewed the overall guidelines for designing a DXL deployment, we will review several common example architectures. These can be easily adapted to meet most needs. The following deployment examples are shown in this section:

- Proof of Concept/Lab architecture example
- Simple Architecture Example with a single McAfee ePO Server
- Simple Architecture Example with a single McAfee ePO Server and DMZ support
- Simple Architecture Example with two McAfee ePO Servers
- Complex Architecture Example with four McAfee ePO Servers

**Proof of Concept/Lab Architecture**

For a Proof of Concept a simple TIE Master/DXL broker combination provides all necessary functionality.

This has a single point of failure. If there is no DXL Broker and/or TIE Server, clients lose DXL connectivity and fallback to direct GTI queries, if enabled.

**Simple Architecture Example With a Single McAfee ePO Server**

For production configurations, redundancy/high availability should be added. The simplest suitable configuration is two servers, each with DXL brokers, one with a TIE Master and one with a TIE Slave. Build each server separately, select the appropriate mode and connect to ePO. Once the systems are up and connected to the DXL, you then connect the DXL brokers into a DXL hub.

It is worth mentioning that running two virtual machines on the same physical host does not qualify as redundant. Each of the two servers below should run on separate VM hosts to provide best resilience.

If you lose a DXL broker the other will take the entire load automatically. If you lose the master, the slave will continue to serve queries, but the database will not update until it is promoted manually.
**Simple Architecture Example With a Single ePO Server and DMZ support**

The DXL can be used safely for clients that are located off your enterprise local network. In the case that DXL connectivity is desirable off network, DXL brokers can be located in a DMZ and the DXL port can be exposed to the internet.

Directly exposing a TIE slave is more complicated. For standard TIE queries, the clients do not need direct access to the TIE server. Clients just need to connect to a DXL broker that does have access to a TIE server.

However there is one major exception to this. Uploads to Advanced Threat Defense require direct access to the HTTP port of the TIE server. Given the security concerns regarding exposing TIE directly to the internet is a security tradeoff to not perform ATD submissions when off network. In that case a simple DXL hub with access back to the DXL is recommended.

If it is an absolute requirement to allow ATD submissions when off net, proper precautions should be taken. A NGFW, IPS or WAF/Reverse proxy should be utilized to limit access. All other ports should be blocked.
Production Architecture with Disaster Recovery

For disaster recovery purposes it is recommended to either move the slave to a separate data center or to locate an additional slave in the Disaster Recovery site. Remember that if you want clients to prefer the primary you will need to define a service zone.

Simple Architecture Example with Two McAfee ePO Servers

In this example, we will assume that the customer site has two McAfee ePO instances with 10,000 clients per server. Each ePO is located in different geographic regions. We want to implement DXL and TIE to support this distributed community.

Note: Appendix A provides a step-by-step walkthrough for implementing this design.

DXL Fabric Design

In our example, we will implement a total of 4 brokers:

- One broker for ePO 1 to support the 10,000 end-points managed by ePO 1.
- One broker for ePO 2 to support the 10,000 end-points managed by ePO 2.
- Two brokers for a root hub to connect ePO 1 and ePO 2.

The brokers in the hub can be managed via the ePO in either datacenter. Ideally this will be the datacenter considered to be the most stable/central. The root hub will not service any clients and will only exist for communication between the two datacenters or failover if a service is temporarily unavailable. The broker policy for those in the root Hub will be configured to prevent clients from connecting.

Service request traffic should be restricted to each datacenter so service zones will be established around each of the 2 “leaf” brokers off the root hub. The two services will be restricted via policy to connect to these two leaf nodes in order to keep one service in each datacenter.

This design contains the lowest possible number of brokers (4). If one of the brokers in either datacenter goes down, the clients will automatically connect to the other.
Simple DXL fabric design example

**DXL Fabric Setup**

After completing the steps on setting up your fabric, you will have the following configured:

- 4 Brokers (3 managed through ePO1, 1 via ePO2)
- 1 Root Hub (containing 2 brokers in ePO1)
- 1 Child Hub (containing 1 broker in ePO2)
- 2 Service zones
- 1 DXL bridge between the Root Hub and the ePO2 Hub
- 1 Broker policy for ePO1
- 1 Client Service Zone Policy for ePO1
- 1 Client Service Zone Policy for ePO2
Complex Architecture Example with Four McAfee ePO Servers

As a final example, consider a large global enterprise spanning four regions. Each region has its own ePO server (4 total) with two datacenters in each region (8 total datacenters). Each datacenter supports 100k users. We want to implement DXL and TIE to support this distributed global community, including the ability for roaming users to send and receive reputation updates, even when they are not located on the enterprise LAN.

