ClearPass
Integration with Ivanti Endpoint Manager

ClearPass
Change Log

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Modified By</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>June 2017</td>
<td>Danny Jump</td>
<td>Draft Version</td>
</tr>
<tr>
<td>0.2</td>
<td>July 2017</td>
<td>Robert Filer</td>
<td>Peer review and wordsmith</td>
</tr>
<tr>
<td>1.0</td>
<td>July 2017</td>
<td>Danny Jump</td>
<td>Initial Published version</td>
</tr>
</tbody>
</table>

Copyright

© Copyright 2017 Hewlett Packard Enterprise Development LP.

Open Source Code

This product includes code licensed under the GNU General Public License, the GNU Lesser General Public License, and/or certain other open source licenses. A complete machine-readable copy of the source code corresponding to such code is available upon request. This offer is valid to anyone in receipt of this information and shall expire three years following the date of the final distribution of this product version by Hewlett-Packard Company. To obtain such source code, send a check or money order in the amount of US $10.00 to:

Hewlett-Packard Company
Attn: General Counsel
3000 Hanover Street
Palo Alto, CA 94304
USA

Please specify the product and version for which you are requesting source code. You may also request a copy of this source code free of charge at HPE-Aruba-gplquery@hpe.com.
Contents

Introduction and Overview .......................................................... 5
Software Requirements .................................................................. 5
ClearPass Installation and Deployment Guide .................................. 6
Pictorial View of the Integration ..................................................... 6
ClearPass and Ivanti Endpoint Manager Integration Overview .......... 7
  Ivanti Endpoint Manager Configuration ...................................... 7
  ClearPass Policy Manager Configuration .................................... 9
  EPM DB Structure ....................................................................... 9
Use-Case #1 - Checking if the Endpoint is managed/known ............... 11
Use-Case #2 - Checking on context for managed Computers ............. 11
Use-Case #3 - Checking on context for managed MobileDevices ........ 13
Appendix A - DbVisualizer ......................................................... 15
Figures

Figure 1: Pictorial View of the ClearPass and Ivanti Endpoint Manager ..........................................................6
Figure 2: Adding a new Database User ..................................................................................................................7
Figure 3: Creating a new MS-SQL login ................................................................................................................8
Figure 4: Adding a MS-SQL authorization-source .................................................................................................9
Figure 5: Defining the detail behind the MS-SQL connection .............................................................................10
Figure 6: Getting ready to define the SQL filters ...............................................................................................10
Figure 7: SQL query to check for a managed endpoint .......................................................................................11
Figure 8: Defining fields to expose returned SQL data into Enforcement Policy ..............................................12
Figure 9: Adding additional fields to be exposed in the Enforcement Policy .......................................................12
Figure 10: Utilizing the SQL fields in an Enforcement Policy .........................................................................13
Figure 11: Defining the MS-SQL Connection .......................................................................................................15
Figure 12: Example #1, a simple SQL query .......................................................................................................16
Figure 13: Example #2, a more complex SQL query ..........................................................................................16
Introduction and Overview

This TechNote covers how to integrate ClearPass Policy Manager with Ivanti Endpoint Manager [EPM], previously known as LANDESK Management Suite. Ivanti® Endpoint Manager, powered by LANDesk, increases user and IT productivity by helping IT administrators gather detailed device data, automate software and OS deployments, and quickly fix user issues. It integrates and unifies the management of all end-user devices. Ivanti Endpoint Manager provides extensive data about managed and unmanaged devices through industry-leading discovery and inventory technology. This information is key for boosting your level of insight into:

- Software compliance fines and over-purchasing licenses
- Failing to meet corporate service level agreements
- Security vulnerabilities on end-user devices

No matter what leading device or OS your users have in their hands, you can manage various endpoints and operating systems with Ivanti Endpoint Manager including:

- PCs/Servers
  - Windows
  - macOS
  - Linux
  - Unix
  - Chromebooks
- Mobile Devices
  - iOS
  - Android
  - Windows

Software Requirements

At the time of writing, ClearPass 6.6.7 is the latest available and recommended release. Any subsequent ClearPass software release will support this integration. ClearPass runs on either hardware appliances with pre-installed software, or as a Virtual Machine under the following hypervisors. Hypervisors that run on a client computer such as VMware Player are not supported.

- VMware ESXi 5.0, 5.1, 5.5, 6.0, 6.5 or higher
- Microsoft Hyper-V Server 2012 or 2016 R2
- Hyper-V on Microsoft Windows Server 2012 or 2016 R2
- KVM on CentOS 6.6, 6.7, or 6.8.

