Forescout

eyeExtend Connect Module: Data Exchange Plugin

Configuration Guide

Versions 3.9.2 and 3.9.3
Contact Information
Forescout Technologies, Inc.
190 West Tasman Drive
San Jose, CA 95134 USA
https://www.Forescout.com/support/
Toll-Free (US): 1.866.377.8771
Tel (Intl): 1.408.213.3191
Support: 1.708.237.6591

About the Documentation
- Refer to the Technical Documentation page on the Forescout website for additional documentation: https://www.Forescout.com/company/technical-documentation/
- Have feedback or questions? Write to us at documentation@forescout.com

Legal Notice
© 2020 Forescout Technologies, Inc. All rights reserved. Forescout Technologies, Inc. is a Delaware corporation. A list of our trademarks and patents can be found at https://www.Forescout.com/company/legal/intellectual-property-patents-trademarks. Other brands, products, or service names may be trademarks or service marks of their respective owners.
# Table of Contents

About eyeExtend Connect Module: Data Exchange .................................................. 4

How to Work with DEX Plugin .............................................................................. 4

How to Install DEX Plugin ................................................................................. 7

Configure DEX Plugin ....................................................................................... 8

DEX Plugin Integration Scenarios ......................................................................... 9

Work with SQL and LDAP Databases ................................................................. 13
  DEX Plugin – Define Connections to External Databases ...................................... 14
  DEX Plugin – Define LDAP or SQL Query Statements ........................................... 19
  DEX Plugin – Define Host Properties .................................................................. 22
  DEX Plugin – Import and Export Definitions ...................................................... 31
  DEX Plugin – About Update Statements ............................................................. 33
  DEX Plugin – Manage Communication with Database Servers ............................ 33

Work with External Web Services ...................................................................... 36
  Define Requests to External Web Services ......................................................... 36
  Define Host Properties Based on Data from External Web Services .................. 51

Work with the CounterACT Web Service ............................................................ 56

Actions and Properties for Data Integration ...................................................... 63

Appendix 1: Forescout Property and Data Types ............................................... 70

Appendix 2: Use Advanced JDBC Attributes ..................................................... 75

Appendix 3: Map Information from External Servers Example ......................... 76

Appendix 4: Submit Data with the Forescout Web API ...................................... 77
  Web Service Implementation Overview ............................................................... 77
  Web Service Interaction ...................................................................................... 78
  CounterACT Web Service Transactions ............................................................ 83

Appendix 5: External Web Service Parser Construction .................................... 100
About eyeExtend Connect Module: Data Exchange

Forescout eyeExtend Connect Module: Data Exchange (DEX) Plugin lets the Forescout platform use the following methods to communicate with external entities:

- SQL®, Oracle®, and LDAP servers: The Forescout platform queries (pull) and updates (push).
- Web services: The Forescout platform queries external services and receives updates via the CounterACT® web service hosted by Data Exchange.

Install DEX on Enterprise Manager or standalone Appliances to work with the CounterACT web service.

To use DEX, you should have a solid understanding of User Directory, SQL database and/or web services concepts, functionality, and terminology, and understand how Forescout policies and other basic features work.

Supported Forescout Platform Version

The following table lists the Forescout platform version that works with each version covered by this guide.

<table>
<thead>
<tr>
<th>Version</th>
<th>Forescout Platform Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.9.2</td>
<td>Minimum version: 8.1.4 or 8.2.1</td>
</tr>
<tr>
<td>3.9.3</td>
<td>Minimum version: 8.1.4 or 8.2.1</td>
</tr>
</tbody>
</table>

About Certification Compliance Mode

Forescout eyeExtend Connect Module: Data Exchange Plugin supports Certification Compliance mode. For information about this mode, refer to “Certification Compliance” in the Forescout Installation Guide.

About Support for Dual Stack Environments

The Forescout platform detects endpoints and interacts with network devices based on both IPv4 and IPv6 addresses. However, IPv6 addresses are not yet supported by this eyeExtend module. The functionality described in this document is based only on IPv4 addresses. IPv6-only endpoints are typically ignored or not detected by the properties, actions, and policies provided by this eyeExtend module.

How to Work with DEX Plugin

This topic describes how to work with the module and module requirements.

Requirements

See the following:

- Forescout Requirements
• Connectivity Requirements
• Forescout eyeExtend Connect Licensing Requirements

Forescout Requirements

DEX requires the following Forescout components:

• A module license for the Open Integration Module or the eyeExtend Connect Module. See Forescout eyeExtend Connect Licensing Requirements for details.
• Authentication Module with the User Directory Plugin.

Connectivity Requirements

Verify that the Forescout platform has permission to connect to external servers.

Forescout eyeExtend Connect Licensing Requirements

This Forescout eyeExtend module requires a valid license. Licensing requirements differ based on which licensing mode your deployment is operating in:

• Per-Appliance Licensing Mode
• Flexx Licensing Mode

To identify your licensing mode:

• From the Console, select Help > About Forescout.

Per-Appliance Licensing Mode

When installing the module, you are provided with a 90-day demo license.

If you would like to continue exploring the module before purchasing a permanent license, you can request a demo license extension. Consult with your Forescout representative before requesting the extension. You will receive email notification and alerts at the Console before the demo period expires.

To continue working with the module after the demo period expires, you must purchase a permanent module license.
Demo license extension requests and permanent license requests are made from the Console.

This module may have been previously packaged as a component of an Integration Module which contained additional modules. If you already installed this module as a component of an Integration Module, you can continue to use it as such. Refer to the section about module packaging in the Forescout Administration Guide for more information.

**Requesting a License**

When requesting a demo license extension or permanent license, you are asked to provide the device *capacity* requirements. This is the number of devices that you want this license to handle. You must define at least the number of devices currently detected by the Forescout platform. You can request a license that handles more to ensure that you are licensed for support on additional devices as your deployment grows.

Enter this number in the **Devices** pane of the Module License Request wizard, in the Console Modules pane.

To view the number of currently detected devices:

1. Select the **Home** tab.
2. In the Views pane, select the **All Hosts** folder. The number in parentheses displayed next to the **All Hosts** folder is the number of devices currently detected.

**Flexx Licensing Mode**

When you set up your Forescout deployment, you must activate a license file containing valid licenses for each feature you want to work with in your deployment,
including eyeExtend modules. After the initial license file has been activated, you can update the file to add additional eyeExtend licenses or change endpoint capacity for existing eyeExtend modules. For more information on obtaining eyeExtend licenses, contact your Forescout sales representative.

- **No demo license is automatically installed during system installation.**

License entitlements are managed in the Forescout Customer Portal. After an entitlement has been allocated to a deployment, you can activate or update the relevant licenses for the deployment in the Console.

Each eyeExtend license has an associated capacity, indicating the number of endpoints the license can handle. The capacity of each eyeExtend license varies by module but does not exceed the capacity of the Forescout eyeSight license.

- **Integration Modules, which package together groups of related licensed modules, are not supported when operating in Flexx Licensing Mode. Only eyeExtend modules, packaging individual licensed modules are supported.** The eyeExtend Connect Module is an eyeExtend module even though it packages more than one module.

### More License Information

For more information on eyeExtend (Extended Module) licenses:

- **Per-Appliance Licensing.** Refer to the Forescout Administration Guide.
- **Flexx Licensing.** Refer to the Flexx Licensing How-to Guide.

You can also contact your Forescout sales representative for more information.

### How to Install DEX Plugin

This topic describes how to install the DEX Plugin.

**To install the module:**

1. Navigate to one of the following Forescout download portals, depending on the licensing mode your deployment is using:
   - **Product Updates Portal** - Per-Appliance Licensing Mode
   - **Customer Portal, Downloads Page** - Flexx Licensing Mode

   To identify your licensing mode, select Help > About ForeScout from the Console.

2. Download the module .fpi file.

3. Save the file to the machine where the Console is installed.

4. Log into the Console and select Options from the Tools menu.

5. Select Modules. The Modules pane opens.

6. Select Install. The Open dialog box opens.

7. Browse to and select the saved module .fpi file.
8. Select **Install**. The Installation screen opens.

9. Select **I agree to the License Agreement** to confirm that you have read and agree to the terms of the License Agreement and select **Install**. The installation cannot proceed unless you agree to the license agreement.

   ▶ The installation begins immediately after selecting Install and cannot be interrupted or canceled.

   ▶ In modules that contain more than one component, the installation proceeds automatically one component at a time.

10. When the installation completes, select **Close** to close the window. The installed module is displayed in the Modules pane.

    ▶ Some components are not automatically started following installation.

### Configure DEX Plugin

To access the configuration pane, select **Options** in the Console, then, select **Data Exchange**.

DEX supports interaction with several types of external data sources, which are completely independent of each other. For example, configuration settings let you define external SQL database servers, but these settings are only relevant if you are defining properties that are populated from a SQL database. You need not define a SQL server if you are working with another external data source, such as an external web source.

See [DEX Plugin Integration Scenarios](#) for configuration procedures required for working with data sources.
DEX automatically reboots when you define new data sources or host properties, or if you change any configuration settings.

The general configuration settings described in this topic are common to all implementation scenarios.

Define Null Values

When the Forescout platform receives information from external sources, some fields may not have values. External sources may indicate an empty field value in different ways. To let you define conditions that identify empty fields, null values must be represented in a uniform way in Forescout properties. You can define how the Forescout platform represents empty fields in properties populated by external data.

In the Data Exchange pane, select the General Settings tab. In the NULL Values section, you can define how the Forescout platform represents empty data fields when it maps received data to corresponding host properties.

DEX Plugin Integration Scenarios

This topic links you to implementation roadmaps for specific integration scenarios.

- Working with external SQL and LDAP databases:
  - Create a Forescout Host Property Based on an External Database
  - Update an External SQL Database
- Working with External Web Services:
  - Update a Forescout Property List
  - Create a Forescout Host Property Based on an External Web Service
  - Update an External Web Service
- Working with the CounterACT Web Service:
  - Define CounterACT Web Service Accounts
Create a Forescout Host Property Based on an External Database

In this scenario, a custom host property is created in the Forescout platform. The Forescout platform populates the property value by polling an external server. For example, the Last Logon Active Directory attribute is retrieved using an LDAP query statement and mapped to a new Last Logon Time custom Forescout property.