DXL Fabric Design

Our overall design will begin with two DXL brokers for a root hub to connect all four ePO’s. The brokers in this hub can be managed in any of the 8 datacenters. Ideally this will be the datacenter considered to be the most stable. The root hub will not service any clients and will only exist for communication between the two datacenters or failover if a service is temporarily unavailable. The broker policy for those in the root Hub will be configured to prevent clients from connecting. The root hub will be created from two brokers in separate datacenters to account for an entire datacenter losing connectivity.

We have 100,000 clients per datacenter. As a result, we should budget a minimum of 3 brokers per datacenter. This provides total capacity for 150k total client connections. In the event of the loss of a DXL broker, we still have capacity for 100k clients via the two remaining local brokers.

Given this, each ePO Server will be manage 6 brokers dedicated to providing client connections (3 brokers per datacenter, 2 datacenters per each regional ePO server).

One additional broker will be added in North America for DMZ access.

TIE Services will be established around each of the leaf brokers/hubs off the root hub to restrict service request traffic inside of each datacenter. Additionally, service zones will be established at the region level if a service request cannot be handled within a datacenter.

The diagram below provides a review of the overall global architecture.
Complex DXL fabric design example
Appendix A: Architecture Example Steps

This appendix provides a step-by-step walkthrough of the process of implementing the *Simple Architecture Example with Two McAfee ePO Servers* example.

Create a new hub and bridge on ePO 1

1. Log into ePO 1 and verify that the DXL Brokers are registered in ePO.
   
   In ePO1 there should be three brokers deployed and properly tagged with the DXLBROKER tag. There is also the TIE service we intend to expose over DXL.
2. Edit the DXL Topology section in Server Settings.
   The Brokers will have arranged themselves in a default topology with one broker set as the parent of all other brokers.

3. Create a new Hub through the Create Hub action.
   This adds an empty hub that is not attached to the existing fabric.
4. Name the Root Hub and assign the two hub brokers to it.

Placing a parent broker into a hub will cause its children to become detached. The remaining broker should be on the same level as the root hub.

5. Drag the remaining broker under the Root Hub.

This joins the two and makes them both part of the same fabric. If they stay detached, it will result in two separate fabrics not visible to each other.
6. Select the Root Hub and run the **Create Incoming Bridge – Remote ePO Hub** action.

   This creates an empty hub under the Root Hub that is the placeholder for the information from the remote ePO. It will remain in an invalid state (highlighted in red) until the information from the remote ePO is uploaded. It is ok to save the fabric in this state.

7. Select the Incoming Bridge and click **Export Local Hub Information**. This downloads a .zip file with the name of the Root Hub, an Outgoing label, and the date. In this example, it is EPO1_Root_Hub_OUTGOING_2016Jan28.zip.
8. Save the topology.

The Incoming Bridge will still display as invalid and does not have any negative effects on the fabric.
Create a hub on ePO 2

1. Log into ePO 2 and verify that the systems are registered correctly. There should be one broker system tagged with the DXLBROKER tag.

2. Edit the DXL Topology in the Server Settings. The single DXL Broker for this ePO will be displayed.
3. Create a Hub using the **Create Hub** action.

4. Name the hub and add the single broker into it as Broker 1.
   This hub only has one broker and therefore does not support failover or high availability for the clients. This is the minimal number of brokers required.
Create a bridge between ePO 1 and ePO 2

1. Select the new Hub and run the **Create Outgoing Bridge – Remote ePO Hub** action.

2. Save the topology. The bridge is not importable until it has been saved as part of the topology.
3. Edit the topology again, select the bridge, and click **Export Local Hub Information**. This downloads a zip file containing the hub information and certificates. The zip will be named after the child hub of the bridge. In this example, it is EPO2_ePO2 Hub_INCOMING_2016Jan28.zip.

![Export Local Hub Information](image)

4. Click **Import Remote Hub Information** on the Outgoing Bridge and select the file exported from ePO1.

   In this example, it is EPO1_Root Hub_OUTGOING_2016Jan28.zip.

![Import Remote Hub Information](image)
5. Verify that the broker information that is being imported is correct.
   The broker information should match the brokers in the Root Hub on ePO 1.