For Ivanti Endpoint Manager, a minimum of release 2016.3 is required.
ClearPass Installation and Deployment Guide

This document assumes your ClearPass environment is already configured and operational. If you require assistance with basic deployment, refer to the following deployment guide: http://www.arubanetworks.com/techdocs/ClearPass/Aruba_DeployGd_HTML/Default.htm

Pictorial View of the Integration

*Figure 1: Pictorial View of the ClearPass and Ivanti Endpoint Manager*

1. User authenticates to ClearPass over wireless or wired network

2. ClearPass processes the authentication and as per policy makes an authorization call to the Ivanti Endpoint Manager [EPM] MS-SQL Database via the embedded ODBC connector.

3. EPM returns context about the endpoint based up on the query sent. This data is made available to the Enforcement Policy engine. From here an enforcement profile can be applied to the authenticating endpoint based upon the returned data.
ClearPass and Ivanti Endpoint Manager Integration Overview

The integration between the two products is accomplished by the use of SQL. EPM runs on top of a standard Windows 2008/2012 Server and utilizes MS-SQL as its data repository.

Within the Ivanti database are a vast number of useful tables and views. This document covers a number of use-cases using a small fraction of the data in these tables. You can extract additional data from other tables, but the views/tables used are based specifically upon advice and guidance from Ivanti as this integration has been jointly researched and developed.

In ClearPass Policy Manager uses EPM as an authorization source. The work flow is such that every time a device either authenticates or re-authenticates, and the authentication event matches a service-policy, ClearPass Policy Manager will make a SQL call to EPM to return endpoint related context that it can be used for role-mapping or policy enforcements.

As a brief example:

When a device authenticates, ClearPass Policy Manager checks to see if the device is being managed by EPM. Policy states that users in the Finance department, as determined by their active-directory group membership, must use devices that are EPM managed. Assuming the device is EPM managed, the device's disk must be encrypted and it must be running the latest software as determined by corporate policy [to minimize any vulnerability exposures].

Ivanti Endpoint Manager Configuration

Minimum configuration is required in EPM. Specifically, ClearPass Policy Manager must be able to connect to the underlying MS-SQL DB. Depending on the MS-SQL deployment and configuration you might need to request that the DB administrator adds a unique SQL user that is used exclusively by Policy Manager to connect and retrieve data. As an example, Figure 2 shows how to add a new user to the MS-SQL DB using the Microsoft SQL Management Studio add-on.

**Figure 2: Adding a new Database User**
As part of generating this new login, pay particular attention to a few key fields as shown below, especially if you use a different policy for the password. The Login name will be used later when defining the authorization source in Policy Manager. In the example below, a SQL user Login is used, though options exist to use a Windows domain user. Carefully consider the password expiration. If the login is a Windows user and password expiration is in use, it's possible for the password to expire and Policy Manager will no longer be able to connect to EPM and verify authorization context. In the below example, password expiration is disabled. Set the Default database to ulddb. This final step is not a necessity but a recommendation.

**Figure 3: Creating a new MS-SQL login**

The creation of the above credentials is not required, but by using a unique account dedicated to this task it allows authentication to the underlying SQL DB to be carefully controlled as well as logged and tracked.

---

**NOTE**

The Policy Manager authorization connector only requires READ-ONLY-ACCESS.
ClearPass Policy Manager Configuration

The majority of the configuration needs to be completed on ClearPass Policy Manager. The connector in use is the embedded Microsoft MS-SQL connector. It’s used to make authorization checks for the authenticating endpoints at the time of authentication. As noted previously, the EPM platform can contain several sub-components such as Ivanti Antivirus Manager or Ivanti Patch manager. If these sub-components are installed it allows the option to make additional, more complex, decisions on the posture/suitability of an endpoint to ensure it matches the corporate policy to gain access to the network and corporate resources.

There are a number of use-cases within this TechNote. The more complex ones that could be developed with additional EPM modules installed are not covered, but the basic configuration in this TechNote provides guidance and insight into how other more complex integrations might be achieved.

Having a basic understanding of the data structure within the EPM database and how it relates to a device authenticating against Policy Manager is key.

EPM DB Structure

EPM stores “interesting” data across a number of different tables. ClearPass uses the mac-address of a device as the primary key for reference into the endpoint table. EPM uses the Computer_Identifier value in the same way as their primary key to identify endpoints across differing SQL tables. When looking up data within the EPM database, you must first search a table called BoundAdapter using the device’s mac-address to extract the Computer_Idn [Computer Identifier], then use this value to lookup data across other tables you want to use.