Implemented by:

- Forescout administrator
- External server administrator

To retrieve data from external databases:

1. Define a user account for the Forescout platform on each external DB server you want to work with.
2. Install the plugin. See How to Install DEX Plugin.
3. Configure the connection to external DB servers. See DEX Plugin – Define Connections to External Databases.
4. Define a custom Forescout property that holds retrieved fields or attributes.
   - Define queries that retrieve data from external servers. See DEX Plugin – Define LDAP or SQL Query Statements.
   - Map retrieved data to custom properties. See DEX Plugin – Define Host Properties.
5. Include the new property in Forescout policies.
   
   The Forescout platform periodically queries the external DB server and updates the host property based on retrieved data.

For general information, see Work with SQL and LDAP Databases.

Update an External SQL Database

Update values in an external database by sending a custom update statement to an external server. For example, you can update a database table value from not compliant to compliant when a Forescout policy detects endpoint Anti-virus compliance status.

Implemented by:

- Forescout administrator
- External server administrator

To update external SQL databases:

1. Define a user account for the Forescout platform on the external DB servers you want to work with. The user account must have write permission for the database tables you want to modify.
2. Configure the connection with external databases.
3. Use the DEX Update External Databases action in a policy. The Forescout platform updates records in external SQL databases for hosts that match policy conditions.

For details, see DEX Update External Database Action and DEX Plugin – About Update Statements.

Create a Forescout Host Property Based on the CounterACT Web Service

In this scenario, a custom host property is created in the Forescout platform. An external application or service platform can then update the property value using the CounterACT Web Service. For example, service call records from a central help desk can be added to host information in the Forescout platform.

Implemented by:
- Forescout administrator
- External application/service programmer

To implement this scenario, perform the following steps:
1. Configure and test web service connectivity to the Forescout platform.
2. Define access credentials for the CounterACT Web Service. See Work with the CounterACT Web Service for details.
3. Create a new host property in the Forescout platform. See Define Host Properties from the CounterACT Web Service and Appendix 1: Forescout Property and Data Types for details.
4. Program the web service call on the external application. See Appendix 4: Submit Data with the Forescout Web API for details.
5. Test the interaction.

Update a Forescout Property List

In this scenario, a custom Property List is created in the Forescout platform. An external application or service platform can then update the Forescout Property List using the CounterACT Web Service. For example, employee IDs can be added to the Property List in the Forescout platform.

Implemented by:
- Forescout administrator
- External application/service programmer

To update a Forescout property list:
1. Configure and test web service connectivity to the Forescout platform.
2. Define access credentials for the CounterACT Web Service. See Work with the CounterACT Web Service for details.
3. Create a new Property List in the Forescout platform. For more information, refer to “Defining and Managing Lists” in the Forescout Administration Guide.
4. Program the web service call on the external application. See Appendix 4: Submit Data with the Forescout Web API for details.
5. Test the interaction.

**Create a Forescout Host Property Based on an External Web Service**

In this scenario, a custom host property is created in the Forescout platform. The Forescout platform populates the property value by polling an external web service.

Implemented by:
- Forescout administrator
- External application/service programmer

**To implement this scenario:**

1. Define a user account for the Forescout platform in the external web service you want to work with.

2. Define a custom Forescout property that holds retrieved fields or attributes.
   - Define a query message that retrieves data from the external web service. See [Define Requests to External Web Services](#) for details.
   - Map retrieved data to custom properties. See [Define Host Properties Based on Data from External Web Services](#) for details.

3. Include the new property in Forescout policies.
   
   To resolve the custom property, the Forescout platform queries the external web service and updates the host property based on retrieved data.

For general information, see [Work with External Web Services](#).

**Update an External Web Service**

Send values to an external web service by submitting a custom request message to the service.

Implemented by:
- Forescout administrator
- External service administrator

**To update external SQL databases:**

1. Define a user account for the Forescout platform in the external service you want to work with. The user account must have write permission.

2. Use the DEX Send Web Service Request action in a policy. The Forescout platform sends the web request to the service for hosts that match policy conditions.
   
   See [DEX Send Web Service Request Action](#).

**Use External Web Server Certificate Validation**

You can configure DEX to set up the connection to the server using common commercial Certificate Authority certificates. Alternately, you can un-check the Validate Server Certificate option so that DEX always sets up the connection.

Implemented by:
• Forescout administrator
• External service administrator

To validate a web server certificate:
1. Define requests to external web services.
2. In the Add External Web Service Request wizard’s Authentication pane, you can select Validate Web Server Certificate.

By default, this option is not selected, meaning that DEX always sets up the connection, regardless of whether the server certificate is a common commercial Certificate Authority issued certificate.

If you select this option, certificate validation is performed.

– If the server certificate is a common commercial Certificate Authority issued certificate, DEX sets up the connection to the server.
– If the server certificate is not a common commercial Certificate Authority issued certificate, DEX does not set up the connection to the server.

Work with SQL and LDAP Databases

Data Exchange supports LDAP user directory servers and SQL database servers. You can work with both types of server simultaneously.

LDAP Directory Servers

LDAP directory servers can be used as your external data source. These include any of the following servers you defined in the CounterACT User Directory Plugin:

• Microsoft Active Directory®
• Novell® eDirectory
• Sun Directory
• IBM® Notes® (formerly IBM Lotus® Notes)
• Open LDAP Directory

LDAP over SSL/TLS, also known as Secure LDAP (LDAPS), is supported.

The servers you defined in the CounterACT User Directory Plugin are automatically available as external data sources, and do not need to be redefined.

SQL Database Servers

The following SQL database servers are supported:

• Oracle® MySQL™ – version 5.5 and above
• Microsoft® SQL Server®
  – SQL Server 2005
  – SQL Server 2008
  – SQL Server 2008R2
Depending on the desired **DEX Plugin Integration Scenarios**, perform the following procedures:

- **DEX Plugin – Define Connections to External Databases**. You must configure the Forescout platform’s communication with external servers to use the features of DEX.

- **DEX Plugin – Define LDAP or SQL Query Statements**. A **Query Statement** is a SQL statement or LDAP filter string that the Forescout platform sends to the external server to retrieve host information.

- **DEX Plugin – Define Host Properties**. Define new Forescout host properties that hold data values retrieved from the external server.

In some cases, it is useful and more efficient to edit existing definitions. See [Map External Data to Host Properties](#) for details on exporting configured query, update, and property definitions.

**DEX Plugin – Define Connections to External Databases**

To retrieve data from an external server, you must configure how the Forescout platform connects and logs into the server.

DEX works with LDAP directory servers and SQL database servers:

- Define the connection to LDAP servers in the User Directory Plugin configuration pane. The Base Domain Name used for data queries is part of this definition.

- Define the connection to SQL databases in the Data Exchange configuration pane.

**To define connection to SQL database servers:**

1. In the Data Exchange configuration pane, select the SQL/LDAP tab and select the DB Servers tab. Then select **Add**.
2. Define the following database server parameters in the General pane.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Server Name</strong></td>
<td>Enter a name for this connection.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Enter a description of this connection.</td>
</tr>
<tr>
<td><strong>IP Address</strong></td>
<td>Enter the server IP address.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Enter the server port.</td>
</tr>
<tr>
<td><strong>Database Instance Name</strong></td>
<td>Specify a database on the server.</td>
</tr>
<tr>
<td><strong>Database Driver</strong></td>
<td>Select a database driver from the <strong>Database Driver</strong> drop-down menu.</td>
</tr>
<tr>
<td><strong>Database Username</strong></td>
<td>Enter the name of a user who can access the database in the <strong>Database Username</strong> field. DEX does not support users defined as &quot;sys&quot; users (system administrators). If your environment uses NTLMv2 authentication, use the standard <code>domain\username</code> format.</td>
</tr>
<tr>
<td><strong>Database Password</strong></td>
<td>Enter the password of the specified user in the <strong>Database Password</strong> field.</td>
</tr>
<tr>
<td><strong>Verify Password</strong></td>
<td>Re-enter the password to verify it.</td>
</tr>
</tbody>
</table>
If you want to update information on SQL databases, the user you specify must have write permission for the database tables you want to update.

3. (Optional) To define additional connection attributes in the Java Database Connectivity (JDBC) connection string, select **Advanced**.

The JDBC protocol represents the database connection as a URL string with attributes. The default string displayed reflects the selected database driver, and includes placeholder tags for the configuration fields you specified earlier.

The default URL string used by the Forescout platform to connect to all SQL servers is shown. Edit this string to add optional configuration attributes. For example, the following string adds the **SSL** attribute to a connection that uses the **PostgreSQL** database driver:

```
jdbc:postgresql://{hostname}:{port}/{dbname}?user={user}&password={passwd}&ssl=true
```

- **Add only attributes supported by the database driver you specify. The Forescout platform does not validate these optional attributes.**

**a.** Select **Default** to clear your modifications and restore the default URL string.

**b.** Select **OK** to save the URL string. The Forescout platform uses the customized string to connect to this SQL server.

- **The customized string is only used with this SQL server.**

4. Select **Next**.
5. Define the following parameters to control traffic to the database server:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Requests</td>
<td>Specify a maximum number of requests that DEX sends to the server in the selected time period. If DEX generates more requests, they are queued to a buffer.</td>
</tr>
<tr>
<td>Maximum Buffered Requests</td>
<td>Enter the maximum number of requests that are queued for submission. When the buffer is full, new requests are dropped, and related host properties are evaluated as Irresolvable.</td>
</tr>
</tbody>
</table>

Specify values in this pane depending on the resources available on the database server.

Additional settings are available to monitor and control traffic with external servers. See DEX Plugin – Manage Communication with Database Servers for details.

6. Select Next.
This pane lists the organizational replica servers that are used if the server defined in the General pane fails.

7. (Optional) To add a replica server, select **Add**.

8. Define the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>Enter a name for the replica server.</td>
</tr>
<tr>
<td><strong>IP Address</strong></td>
<td>Enter the IP address of the replica server.</td>
</tr>
<tr>
<td><strong>Accessed By</strong></td>
<td>Specify whether this replica server can be accessed by all CounterACT devices, or by one particular Appliance or Enterprise Manager. If you select <strong>All</strong>, verify that all Appliances have access to the server. This option is recommended as it enables faster resolution.</td>
</tr>
</tbody>
</table>

9. Select **OK**. The server you defined is added to the replica server table. Repeat for additional replica servers as required.
10. Select **Finish**. The database and server information is added to the table in the DB Servers tab.