6. Click OK to import the data.
   The Outgoing Bridge will now show the information of the two brokers managed by ePO1.
7. Select the ePO2 Hub and check the **Enable Service Zone** box.

8. Click **Save** to save the configuration, then perform a Wake-up on the broker to ensure it gets the configuration change.
**Complete the bridge between ePO 1 and ePO 2**

1. Return to ePO1 and edit the DXL Topology in Server Settings.

![Server Settings](image1)

2. Select the Incoming Bridge and click **Import Remote Hub Information**. Choose the zip exported from ePO2. In this example, the zip is named `EPO2_ePO2_Hub_INCOMING_2016Jan28.zip`.

![Import Remote Hub Information](image2)
3. Click **Ok** and review the imported broker information. It should be the single broker managed by ePO2.

4. Click **Ok**. The Incoming Bridge will now show the information for the broker managed by ePO2.
5. Select the broker outside of the hub and check the **Enable Service Zone** box.

6. Click **Save** and perform a wake-up on all of the DXL Brokers to ensure they get the configuration changes.
**Verify the bridge on the DXL Fabric**

Go to the Data Exchange Layer Fabric page and verify that you can see all brokers in the fabric including the ones from ePO1 and ePO2.

You can change the Labels to **Hub Name** and the View to **Brokers Grouped By Managing ePO** for a clearer distinction.
Create a DXL broker policy on ePO 1 and apply it to the Root Hub systems

1. In ePO1 create a new DXL Broker Policy.

   ![Policy Catalog screenshot]

   1. In the Policy Catalog, select New Policy.
   2. Name the policy.

   ![Duplicate Existing Policy dialog]

   - Name: Root Hub Broker Policy
   - Notes: Lowers the connection count of the Root Hub brokers to 3 to prevent clients from connecting
   - Click "OK" to duplicate and edit the last saved version of this policy.
3. Edit the policy and lower the Client Connection Limit from 50,000 to 1.
   DXL does not support Connection Limits of 0, so this is the closest we can come to preventing all client connections.

4. Save the policy and apply it ONLY to the systems containing the two brokers in the Root Hub. In this example, the only systems to receive this policy would be epo1-broker1 and epo1-broker2.
Create a DXL Client policy on ePO 1 and apply it to the TIE system

1. In ePO1 create a new DXL Client Policy.

2. Name the policy to reflect its use as a Restriction to a Service Zone.
3. Edit the new policy.

Check **Enable client broker preference**
Check **Restrict to the selected broker or hub**
Uncheck **Include all descendants of the broker or hub**
Select the single broker in the topology outside of the hub

This setup restricts any client with the policy to connect only to the selected broker. Any children added under this broker will not be available to the client.

4. Save the policy and apply it to the system hosting the TIE service for the ePO1 Service Zone.

The Service must have a DXL Client version 2.0 or greater. If the Service runs an embedded client of a lower version, the Service will have to be upgraded to one that embeds a DXL client of sufficient version.
Create a DXL client policy on ePO 2 and apply it to the TIE system

1. In ePO2 create a new DXL Client policy.

2. Name the policy.
3. Edit the new policy.

   Check **Enable client broker preference**
   Check **Restrict to the selected broker or hub**
   Uncheck **Include all descendants of the broker or hub**
   Select the ePO2 Hub

4. Save the policy and enforce it on the system containing the TIE service.
Appendix B: References

McAfee Documentation References
The following documents are available for more information:

- McAfee Data Exchange Layer Product Guide
- McAfee Threat Intelligence Exchange Installation Guide
- McAfee Threat Intelligence Exchange Product Guide

Knowledgebase Articles
KB83713 - Required communication ports for Threat Intelligence Exchange, Data Exchange Layer, and Advanced Threat Defense

DXL Network Communications

DXL Message Base Size
- 1 DXL Request/Response = ~2.2kB
- 1 DXL Event = ~.7kB

TIE Message Size
- Cert reputation req/resp (~2.2k)
- File reputation req/resp (~2.2k)

File detection event (~.7k)
- First detection event (.7k)
- Running 1 PE file may result in up to 4 DXL transmissions
- Real-world bandwidth usage is HIGHLY dependent on
  - Caching results at the TIE Client (first few days will be high)
  - Phased deployments will help spread this load over time
  - Relative amount of PEs signed by known good certs

Other factors to be determined.