Before building the SQL queries/filters you need to configure the SQL authentication-source. Under Configuration -> Authentication -> Sources -> Add [Choose Generic SQL DB]

Figure 4: Adding a MS-SQL authorization-source
Take special notice of the settings below:

- **Server Name**: IP address of target
- **Port**: 1433 [assuming you've set the MS-SQL to use 1433]
- **Database Name**: ulddb
- **Login Username/Password**: Credentials you configured earlier
- **ODBC**: MSSQL
- **Password Type**: Cleartext

Figure 5: Defining the detail behind the MS-SQL connection

The filter configuration, SQL command query string, and field names will be discussed next.

Figure 6: Getting ready to define the SQL filters
Use-Case #1 - Checking if the Endpoint is managed/known

Within the Authentication Source, define multiple queries to perform multiple tasks. For the use-case of checking if the device is managed, the SQL filter 'device-exist' counts the number of times the mac-address was found in BoundAdpater. If the value numdev [Number of Devices] is equal to zero, then the device is not found and its assumed to be un-managed. This can be used as a simple check in an Enforcement Policy. See Figure 10 for an example.

**Figure 7: SQL query to check for a managed endpoint**

The SQL below is provided for ease of copy/paste and is the same as in the above graphic Filter Query.

```
select count(PhysAddress) numdev from BoundAdapter where PhysAddress = '%{Connection:Client-Mac-Address-NoDelim}'
```

Use-Case #2 - Checking on context for managed Computers

The use-cases following will depend heavily on your requirements, and the data available from EPM based on the components enabled inside of EPM. To expand, some of the data available in the 'Computer' table is only populated if you are using the Patch or Anti-Virus features of EPM. However, there are numerous common attributes ClearPass Policy Manager can check without these add-ons. Listed below is a sample of some of the common security contextual policy data you may find relevant:

- **Device-Type**: Workstation, Virtual Workstation, Virtual Server, MAC, MobileDevice, Portable
- **DeviceName**: - Various
- **LoginName**: - Various
- **DomainName**: Various
- **HWLastScanDate**
- **SWLastScanDate**
- **VALastScanDate**
- **SpywareLastScanDate**
- **SecurityLastScanDate**
- **ScanType**: Full, Delta, Agentless Mobile
- **LastPolicySync**

Once you decide on the attributes required, there are two steps required to populate the data and make it available to an Enforcement Policy. The first is to selectively name the fields you want to expose from the Computer table in your SQL statement in the Filter Query. These fields must also be defined correctly in the filter query. Pay special attention to the **Field Name** and **Data Type** in the below example [Figure 8]. Also
pay attention to the field-names in the SQL and ensure the same fields are created in the Field definition (Note: these are case-sensitive)

```
select Type, LoginName, HWLastScanDate, SWLastScanDate, ScanType, DomainName from Computer where Computer_Idn in (select Computer_Idn from BoundAdapter where PhysAddress = '%{Connection:Client-Mac-Address-NoDelim}')
```

**Figure 8: Defining fields to expose returned SQL data into Enforcement Policy**

<table>
<thead>
<tr>
<th>Name</th>
<th>Alias Name</th>
<th>Data type</th>
<th>Enabled As</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Type</td>
<td>String</td>
<td>Role</td>
</tr>
<tr>
<td>LoginName</td>
<td>LoginName</td>
<td>String</td>
<td>Role</td>
</tr>
<tr>
<td>HWLastScanDate</td>
<td>HWLastScanDate</td>
<td>Date-Time</td>
<td>Role</td>
</tr>
<tr>
<td>SWLastScanDate</td>
<td>SWLastScanDate</td>
<td>Date-Time</td>
<td>Role</td>
</tr>
<tr>
<td>ScanType</td>
<td>ScanType</td>
<td>String</td>
<td>Role</td>
</tr>
<tr>
<td>DomainName</td>
<td>DomainName</td>
<td>String</td>
<td>Role</td>
</tr>
</tbody>
</table>

In the above example, there are SIX fields defined. If you wanted to add a new field, say **SpywareLastScanDate**, then your SQL and definition would look like the following.

```
select Type, LoginName, HWLastScanDate, SWLastScanDate, ScanType, DomainName, SpywareLastScanDate from Computer where Computer_Idn in (select Computer_Idn from BoundAdapter where PhysAddress = '%{Connection:Client-Mac-Address-NoDelim}')
```

Then add this new additional field to the filter, as shown below on line 7.