11. Select **Apply**.

12. (Optional) Repeat this procedure to define additional SQL server connections.

### Test SQL Server Connections

Use the following procedure to test connections to external SQL servers.

**To test server connections:**
1. In the Data Exchange pane, select the DB Servers tab.
2. Select the servers you want to test.
3. Select **Test**. The Configuration Test dialog box opens.
   
   Each CounterACT device attempts to log in to each server. The results are listed in the dialog box.

### DEX Plugin – Define LDAP or SQL Query Statements

A *Query Statement* is a set of SQL statements or User Directory filter strings that the Forescout platform submits to the external server to retrieve host information.

Before you perform this procedure, see About Query Statements. If you are working with a SQL database, see Tips for SQL Query Statements.

**To define a query statement:**
1. In the Data Exchange pane, select the SQL/LDAP tab and then select the DB Queries tab.

2. Select **Add**.
3. Define the following query statement parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Enter a name for the query.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Enter a description.</td>
</tr>
<tr>
<td>Type</td>
<td>From the <strong>Type</strong> drop-down menu, set the type of server (SQL or LDAP) to which the query statement is directed. <strong>This value cannot be edited later.</strong></td>
</tr>
</tbody>
</table>

4. Select **Next**.

5. Enter a SQL query statement or LDAP filter string.

6. To insert property tags that resolve to host property values, select **Add Tags**.
7. Scroll through the list or use the **Tag Filter** field to search for a specific tag. Select the tag and then select **OK**.

8. In the Content pane, select **Next**.

9. The **Cache period** defines how long the values returned by this query are valid and are kept in DEX’s memory. Within the cache period, queries are answered using the value stored in memory. Queries are not sent to the external server.

10. Select **Finish**. The query statement is added to the Query Statements list.

11. In the Data Exchange (DEX) pane, select **Apply**.

**Tips for SQL Query Statements**

Consider the following points when you write a SQL query statement:

- If your SQL syntax requires it, insert an apostrophe before and after the tag.
Avoid using the * syntax or other wildcards. These queries can generate significant unnecessary communication overhead. It is more efficient to specify column names that are used in host properties.

You can write a statement that retrieves information from several linked database tables.

Use REPLACE or regular expression (REGEXP) terms to translate from external database formats. For example, the following statements translate the colon-format MAC address supported by the Forescout platform to a dot-format MAC address (01.23.45.ab.cd.ef).

```
SELECT column FROM table
WHERE mac = REPLACE ('{eds_mac_fmt_colon}', ':', '.');
```

```
SELECT column FROM table
WHERE mac = REGEXP_REPLACE ({mac}, '(..)(..)(..)(..)(..)(..)', '\1.\2.\3.\4.\5.\6');
```

**DEX Plugin – Define Host Properties**

Host properties store information that the Forescout platform discovers for each endpoint. When you work with DEX, you create new Forescout host properties to hold data extracted by querying external servers. This makes retrieved data available for use in Forescout policies. Before you perform this procedure, see Appendix 1: Forescout Property and Data Types.

**Define a Host Property Based on Data from External Databases**

You can define host properties based on data from external databases.

**To create a host property:**

1. In the Data Exchange pane, select the SQL/LDAP tab and then select the Properties tab.
2. Select Add.
3. Configure the property values:

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Enter a name for the property. This name is used wherever the property is displayed in the Console, for example, in the Properties tree.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Tag (ASCII only)</td>
<td>Enter a unique text string using ASCII characters. The Forescout platform references the property using this unique identification string. You cannot use spaces in the string.</td>
</tr>
<tr>
<td>Description</td>
<td>Enter a description of the property. This description is used for the property in the Conditions dialog box.</td>
</tr>
<tr>
<td>Server</td>
<td>Select a server from the drop-down menu. Your selection is displayed in the <strong>DB Queries</strong> field at the bottom of the pane.</td>
</tr>
<tr>
<td>Query Statement</td>
<td>Depending on what is selected in this field, the Query Statement is populated and the details of the statement are displayed in the <strong>DB Queries</strong> field.</td>
</tr>
</tbody>
</table>
4. Select **Next**.

5. Select the type of property you want to create:
   - **Single property**: Contains one retrieved value. This is the default setting. To create a Single Property, go to step 6.
   - **List property**: Contains a list of unique values. To create a List property, go to step 6.
   - **Composite property**: Contains several retrieved database columns. To activate this property, a SQL server you must be set as the server in the General tab. To create a Composite property, go to step 8.
   - **Record Exists** property: Indicates that data was returned for a host. To create a Record Exists property, go to step 9.

6. To define a Single Value or List property:
   - From the **Data Type** drop-down menu, select the type of data the property contains. See Appendix 1: Forescout Property and Data Types for details.
   - In the **Column Name** field, specify the column or attribute that is mapped to the property. This column must be retrieved by the query statement.

    For a **Single** property, the query must return a single value for the field/attribute.
For a List property, the query can return multiple values for the field/attribute.

- The Query Statements field is Read-only and is a reflection of the Query Statement selection in the previous pane.

7. For a List property, select **Aggregate new values from each update** to retain existing values when the property is updated. New property values are appended to the list. If you clear this option, the entire list stored by the property is overwritten by each update.

Continue to step 10.

8. To define a Composite property:

   a. Select **Add**.
b. Configure the Sub-property parameters:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Enter the name of the composite property.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Enter a short description of its content.</td>
</tr>
<tr>
<td>Source DB Column</td>
<td>Enter the name of the column from the external database that is mapped to the property. This column must be retrieved by the query statement, and can contain a single value or multiple values.</td>
</tr>
<tr>
<td>Data Type</td>
<td>From the drop-down menu, select the type of data the property contains. See Appendix 1: Forescout Property and Data Types for details.</td>
</tr>
<tr>
<td>Create Inventory Key</td>
<td>Select this option to let administrators list hosts based on this field in the Asset Inventory view. When this option is selected, the property is displayed as an index key in the Views pane of the Asset Inventory view.</td>
</tr>
</tbody>
</table>

c. Select OK. The field is displayed in the composite property table.
d. Repeat these steps to define additional composite property fields.
e. In the Map Data pane, select **Aggregate new values from each update** to retain existing values when the property is updated. New property values are appended as a new row in the composite property. If you clear this option, the entire table stored by the property is completely overwritten by each update.

9. To define a Record Exists property, select **Record Exists**.

Select the Record Exists option if you only want to know of the existence of the data and not about the content of the data.

10. Select Next.
11. Define the following parameters to specify where the property is displayed in the Console.

<table>
<thead>
<tr>
<th><strong>Display Property in Inventory View</strong></th>
<th>Select this option to list this property in the Inventory.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>This field is enabled when <strong>Display Property in Inventory View</strong> is selected. Enter the title of the Inventory item for this property.</td>
</tr>
<tr>
<td><strong>Display Property in Host Profiles pane of Home View</strong></td>
<td>Select this option to list this property in the Profiles tab of the Home View.</td>
</tr>
<tr>
<td><strong>Display Property in the Assets Portal</strong></td>
<td>Select this option to display this property in the Assets Portal.</td>
</tr>
<tr>
<td><strong>Enable Track Changes</strong></td>
<td>(For Single, Composite and List properties only) Select this option to create a second, parallel <em>change</em> property under the Track Changes folder of the Properties tree. Use the change property in policies to identify changes in the property values retrieved from the external server.</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>This field is enabled when <strong>Enable Track Changes</strong> is selected. Enter a name of the Track Change for this property or accept the default name that is pulled from the Property Name field.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>This field is enabled when <strong>Enable Track Changes</strong> is selected. Enter a description describing the change in the selected property or accept the default description.</td>
</tr>
</tbody>
</table>
12. Select Finish. The property is added to the table in the Property tab.

When you create policy conditions, your properties are displayed in the Properties tree under the **DEX: External DB** folder. Related change tracking properties are displayed in the Track Changes folder.

- **The Forescout platform only retrieves data and updates values for properties that are referenced by an active policy.**

**Define a Property Based on an Existing Property**

You can define a property by duplicating an existing property and editing its settings as required.

**To duplicate a property:**

1. In the Data Exchange pane, select the SQL/LDAP tab, then select the External Web Services tab or the CounterACT Web Service tab, and then select the Properties tab.
2. Select an existing property, and select **Duplicate**.
3. The Edit Property wizard opens. Fields in this wizard are identical to the Add Property wizard, and contain the values of the existing property you selected.

   - **Property Name** field contains the following dummy value based on the existing property:
     
     \[
     \text{Copy of } <\text{existing\_property\_name}>
     \]

   - **Property Tag** field contains a unique string generated by the Forescout platform.

   *Edit these values to create unique identifiers for the new property.*

5. Select **OK**. The property is added to the table in the Properties tab.

**Test the Property**

The property test verifies the definition of the property and the query that uses it. Because the query uses Forescout tags, you must enter a property value for tags in the query statement when running a test.

**To test a custom host property:**

1. In the Data Exchange pane, select a Properties tab.
2. Select an item and then select **Test**.
3. The field names in the Tag values dialog box correlate to the query statement selected within that property. Enter test values for the property tags and then select OK.

Using the property tag values you enter, the Forescout platform submits a test query statement referenced by the new host property to the external server.

4. Verify that the correct data is retrieved, and that the new host property is assigned the correct value from the retrieved data.

Map External Data to Host Properties

To map information submitted through the CounterACT Web Service to host properties, see Create a Forescout Host Property Based on the CounterACT Web Service.

To assign values to properties based on external database information, you map the results of a query statement to the property. The query statement is submitted to the external server, and determines the data fields that are retrieved. The query statement must return single, list, or composite data values corresponding to the target host property. For example:

- To populate a Single property, use a query statement that retrieves a single field or attribute value from the external server. If you specify a query that returns multiple values, the property is evaluated as irreolvable.

- To populate a List property, use a query that retrieves several rows from a column of a SQL database, or an LDAP attribute with multiple values.

The Forescout platform limits the number of values that can be retrieved for List or Composite properties. When the query statement retrieves a column or attribute with more values than this maximum, the Forescout platform evaluates the corresponding property as irreolvable. Define the Maximum Values Retrieved setting in the General Settings tab to control this limit. See DEX Plugin – Manage Communication with Database Servers for details.

About Query Statements

A Query Statement is a set of SQL statements or User Directory filter strings that the Forescout platform submits to the external server to retrieve host information.

Query statements are server independent. The same query statement can be used with several different external servers that use the same query syntax.

To increase efficiency, it is recommended to compose one query statement that supports several Forescout properties. You define a query statement that returns several related data fields, and map them to different Forescout properties.
One Query Statement, Many Endpoints

The Forescout platform maps the data retrieved by the query to host properties. Typically, data is retrieved for each host, and is mapped to the host properties of that host. This means that each query sent to the external server must select data for a single host, based on a unique key value, such as the MAC or IP address.

To do this, the query statement must include at least one property tag that specifies host-specific key values. Property tags are placeholder parameters that represent endpoint property values. When the Forescout platform evaluates the query statement string, it replaces the tags with actual property values.

For example, the following SQL statement uses the \{ip\} property tag:

```
select NAME, PHONE from MGRS where IP='\{ip\}'
```

To retrieve data, the Forescout platform submits multiple queries, replacing the \{ip\} property tag with the IP address of each host:

```
select NAME, PHONE from MGRS where IP='120.40.40.12'
select NAME, PHONE from MGRS where IP='100.30.30.10'
```

Similarly, the following LDAP filter uses the \{hostname\} property tag:

```
(name=\{hostname\})
```

To retrieve data, the Forescout platform submits a query for each host, replacing the \{hostname\} property tag with the host name of the host:

```
(name=QA_TOSHIBA_1)
(name=SALES_LAPTOP2132)
```

Data is retrieved for each host and can be mapped to the properties of that host.

Refer to "Property Tags" in the Forescout Administration Guide.
You cannot use tags that resolve to lists as index keys.

Because the MAC address is often used as an index key, DEX provides property tags to support additional MAC address formats:

- `{eds_mac_fmt_colon}` resolves to the MAC address in colon-separated format.
- `{eds_mac_fmt_dash}` resolves to the MAC address in dash-separated format.
- `{eds_mac_lc_fmt_colon}` resolves to the MAC address in lowercase colon-separated format.
- `{eds_mac_lc_fmt_dash}` resolves to the MAC address in lowercase dash-separated format.
- `{eds_mac}` resolves to the MAC address in uppercase.

Query Statements Without Endpoint Values

In some cases, you may want to compose a query statement that does not use a host property as an index key, for example, to verify connectivity or return an external database parameter that is not endpoint-specific for use in a policy condition. These queries return a uniform value for all hosts.

The Forescout platform only retrieves data and updates values for properties that are referenced by an active policy.

DEX Plugin – Import and Export Definitions

It is sometimes useful and more efficient to copy and edit existing query statements or property definitions. For example:

- To quickly duplicate settings on all CounterACT devices
- To retrieve similar information from Directory Servers with slightly different naming conventions

You can export query statements and properties, edit them, and import the new definitions.

The Forescout platform uses a simple XML format to represent query statements and property definitions:

- Each query statement or property is represented as an XML element. Attribute flags correspond to data fields and options you set when you defined the statement or property.
- A composite property is represented as an XML element containing sub-property child elements.
- Each property and sub-property is assigned an internal label used by the Forescout platform.

Remember to import and export SQL server definitions specified by the properties you want to work with.
To export definitions:

1. In the Data Exchange pane, select the SQL/LDAP, External Web Services or CounterACT Web Service tab and then select a sub-tab.

2. (Optional) To export a subset of defined elements, select them in the list. For example, in the SQL/LDAP tab, you can select only LDAP query statements for export.


4. To encrypt data, enter the password. Confirm the password and then select OK. The Export Table dialog box opens.

5. (Optional) Edit the default path name. If you are exporting a subset of the defined elements, select the Selected rows only option.

6. Select OK. The elements are exported to the specified file.

To import definitions:

1. In the Data Exchange pane, select the SQL/LDAP, External Web Services or CounterACT Web Service tab.

   If you are importing properties and their related query statements, first import the SQL servers and query statements referenced by the properties and apply changes. Next, import the properties.

2. Select Import. The Import dialog box opens.

3. Browse to and select the source XML file you want to import, and then select OK.

4. Elements of the type listed in the selected tab are imported from the source file. They are added after existing elements of the list. When imported elements have the same name as existing items, the label (duplicate n) is appended to their name.

5. Select Apply. New imported elements are created, and imported changes are applied to existing elements.

   Apply changes to query statements before you import properties.
DEX Plugin – About Update Statements

An Update Statement is a set of SQL statements that the Forescout platform sends to the external server to modify the external database. You specify an update statement when you use the DEX Update External Database action in a policy.

Typically, update statements write host-specific information to the external database. This means that each statement sent to the external server must reference a single host, based on a unique key value, such as the MAC or IP address. To do this, the update statement must use at least one Forescout property tag to specify host-specific key values.

For example, the following SQL statement uses the {mac} and {user} property tags:

```
INSERT INTO mactable VALUES('{mac}','{user}');
```

When this update statement is used in a policy, the Forescout platform submits multiple statements to the external database. For each host that matches the policy conditions, the Forescout platform creates a statement by replacing the property tags with the MAC address and user name of the host:

```
INSERT INTO mactable VALUES('00:A0:C9:14:C8:29','admin');
```

The update statement modifies the external database record for the specified host. See Update an External SQL Database for details.

DEX Plugin – Manage Communication with Database Servers

Policies that interact with external database and directory servers can generate significant communications between the Forescout platform and external servers. DEX provides tools and options that let you monitor traffic with external servers, and tune behavior to match available bandwidth and resources.

You can also use standard Forescout options, such as policy recheck schedules, to control the volume of communication with external servers.

- For tools to control interaction with external servers, see General Tools.
- For tools to control traffic for specific queries or servers, see Tools to Manage Individual Entities.

General Tools

The tools described in this section apply to all query/update communication with external servers using DEX. These general tools are available on the General Settings tab of the Data Exchange pane.

- To use the JDBC statistics log, select Log JDBC Statistics.

  CounterACT devices record external server interactions in the following file: `usr/local/forescout/log/plugin/eds/stats.log`

  New records are appended to any existing records in the file.

  To stop recording JDBC statistics, clear the Log JDBC Statistics option.
Set the following options to control communication with external servers:

- The **Maximum Values Retrieved** setting limits the results the Forescout platform accepts in response to a query statement. When the query statement retrieves a column or attribute with more values than this maximum, the Forescout platform evaluates the corresponding property as *Unresolvable*.

- The **Timeout** setting determines how long the Forescout platform waits after submitting a request to the external server. If the external server does not respond within the timeout period, the Forescout platform evaluates related host properties as Irresolvable. If **Log JDBC Statistics** is selected, an error is recorded for this session in the log.

**Tools to Manage Individual Entities**

Use these tools to fine-tune traffic to a specific external server, or traffic generated by an individual query statement.

- To control traffic for a specific external server, select the SQL Servers tab. Select the server, and select **Edit**. Modify the **Maximum Requests** and **Maximum Buffered Requests** settings to limit the volume of requests from the Forescout platform to the SQL server. See *DEX Plugin – Define Connections to External Databases* for details.


- To control traffic for a specific query statement, select the Query Statement tab. Select the query statement, and select **Edit**. Modify the **Cache** setting to change the validity period for values retrieved with that query. This setting affects how often the Forescout platform sends the query to the external server. See *DEX Plugin – Define LDAP or SQL Query Statements* for details.

**The JDBC Statistics Log**

DEX uses the Java Database Connectivity API to implement connectivity with external servers. Using the JDCB log option, you can record all requests the Forescout platform submits to external servers. This information is useful in fine-tuning the performance.

The Forescout platform records the following data for each request submitted to external servers:

- The type of request submitted to the server. Valid values include:
  - **QUERY**: A query statement was submitted to retrieve information.
  - **UPDATE**: An update statement was submitted to write information.
• The name of the query or update statement.
• The name of the property that caused the interaction. The Forescout platform submitted the query request to resolve the value of this property. This field is not defined for update interactions.
• The time when the request was sent to the external server, in Unix epoch time format.
• The time when the complete results were received from the external servers, in Unix epoch time format.
• The number of rows/values in the request. For a query statement, this is the number of rows or values received. For an update statement, this is the number of rows inserted or updated.
• Total time to process the statement, measured as the elapsed time between submission and response.
• Bytes received in response to a query statement, up to the maximum row/value cutoff. This field is not defined for update interactions.
• Text of the SQL/LDAP statement as it was submitted to the external server, after property tags are resolved to actual host values.
• JDBC URL: The request as it was transmitted in URL format.

The following example shows typical log entries.

<Server_IP>:/<Port>/<User_Name> is written as a substitute for the actual value that was in the example.
Work with External Web Services

Data exchange between the Forescout platform and external APIs and web services is a REST interaction based on HTTP messaging.

Exchanged data typically uses an XML or JSON data structure. When you submit a request message to retrieve data, the returned payload is parsed to yield Forescout property values. When you submit a request message with data to an external service, the message header should conform to the required structure.

In addition to requests initiated by the Forescout platform, external platforms can submit REST messages to the CounterACT web service. See Work with the CounterACT Web Service for details.

Depending on the desired DEX Plugin Integration Scenarios, perform the following procedures:

- Define Requests to External Web Services
- Define Host Properties Based on Data from External Web Services

Define Requests to External Web Services

You can define an HTTP request message that the Forescout platform submits to an external web service. To let DEX support more REST APIs that expect a body in the request, you can configure the POST Method and the HTTP Request Body fields in REST format. You can also use a JSON Web Token or other standard tokens for authentication purposes.

Typically, this message requests structured data from the web service. The Forescout platform parses the returned payload to update a custom endpoint property with data from the web service.

Several optional settings let you tune the volume of requests that the Forescout platform sends to the web service and specify which CounterACT Appliance(s) contact the web service.

The maximum number of HTTP request messages that can be configured is 10; a cluster of the same HTTP request message is counted as one message.