**Figure 9: Adding additional fields to be exposed in the Enforcement Policy**

<table>
<thead>
<tr>
<th>Name</th>
<th>Alias Name</th>
<th>Data type</th>
<th>Enabled As</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Type</td>
<td>String</td>
<td>Role</td>
</tr>
<tr>
<td>LoginName</td>
<td>LoginName</td>
<td>String</td>
<td>Role</td>
</tr>
<tr>
<td>HWLastScanDate</td>
<td>HWLastScanDate</td>
<td>Date-Time</td>
<td>Role</td>
</tr>
<tr>
<td>SWLastScanDate</td>
<td>SWLastScanDate</td>
<td>Date-Time</td>
<td>Role</td>
</tr>
<tr>
<td>ScanType</td>
<td>ScanType</td>
<td>String</td>
<td>Role</td>
</tr>
<tr>
<td>DomainName</td>
<td>DomainName</td>
<td>String</td>
<td>Role</td>
</tr>
<tr>
<td>SpywareLastScanDate</td>
<td>SpywareLastScanDate</td>
<td>String</td>
<td>Role</td>
</tr>
</tbody>
</table>

After the fields are defined, you can use them in an Enforcement Policy like normal. As shown in Figure 10 below, all the available fields are exposed based upon the filters defined above. Note: **numdev** comes from the first SQL query to check if the mac-address is known.
Enforcement Policies can be created using the fields defined above. Associated Enforcement Profiles can be used to enforce the defined policies. The above example has three conditions/actions used in testing.

1. If numdev = 0, then Deny Access. Essentially, if the device is NOT under management then deny access. A more complex enforcement action could be one that redirects the user to a captive portal, asking them to enroll their device under EPM management to comply with corporate policy.

2. If the device Type is 'Portable', only allow limited access.

3. If LoginName exists, allow full access.

These are just simple use-case examples. One final thing to cover in this overview is how data is accessed in the underlying EPM database. Look again at the SQL used above:

```sql
select Type, LoginName, HWLastScanDate, SWLastScanDate, ScanType, DomainName, SpywareLastScanDate
from Computer
where Computer_Idn in (select Computer_Idn from BoundAdapter
where PhysAddress = '%{Connection:Client-Mac-Address-NoDelim}')
```

The key thing here, beyond naming the fields to be returned [Type, LoginName, HWLastScanDate, SWLastScanDate, ScanType, DomainName], is choosing the records that match the mac-address of the authenticating device. This is based upon the value returned as Computer_Idn from the BoundAdapter table. This is then matched to the same record in the Computer table. Within EPM, the Computer_Idn is used across multiple tables to refer to the same endpoint. Said another way, Computer_Idn is used as EPM's primary key to refer to endpoints.

**Use-Case #3 - Checking on context for managed MobileDevices**

Using the same logic and almost the same SQL as above, this can also be used for MobileDevices. Some of the fields names used in the Computer table are not present, but new ones which are more appropriate to
MobileDevices are such as CarrierNetwork. Below is a short list of fields you may find appropriate for MobileDevices.

- **NotifyType**: Android, Mac OS X, Windows, iOS
- Model
- AgentVersion
- SubscriberCarrierNetwork
- Rooted
- HotspotEnabled
- MobileLastScanDate
Appendix A - DbVisualizer

In developing this and other SQL-based integrations, it was useful to be able to view the available fields across different tables. There are multiple ways to get visibility into the SQL tables and list their fields. One such way is to use a SQL tool like DbVisualizer. It's available in a free or paid version. The free version is sufficient for the purpose required here. After installing the tool, configure a connection similar to the below.

*Figure 11: Defining the MS-SQL Connection*

Choose the same settings as shown above, paying attention to the Driver, Database Port and the Database. You'll also need the UserID/Password you created previously to authenticate to the SQL DB.

After configuring your connection settings, start the SQL Commander and enter your SQL commands to retrieve the data required. Below are a couple of examples.

The below query returns a list of all the records in the **BoundAdapter** table, which includes all the known mac-addresses [PhysAddress].
Figure 12: Example #1, a simple SQL query

The below query counts all the records from the BoundAdapter table where the mac-address is 3010E41DB39. This is the check to see whether a mac-address is managed or not. If the count is greater than 0 then it's a known endpoint.

Figure 13: Example #2, a more complex SQL query