To define an HTTP request message to an external web service:

1. In the Data Exchange pane, select the External Web Services tab and then select the Web Service Requests tab.
2. Select Add.
3. In the General pane, define the following parameters:

<table>
<thead>
<tr>
<th><strong>Request Name</strong></th>
<th>Enter the name of this request. For example: <em>My Request</em>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Enter a brief description of the purpose of this request, and its target service.</td>
</tr>
<tr>
<td><strong>HTTP Method</strong></td>
<td>Select the HTTP method for retrieving data from an external web service.</td>
</tr>
<tr>
<td></td>
<td>- <strong>GET</strong>: Retrieves information from the external web service. Using the same parameters gives the same results.</td>
</tr>
<tr>
<td></td>
<td>- <strong>POST</strong>: Creates or updates the external web service.</td>
</tr>
<tr>
<td><strong>URL (with request parameters)</strong></td>
<td>Enter the request URL. Typically, this is the URL exposed by the web service, plus query parameters. Because returned data is parsed into endpoint properties, the IP or MAC address should be an index used to select returned data. Select <strong>Add Tags</strong> to insert runtime variables that are replaced by the IP/MAC address of the endpoint. This occurs when the request is submitted. Example URL: <a href="https://123.123.123.123:3000/endpoints/%7Bip%7D">https://123.123.123.123:3000/endpoints/{ip}</a></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>HTTP Message Headers</strong></td>
<td>Enter the HTTP headers included in the message. Example message header: Authorization: Bearer {eds_ext_web_auth_token} Content-type: Application/json</td>
</tr>
</tbody>
</table>
| **HTTP Request Body** | Enter the actual HTTP request data. The default limit is set to 5 KB. Example request body: 

```json
{
  "time":{eds_time_epoch},
  "mac":{mac},
  "hostname":{nbtab}
}
```

| **No Record Returns HTTP Status** | Select the HTTP status code that this service returns when no record matches the query parameters. |
| **Tags** | To insert runtime variables, put the cursor in a text field and select **Add Tags**. These variables are replaced with:

1. Actual endpoint property values. Refer to "Property Tags" in the Forescout Administration Guide for details.
2. Non-endpoint-related information. These special tags are as follows:
   - `{eds_time_epoch}`: The current time in epoch format.
   - `{eds_time_utc}`: The current time in UTC format.
   - `{eds_ext_web_auth_token}`: The authorization token that refers to the authorization token selected in the Authentication step. This tag can be placed in URL/Headers/Body. Note: when the token is in the form of an HTTP cookie, there is no need to add this tag, because the HTTP request uses the cookie for authorization.
   - `{eds_mac}`, `{eds_mac_fmt_colon}`, `{eds_mac_fmt_dash}`, `{eds_mac_lc_fmt_colon}`, `{eds_mac_lc_fmt_dash}`: The MAC address in uppercase, colon-separated format, dash-separated format, lowercase colon-separated format, or lowercase dash-separated format.
   - `{eds_user_xxx}`, `{eds_user_xxx_password}`: The username and password to be used in external web service requests, URLs, and actions, where xxx is the user-defined username. |
4. Select **Next**.

5. In the Authentication section, define the following parameters:

<table>
<thead>
<tr>
<th><strong>No Authentication</strong></th>
<th>Select this option if you do not want to use a web service username/password or an Authorization Token.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use Basic Authentication Header</strong></td>
<td>Select this option if you want to use the basic authentication header.</td>
</tr>
<tr>
<td><strong>Web Service Username</strong></td>
<td>Enter the username for authorizing credentials for the request message. This is sent as part of the message header.</td>
</tr>
<tr>
<td><strong>Web Service Password Verify Password</strong></td>
<td>Enter the password for authorizing credentials for the request message that is sent as part of the message header. Re-enter the password to verify it.</td>
</tr>
<tr>
<td><strong>Use Authorization Token</strong></td>
<td>Select this option to use an Authorization Token, and then select the token name from the drop-down menu. The available Authorization Tokens are taken from the External Web Services tab, Authorization Token sub-tab.</td>
</tr>
</tbody>
</table>
**Validate Web Server Certificate**

By default, this option is not selected, meaning that DEX always sets up the connection, regardless of whether the server certificate is a common commercial Certificate Authority issued certificate. If you select this option, you are adding certificate validation.

- If the server certificate is a common commercial Certificate Authority issued certificate, DEX sets up the connection to the server.
- If the server certificate is not a common commercial Certificate Authority issued certificate, DEX does not set up the connection to the server.

6. In the Proxy section, define the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use Proxy</strong></td>
<td>Select this option to access the web service via a proxy.</td>
</tr>
<tr>
<td><strong>Proxy IP Address</strong></td>
<td>Enter the IP address of the proxy server.</td>
</tr>
<tr>
<td><strong>Proxy Port Number</strong></td>
<td>Specify the port on the proxy server to which the HTTP request is sent.</td>
</tr>
<tr>
<td><strong>Proxy User name</strong></td>
<td>Enter the username for authorizing the Forescout platform to access the proxy server.</td>
</tr>
<tr>
<td><strong>Proxy Password</strong></td>
<td>Enter the password for authorizing the Forescout platform to access the proxy server. Re-enter the password to verify it.</td>
</tr>
</tbody>
</table>

7. Select **Next**.

8. By default, all CounterACT devices interact directly with the external web service. Each Appliance submits a request message to the service when it must resolve a property that contains web service data for an endpoint it manages. When all CounterACT devices can access the web service in the network, this is the preferred working method.

Sometimes, it may be necessary to route request messages through a designated CounterACT device. For example:
When some Appliances cannot directly contact the web service.
To optimize the traffic between the Forescout platform and the web service.

In such cases, select **Use Connecting CounterACT Device**. From the **Connecting CounterACT Device** drop-down menu, select the CounterACT device that submits this request message to the external web service. This device handles all communication with the external service. It forwards requests submitted to it by the other CounterACT devices, and it returns responses to CounterACT devices.

*It is a best practice to assign a maximum of five URLs to one connecting CounterACT device.*

You can define clusters of CounterACT devices that use a different connecting device. For example, CounterACT devices in each region or network segment can send requests through a local CounterACT device. See **Define a Cluster of CounterACT Devices** for details.

9. Select **Next**.

10. Define the following parameters:

| **Maximum Requests Device Submits per** | Select the maximum number of instances of this web service request message that each CounterACT device submits to the external web service during the specified time interval (second, minute, hour, or day). |
| **Maximum Buffered Requests** | Select the maximum number of request messages that are queued when the CounterACT device is already handling the maximum number of concurrent requests. |

11. Select **Finish**. The request is added to the table in the Web Service Requests tab.
Authentication Tokens

You can configure DEX to integrate with the APIs of third-party solutions that implement JSON web tokens or other standard tokens. This lets you integrate with third-party web services.

To add an authentication token:

1. In the Data Exchange pane, select the External Web Services tab and then select the Authentication Tokens tab.

2. Select Add.
3. Define the following parameters:

<table>
<thead>
<tr>
<th><strong>Name</strong></th>
<th>Enter the name of your Authentication Token.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Enter a brief description of the authentication token, if needed.</td>
</tr>
<tr>
<td><strong>Token Expiry (in minutes)</strong></td>
<td>Select the number of minutes the token lasts after it is issued. The minimum is 10 minutes (default). The frequency at which DEX refreshes the token is 3 minutes less than what is configured in this field. In practice, there is a delay between the time that DEX sends a request using the token and the time the web service begins processing the request.</td>
</tr>
<tr>
<td><strong>Method</strong></td>
<td>Select the HTTP method for retrieving data from an external web service.</td>
</tr>
<tr>
<td></td>
<td>- <strong>GET</strong>: Retrieves information from the external web service. Using the same parameters gives the same results.</td>
</tr>
<tr>
<td></td>
<td>- <strong>POST</strong>: Creates or updates the external web service.</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td>Enter a URL to submit a request for a JSON Web Token or other standard token to be generated. For example: https://{company IP}/companyapi/login</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Headers</strong></td>
<td>Enter the HTTP headers included in the message.</td>
</tr>
<tr>
<td><strong>Body</strong></td>
<td>Enter the actual HTTP request data.</td>
</tr>
<tr>
<td><strong>Use Basic Authentication Header</strong></td>
<td>Select this option if you want to use the basic authentication header.</td>
</tr>
<tr>
<td><strong>Web Service Username</strong></td>
<td>Enter the username for authorizing credentials for the request message. This is submitted as part of the message header.</td>
</tr>
<tr>
<td><strong>Web Service Password Verify Password</strong></td>
<td>Enter the password for authorizing credentials for the request message that is submitted as part of the message header. Re-enter the password to verify it.</td>
</tr>
<tr>
<td><strong>HTTP Response Field</strong></td>
<td>Select the field that the HTTP Response should look up for the authorization token.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Headers</strong>: The authorization token is in one of the headers.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Entire Content</strong>: The authorization token is the entire content.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Content: XML</strong>: The authorization token is in content that is in XML format.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Content: JSON</strong>: The authorization token is in content that is in JSON format.</td>
</tr>
<tr>
<td></td>
<td>- <strong>In Cookie</strong>: The Authorization token is in the form of an HTTP cookie.</td>
</tr>
<tr>
<td><strong>Keyword/Parser</strong></td>
<td>Enter the keyword or parser needed to extract the token from the HTTP response based on the HTTP Response Field.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Headers</strong>: The name of one header that indicates the authorization token.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Entire Content</strong>: Parser is not required.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Content: XML</strong>: The parsing pattern of XML Path. See Appendix 5: External Web Service Parser Construction for details.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Content: JSON</strong>: The parsing pattern of JSON Path. See Appendix 5: External Web Service Parser Construction for details.</td>
</tr>
<tr>
<td></td>
<td>- <strong>In Cookie</strong>: Parser is not required.</td>
</tr>
</tbody>
</table>

4. Select **OK**.

- The proxy configuration that DEX uses for the token refreshment call depends on the external web service request that uses the token. If multiple external web service requests use the same token but different proxies, upon refreshing the token, the proxy for the token refreshment call is based on the most recent external web service request that used the token.
Import Authentication Token

You can import your own XML-based Authentication Token.

1. In the Data Exchange pane, select the External Web Services tab and then select the Authentication Tokens tab.
2. Select Import. The Import dialog box opens.
3. Select Browse and select the XML file to run on web services.
4. Select OK. The file is imported and is displayed in the Authentication Tokens tab.

Export Authentication Token

If you want to duplicate the configuration of a DEX setup, you can export the Authentication Token.

1. In the Data Exchange pane, select the External Web Services tab and then select the Authentication Tokens tab.
2. Select Export. The Import/Export Password dialog box opens.
3. Enter the password in the Password and Confirm Password fields, and then select OK. The Export Table dialog box opens.
4. Select Browse to and select a location for the XML file. Select Open.
5. Select OK. In the Export Table dialog box, select OK.
6. If prompted to create a directory on the computer, select Yes.

The XML file is exported.

Usernames and Passwords as Tags

You can configure usernames and passwords as tags to use within DEX. These tags can be used in external web service requests, URLs, and actions. Multiple usernames and passwords can be added as tags.

The benefits of using tags are:

- Passwords do not appear as clear text in the DEX configuration.
- Usernames and passwords do not need to be entered multiple times in Web Service Requests. Define them once and then use the tags multiple times.

To add usernames and passwords as tags:

1. In Options, Data Exchange pane, select the External Web Services tab and then select the Users tab.
2. Select **Add**.

3. Define the following parameters:

<table>
<thead>
<tr>
<th><strong>Username</strong></th>
<th>Enter the username.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Password</strong></td>
<td>Enter the password. An empty password is not allowed.</td>
</tr>
<tr>
<td><strong>Verify Password</strong></td>
<td>Re-enter the password to verify it.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>(Optional) Enter a description of the username or password.</td>
</tr>
</tbody>
</table>

4. Select **OK**. The new user is displayed in the Data Exchange (DEX) pane.
5. Select **Apply**.

6. To add another user, select **Add** and repeat the steps.

7. To edit a username, password, or description, select a defined user and then select **Edit**.

8. To remove a user, select a defined user and then select **Remove**.

**Example 1: Use Tag in Web Service Request**

**To use a username and password tag in a Web Service Request:**

1. In **Options**, Data Exchange pane, select the External Web Services tab and then select the Web Service Requests tab.

2. Select **Add**.

3. Put the cursor in a text field to activate **Add Tags**.

4. Select **Add Tags**.
5. In **Tag Filter**, you can search on either **Name** or **Tag**:
   - Type the username you defined, or
   - Type `eds_user`

![Tag Filter screenshot](image)

6. The tags for the username and password you defined are in the list. They are named `eds_user_<username>` and `eds_user_<username>_password`.

7. Select the tags that you want to use in the Web Service Request and select **OK**.

8. Complete the configuration of the Web Service Request and select **Next**. See [Define Requests to External Web Services](#) for details.

**Example 2: Use Tag in Action**

**To use a username and password tag in an action:**

1. In the Console, select **Home**.

2. In the All Hosts pane, right-click an endpoint and select **Audit > DEX Send Web Service Request**.

![All Hosts screenshot](image)

3. Put the cursor in a text field to activate **Add Tags**.

4. Select **Add Tags**.

5. In **Tag Filter**, search on either **Name** or **Tag**:
   - Type the username you defined, or
   - Type `eds_user`
6. The tags for the username and password you defined are in the list. They are named eds_user_<username> and eds_user_<username>_password.

7. Select the tags that you want to use in the action and select **OK**.

8. Complete the configuration of the action and select **OK**. See [DEX Send Web Service Request Action](#) for details.

**Define a Cluster of CounterACT Devices**

Each CounterACT device must query the external web service to resolve a property that contains web service data for an endpoint it manages. Request messages can be routed from CounterACT devices to the web service in several ways:

- All CounterACT devices interact directly with the external web service. When all CounterACT devices can access the web service in the network, this is probably the simplest, and most preferred, method.

- Define a single, default Connecting CounterACT Device for the request message. This device handles all communication with the external service. It forwards requests submitted to it by the other CounterACT devices, and it returns responses to CounterACT devices. You specify this device when you create or edit the request message. All CounterACT devices are automatically assigned to this connecting device.

- In addition to defining a single Connecting CounterACT Device, you can define one or more clusters of CounterACT devices that route request messages through the cluster's Connecting CounterACT device.

To define a cluster:

- Specify a group of CounterACT devices.
- Specify a Connecting CounterACT Device for the cluster.

All devices in the cluster route request messages through the cluster's Connecting CounterACT Device. For example, CounterACT devices in each region or segment can send requests through a local CounterACT device.

This section describes how to define CounterACT device clusters to handle web service request messages.
Before you perform this procedure, you must define a default CounterACT Connecting Device for the request message, as described in Define Requests to External Web Services. If not assigned to a cluster, devices route their request messages through this default Connecting Device.

To define a cluster of CounterACT devices:

1. In the Data Exchange pane, select the External Web Services tab and then select the Web Service Requests tab.

2. Select an existing web service request and then select Cluster. The Add Connecting CounterACT Device Cluster dialog box opens.

   This dialog box duplicates the Web Service Request definition and adds the suffix _cluster to the Request Name. Only the following fields can be edited:
   
   – The Description field in the General tab. It is recommended that the word "cluster" be used in the description field.
   – The fields of the CounterACT Devices tab, which are used to specify the CounterACT device that handles the request.

3. Select the CounterACT Devices tab.
4. **Use Connecting CounterACT Device** is selected by default. Define the cluster in the following fields:

<table>
<thead>
<tr>
<th>Connecting CounterACT Device</th>
<th>Select the CounterACT device that handles this request message for all devices in the cluster. Only this device connects to the external web service. It forwards requests submitted to it by the other members of the cluster, and it returns the responses it receives from the external web service.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available Devices</td>
<td>To specify the CounterACT devices that submit this request message through the specified connecting device:</td>
</tr>
<tr>
<td>Selected Devices</td>
<td>1. Select an item in the Available Devices list.</td>
</tr>
<tr>
<td></td>
<td>2. Select <strong>Add</strong>. The selected device is displayed in the Selected Devices list.</td>
</tr>
<tr>
<td></td>
<td>Devices in the Selected Devices list send the request message to the cluster’s connecting device. They do not interact directly with the external web service, and they do not use the default Connecting Device defined for the request message.</td>
</tr>
</tbody>
</table>

5. Select **OK**. The new request message is displayed in the Web Service Requests tab.

### Define Host Properties Based on Data from External Web Services

Properties store information that the Forescout platform discovers for each endpoint. You can create new Forescout properties to hold data retrieved from external services. This makes retrieved data available for use in Forescout policies.

Before you perform this procedure, define a request message that retrieves structured data from an external web service. This data is parsed to populate the property you create.

See **Appendix 1: Forescout Property and Data Types** for details.

**To create a property from external web service data:**

1. In the Data Exchange pane, select the External Web Services tab and then select the Properties tab.

2. Do one of the following:
   - To create a property based on an existing property, select the already defined property, and then select **Duplicate**. The Forescout platform creates a copy of the property in the Properties tab. Select the duplicate, and then select **Edit**.
   - To create a new property, select **Add**. The General pane of the Add Property from External Web Service wizard opens.
3. Define the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property Name</strong></td>
<td>Enter the name of the property.</td>
</tr>
<tr>
<td><strong>Property Tag (ASCII only)</strong></td>
<td>Enter the internal runtime tag of the property. The Forescout platform references the property using this unique identification string. Do not use spaces in the string.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Enter a brief description of the purpose of this property, and its source web service.</td>
</tr>
<tr>
<td><strong>Property Requires Host Access</strong></td>
<td>Select this option if the property requires Host Access. For this property to be resolved, the Forescout platform requires access to the endpoint.</td>
</tr>
<tr>
<td><strong>Web Service Request</strong></td>
<td>Enter the name of the web service request that retrieves the data contained in this property.</td>
</tr>
</tbody>
</table>

4. Select **Next**.
5. Select the type of property you want to create:
   - **Single Value Property** contains one retrieved value.
   - **List Property** contains a list of unique values.
   - **Composite Property** contains a flat record of several data types, similar to a database row. To define a Composite property, continue to step 7.
   - **Record Exists** properties contain a Boolean value that indicates whether data was returned for this endpoint. Select this option if you only want to know that the data exists, not what the data content is.

6. To define a **Single Value Property** or **List Property**, define the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Type</strong></td>
<td>Select the type of data the property contains. See Appendix 1: Forescout Property and Data Types for details.</td>
</tr>
<tr>
<td><strong>Parse Data Using</strong></td>
<td>Select the type of format of the expression used to extract data from the payload returned by the external web service. The options are XML Path, JSON Path, and Regular Expression. See Parsing Pattern or Appendix 5: External Web Service Parser Construction for details.</td>
</tr>
</tbody>
</table>
| **Parsing Pattern**     | Enter the expression used to extract data.  
  - For XML Path, the Parsing Pattern could look like: `/response/result/serial_number/text()`  
  - For JSON Path, the Parsing Pattern could look like: `$.serial_number`  
  - Regular Expression, the Parsing Pattern could look like: `Begin(.*?)End`  

See Appendix 5: External Web Service Parser Construction for details.
Aggregate new values from each update

(Applies to List Property only) By default, the entire list is overwritten each time new data is received from the external web service. Select this option to add new, unique values to the existing list.

Continue to step 9.

7. To define a Composite Property:

a. Define the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parse Data Using</td>
<td>Select the format of the expression used to extract data from the payload returned by the external web service. The options are XML Path and JSON Path. See Appendix 5: External Web Service Parser Construction for details.</td>
</tr>
<tr>
<td>Aggregate new values from each update</td>
<td>By default, the entire record is overwritten each time new data is received from the external web service. Select this option to add new, unique records to the existing values.</td>
</tr>
</tbody>
</table>

b. Select Add.

c. Define the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field Name</td>
<td>Enter the name of the Sub-property.</td>
</tr>
<tr>
<td>Description</td>
<td>Enter a description of the data contained in this Sub-property.</td>
</tr>
<tr>
<td>Parsing Pattern</td>
<td>Enter the expression used to extract data. See Appendix 5: External Web Service Parser Construction for details.</td>
</tr>
<tr>
<td>Data Type</td>
<td>Select the type of data the property contains. See Appendix 1: Forescout Property and Data Types for details.</td>
</tr>
<tr>
<td>Create Inventory Key</td>
<td>Select this option to display the property as an index key in the Views pane of the Inventory view.</td>
</tr>
</tbody>
</table>

d. Select OK. The field is displayed in the composite property table. Repeat these steps to define additional fields of the composite property.

e. Continue to step 9.

8. To define a Record Exists Property:

Define the following parameters.
Parse Data Using | Select the format of the expression used to extract data from the payload returned by the external web service. The options are XML Path, JSON Path and Regular Expression. See Appendix 5: External Web Service Parser Construction for details.

Parsing Pattern | Enter the expression used to extract data.

9. Select **Next**.

10. In the Display/Track pane, specify where the property is displayed in the Console:

<table>
<thead>
<tr>
<th>Display Property in Inventory View</th>
<th>Select this option to show this property in the Inventory.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This field is enabled when <strong>Display Property in Inventory View</strong> is selected. Enter the title of the Inventory item for this property.</td>
</tr>
<tr>
<td>Display Property in Host Profiles Pane of Home View</td>
<td>Select this option to list this property in the Profiles tab of the Home view.</td>
</tr>
<tr>
<td>Display Property in the Assets Portal</td>
<td>Select this option to display this property in the Assets Portal.</td>
</tr>
<tr>
<td>Enable Track Changes</td>
<td>(For Single Value and List Properties only) Create a second, parallel change property under the Track Changes folder of the Properties tree. Use the change property in policies to identify changes in the property values retrieved from the external server.</td>
</tr>
</tbody>
</table>
**11.** Select **Finish**. The property is added to the table in the Properties tab.

When you create policy conditions, these properties are displayed in the Properties tree under the **DEX: External Web Service** folder. Related change tracking properties are displayed in the Track Changes folder.

- *The Forescout platform only retrieves data and updates values for properties that are referenced by an active policy.*

---

**Work with the CounterACT Web Service**

Data exchange between the Forescout platform and external APIs and web services is a REST/SOAP interaction based on HTTP messaging.

- *In addition to requests submitted to the CounterACT web service, the Forescout platform can initiate messaging to external platforms. See [Working with External Web Services](#).*

Depending on the desired **DEX Plugin Integration Scenarios**, perform the following procedures:

- **Define CounterACT Web Service Accounts**, Define authorization credentials that grant access to the CounterACT Web Service.
- **Define Host Properties from the CounterACT Web Service**, Define new Forescout host properties that hold data submitted via the CounterACT Web Service.
Define CounterACT Web Service Accounts

DEX supports the CounterACT Web Service, which lets external platforms and services interact with the Forescout platform by submitting web service request messages. Typically, the web server on the Enterprise Manager hosts the CounterACT Web Service.

Use this procedure to define authorization credentials for the web service. Request messages sent to the CounterACT Web Service must include valid account credentials.

See Appendix 4: Submit Data with the Forescout Web API for details.

Each account grants unique permissions in the web service. When you define custom web service properties, you associate each property with an account. To modify a property, the web service request message must use the authorization credentials of the account associated with the property. You are limited to ten external web service accounts.

To define a CounterACT web service account:

1. In the Data Exchange pane, select the CounterACT Web Service tab, and then select the Accounts tab.

2. Select Add.
3. Define the following parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Enter a name for the CounterACT web service account.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Enter a brief description of the purpose of this web service account.</td>
</tr>
<tr>
<td>Username</td>
<td>Enter the username for authorizing the Forescout platform to access this web service account.</td>
</tr>
<tr>
<td>Password</td>
<td>Enter the password for authorizing the Forescout platform to access this web service account.</td>
</tr>
</tbody>
</table>

4. Select OK. The account is displayed in the Accounts tab.

Web service clients include this login information in the web service request message header. Your login target for the web service is `username@account`. In the following example, the login target is `Servicedesk@Corporate Data Web Server`.

Define Host Properties from the CounterACT Web Service

Host properties store information that the Forescout platform discovers for each endpoint. When you work with the CounterACT Web Service, you create new Forescout host properties to hold data submitted by external platforms. This makes submitted data available for use in Forescout policies.

For more information about the CounterACT Web Service, see Appendix 4: Submit Data with the Forescout Web API. Also refer to the Forescout eyeExtend Connect Module: Web API Plugin Configuration Guide.

Before you perform this procedure, see Appendix 1: Forescout Property and Data Types for detailed information about Forescout host property data structures.

To create a property:

1. In the Data Exchange pane, select the CounterACT Web Service tab and then select the Properties tab.
2. Select Add.
3. Define the following parameters:

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Enter the name of the host property.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Tag (ASCII only)</td>
<td>Enter the internal runtime tag of the property. The Forescout platform references the property using this unique identification string. Do not use spaces in the string.</td>
</tr>
<tr>
<td>Description</td>
<td>Enter a brief description of the purpose of this property, and its source CounterACT web service.</td>
</tr>
<tr>
<td>Account</td>
<td>Select the name of the CounterACT web service account to which this property is mapped.</td>
</tr>
</tbody>
</table>

4. Select Next.
5. Select the type of host property you want to create.

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Value Property</td>
<td>Contains one retrieved value. To define a single value property, select the type of data the property contains from the <strong>Data Type</strong> drop-down list. See Appendix 1: Forescout Property and Data Types for details. Continue to step 6.</td>
</tr>
<tr>
<td>List Property</td>
<td>Contains a list of unique values. To define a List property, select the <strong>Aggregate new values from each update</strong> option to retain existing values when the property is updated. New property values are appended to the list. If you clear this option, the entire list stored by the property is completely overwritten by each update. Continue to step 6.</td>
</tr>
</tbody>
</table>
| Composite Property  | Contains several retrieved database columns. To define a Composite property:  
|                     | 1. Select **Add**. The Add Subproperty dialog box opens.  
|                     | 2. In the **Field Name** and **Description** fields, enter the field name of the composite property and a short description of its content.  
|                     | 3. From the **Data Type** drop-down menu, select the type of data the property contains. See Appendix 1: Forescout Property and Data Types for details.  
|                     | 4. Select the **Create Inventory Key** option to let administrators list hosts based on this field in the Inventory view. When this option is selected, the property is displayed as an index key in the Views pane of the Inventory view.  
|                     | 5. Select **OK**. The field is displayed in the composite property table.  
|                     | 6. Repeat these steps to define additional fields of the composite property.  
|                     | 7. In the Map Data pane, select the **Aggregate new values from each update** option to retain existing values when the property is updated. New property values are appended as a new row in the composite property. If you clear this option, the entire table stored by the property is completely overwritten by each update. |

6. Select **Next**.
7. In the Display/Track pane, specify where the property is displayed in the Console.

| **Display Property in Inventory View** | Select this option to show this property in the Inventory. In the **Description** field, enter the title of the Inventory item for this property. |
| **Display Property in Host Profiles Pane of Home View** | Select this option to list this property in the Profiles tab of the Home view. |
| **Display Property in the Assets Portal** | Select this option to show this property in the Assets Portal. |
| **Enable Track Changes** | (For Single Value and List Properties only) Select this option to create a second, parallel change property under the Track Changes folder of the Properties tree. Use the change property in policies to identify changes in the property values retrieved from the external server. |

8. Select **Finish**. The property is added to the table in the Property tab.

When you create policy conditions, your properties are displayed in the Properties tree under the **DEX: CounterACT Web Service** folder. Related change tracking properties are displayed in the Track Changes folder.
CounterACT Web Service Security Settings

This section describes security settings for the CounterACT web service.

A white list is used to grant access to the CounterACT web service. Perform the following procedure to let users contact the CounterACT web service and to define login protection settings.

To define security settings for the CounterACT web service:

1. In the Data Exchange pane, select the CounterACT Web Services tab, and then select the Security Settings tab.

2. A white list of IP addresses is used to grant access to the CounterACT web service. To add a network range:
   a. Select Add.
   b. Select All IPv4 or enter an IPv4 Range.
   c. Select OK. The IP range is displayed in the IP Address Range list.

3. To configure Authentication Security Settings, scroll to the bottom of the Security Settings tab. In the Authentication Security Settings section, set the parameters for account lock-out after a specific number of password failures.
4. In the Data Exchange pane, select **Apply**.

**Actions and Properties for Data Integration**

In addition to the custom properties you define, DEX provides the following actions and properties that support integration with external databases and servers. See:

- **DEX Update External Database Action**
- **DEX Send Web Service Request Action**

**DEX Update External Database Action**

Use the DEX Update External Database action to update values in an external database. This action sends an update statement you define to an external server you specify. For example, you can update a database table value from *not compliant* to *compliant* when a Forescout policy detects endpoint Antivirus compliance status.

When you use the DEX Update External Database action in a policy, the Forescout platform updates external database records for hosts that match the conditions of the policy.

Because update statements modify external databases, it is important to test the statement before you include it in a policy. Use this action as follows:

1. First, compose and verify update statements using the **Simulate** option of the action. The Forescout platform generates actual SQL statements *without submitting them to the external database*.

2. When the update statement works as desired, apply the action without the **Simulate** option. The Forescout platform generates actual SQL statements *and submits them to the external database*.

**To use the DEX Update External Database action:**

1. Using a text editor, plan the SQL statement you want to submit. Use dummy values or placeholders for host property values.

2. Create a policy with conditions that select endpoints for which you want to send the update statement.

3. Add a new action. In the Actions tree, expand the Audit group and select **DEX Update External Database**.
4. In the Parameters tab, configure the following:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEX DB Server</td>
<td>From the drop-down menu, select the target external server. The update statement is submitted to this server.</td>
</tr>
<tr>
<td>Use predefined SQL statement</td>
<td>Select this option to use a predefined SQL statement. From the drop-down menu, select a <strong>Predefined query name</strong>.</td>
</tr>
<tr>
<td>Define a new SQL statement for this action</td>
<td>Select this option to define a new SQL statement for this action.</td>
</tr>
<tr>
<td>Statement</td>
<td>Enter the SQL statement that the Forescout platform submits to the external database. Select <strong>Add Tags</strong> to insert Forescout property tags that resolve to host property values. For the index key of the query, you cannot use tags that resolve to a list.</td>
</tr>
<tr>
<td>Simulate</td>
<td>Select this option to generate SQL statements without submitting them to the external database.</td>
</tr>
</tbody>
</table>

5. Select **OK** and save the policy. Select **Apply**.

For each host that satisfies the conditions of the policy, the Forescout platform replaces property tags in the update statement with actual host property values. The result is the actual SQL statement that the Forescout platform would submit to the external server.

6. In the Views tree of the Home view, select **Policy** and go to the results of the simulated update statement. Select an endpoint that was discovered by the policy. To view the SQL statement generated for this host, do one of the following:

- Double-click the host to open the Host Details dialog box. Select the Policy Actions tab.
- Hover over the Actions column. When the tooltip is displayed, press `<F2>`.
If the generated SQL statement is very long, it may be truncated in tooltip or Policy Actions views.

7. Verify the update statement:
   – Check that Forescout tags resolved to host property values as expected.
   – Copy the SQL statement for a host and manually submit it to the external SQL server. Verify that the statement affects the database as desired.

If necessary, edit the update statement in the policy action and retest it.

8. When the update statement is finalized, edit the policy. Select the DEX Update External Database action and edit it. Clear the Simulate option and save the changes. The action generates SQL statements for each host selected by policy conditions and submits these statements to the external database.

When a host satisfies the conditions of the policy, the Forescout platform updates values in the corresponding record of the external database according to the statement defined for the action.

**DEX Send Web Service Request Action**

Use the DEX Send Web Service Request action to generate a raw HTTP message and send it to an external target. Use this action to submit request messages to external web services that use the REST protocol.

When you use this action in a policy, the Forescout platform submits a request message for each endpoint that matches the conditions of the policy. You can use Property Tags to include host-specific property values in the request message.

Test the request message thoroughly to verify that it triggers the desired response at the external service.

**To use the DEX Send Web Service Request action:**

1. In a policy, add a new action. In the Actions tree, expand the Audit group and select **DEX Send Web Service Request**.
2. In the Parameters tab, define the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Action Identifier</strong></td>
<td>Set the identifier that identifies requests submitted when this action is applied to an endpoint.</td>
</tr>
<tr>
<td><strong>Use predefined Web Service Request to fill in parameters</strong></td>
<td>Select this option to base the action on a request message you previously defined, as described in Define Requests to External Web Services. From the Request Name drop-down menu, select a pre-defined web service request. Parameters reflect settings of the web request.</td>
</tr>
<tr>
<td><strong>Define a new message for this action</strong></td>
<td>Define an HTTP request message from scratch.</td>
</tr>
</tbody>
</table>

3. In the HTTP Request section, define the following components of the HTTP message:

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HTTP Method</strong></td>
<td>Select the appropriate HTTP message method.</td>
</tr>
<tr>
<td><strong>Request URL</strong></td>
<td>Enter the URL of the target web service.</td>
</tr>
<tr>
<td><strong>HTTP Message Headers(raw)</strong></td>
<td>Enter any additional message headers in this area. To include endpoint-specific data, select Add Tags and add Property Tags that resolve to host property values when the message is created.</td>
</tr>
<tr>
<td><strong>HTTP Message Body</strong></td>
<td>Enter the body of the request message. To include endpoint-specific data, select Add Tags and add Property Tags that resolve to host property values when the message is created. Only add tags that reference Single Value properties. You cannot add tags that refer to list or composite properties.</td>
</tr>
</tbody>
</table>
### Upload Body as File

Select this option to append the contents of the Message Body field to the request message as a text file.

### Tags

To insert runtime variables, put the cursor in a text field and select **Add Tags**. These variables are replaced with:

1. **Actual endpoint property values.** Refer to “Property Tags” in the *Forescout Administration Guide* for details.
2. **Non-endpoint-related information.** These special tags are as follows:
   - `{eds_time_epoch}`: The current time in epoch format.
   - `{eds_time_utc}`: The current time in UTC format.
   - `{eds_ext_web_auth_token}`: The authorization token that refers to the authorization token selected in the Authentication step. This tag can be placed in URL/Headers/Body. Note: when the token is in the form of an HTTP cookie, there is no need to add this tag, because the HTTP request uses the cookie for authorization.
   - `{eds_mac}`, `{eds_mac_fmt_colon}`, `{eds_mac_fmt_dash}`, `{eds_mac_lc_fmt_colon}`, `{eds_mac_lc_fmt_dash}`: The MAC address in uppercase, colon-separated format, dash-separated format, lowercase colon-separated format, or lowercase dash-separated format.
   - `{eds_user_XXX}`, `{eds_user_XXX_password}`: The username and password to be used in external web service requests, URLs, and actions, where `XXX` is the user-defined username.

### 4. In the Authentication section, define the following parameters:

<table>
<thead>
<tr>
<th><strong>No Authentication</strong></th>
<th>Select this option if you do not want to use a web service username/password or an Authorization Token.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use Basic Authentication Header</strong></td>
<td>Select this option if you want to use the basic authentication header.</td>
</tr>
<tr>
<td><strong>Username</strong></td>
<td>Enter the username for authorizing credentials for the request message. This is submitted as part of the message header.</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>Enter the password for authorizing credentials for the request message that is submitted as part of the message header. Re-enter the password to confirm it.</td>
</tr>
</tbody>
</table>
1. Select this option to use an Authorization Token.

2. Select the token name from the drop-down menu. This Authorization Token is from the External Web Services tab, Authorization Token sub-tab.

 Validate Web Server Certificate

By default, this option is not selected, meaning that DEX always sets up the connection, regardless of whether the server certificate is a common commercial Certificate Authority issued certificate. If you select this option, you are adding certificate validation.

- If the server certificate is a common commercial Certificate Authority issued certificate, DEX sets up the connection to the server.
- If the server certificate is not a common commercial Certificate Authority issued certificate, DEX does not set up the connection to the server.

5. In the Proxy section, define the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use HTTP Proxy</td>
<td>Select this option to access the web service via a proxy.</td>
</tr>
<tr>
<td>Proxy IP Address</td>
<td>Enter the IP Address of the proxy server.</td>
</tr>
<tr>
<td>Proxy Port Number</td>
<td>Enter the port number on the proxy server where the HTTP request is sent.</td>
</tr>
<tr>
<td>Proxy Username</td>
<td>Enter the username for authorizing the Forescout platform to access the proxy server.</td>
</tr>
<tr>
<td>Proxy Password Confirm Proxy Password</td>
<td>Enter the password for authorizing the Forescout platform to access the proxy server. Re-enter the password to confirm it.</td>
</tr>
</tbody>
</table>

6. Select **OK** and save the policy. Select **Apply**.

For each host that matches the conditions of the policy, the Forescout platform replaces property tags in the request message with actual host property values and submits the request to the specified URL.

7. View the results of the action in the Home view. The HTTP return code and any message body returned by the external service are included in the status report for the action, as shown in the following example. The Forescout platform interprets all HTTP 2xx return codes as action success. All other HTTP return codes are interpreted as action failure.
### Host Details

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>putf</td>
</tr>
<tr>
<td>IPv4 Address</td>
<td>10.0.0.100</td>
</tr>
<tr>
<td>Hostname</td>
<td>ZTA</td>
</tr>
<tr>
<td>MAC Address</td>
<td>00:00:00:00:00:00</td>
</tr>
<tr>
<td>Domain</td>
<td>WORKGROUP</td>
</tr>
<tr>
<td>User Name</td>
<td></td>
</tr>
<tr>
<td>Inventory Number</td>
<td>99999999</td>
</tr>
<tr>
<td>Data Set</td>
<td>Yes</td>
</tr>
<tr>
<td>Record Date</td>
<td>7/29/07 11:50:22 AM</td>
</tr>
<tr>
<td>task:</td>
<td></td>
</tr>
<tr>
<td>30_sec</td>
<td>7/29/07 9:00:00 AM</td>
</tr>
<tr>
<td>task:</td>
<td></td>
</tr>
<tr>
<td>5_min</td>
<td>7/29/07 11:50:22 AM</td>
</tr>
</tbody>
</table>

### General

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>dex_external_webservice</td>
<td></td>
</tr>
<tr>
<td>boolean_retumed_registerList</td>
<td></td>
</tr>
<tr>
<td>dex_external_webservice</td>
<td></td>
</tr>
<tr>
<td>composite</td>
<td></td>
</tr>
<tr>
<td>mac_address</td>
<td>ZTA</td>
</tr>
<tr>
<td>customerName</td>
<td>ZTA</td>
</tr>
<tr>
<td>Inventory number</td>
<td>99999999</td>
</tr>
<tr>
<td>Data Set</td>
<td>Yes</td>
</tr>
<tr>
<td>Record Date</td>
<td>7/29/07 11:50:22 AM</td>
</tr>
<tr>
<td>task:</td>
<td></td>
</tr>
<tr>
<td>30_sec</td>
<td>7/29/07 9:00:00 AM</td>
</tr>
<tr>
<td>task:</td>
<td></td>
</tr>
<tr>
<td>5_min</td>
<td>7/29/07 11:50:22 AM</td>
</tr>
</tbody>
</table>

### More

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prop:</td>
<td></td>
</tr>
<tr>
<td>Composite_Sub_Time</td>
<td>7/29/07 1:54:06 AM</td>
</tr>
<tr>
<td>Prop:</td>
<td></td>
</tr>
<tr>
<td>Composite_Sub_Sint</td>
<td>5910</td>
</tr>
<tr>
<td>Prop:</td>
<td></td>
</tr>
<tr>
<td>Composite_Sub_Mac</td>
<td></td>
</tr>
<tr>
<td>Prop:</td>
<td></td>
</tr>
<tr>
<td>Composite_Sub_Decimal</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Appendix 1: Forescout Property and Data Types

Host properties store information that the Forescout platform discovers for each endpoint. When you work with DEX, you create new Forescout host properties to hold data extracted by querying external servers. This makes retrieved data available for use in Forescout policies.

You can create the following types of properties:

A **Single Value** property contains one value. For example:

- A string property that contains the GUID of the endpoint

A **List** property contains a list of unique values. All items in the list are the same type of data. For example, a list property can contain:

- A list of all users in a directory group
- A list of previous host logins

Single Value and List properties are represented in requests to CounterACT Web Service using the `<PROPERTY>` element and, optionally, the `<VALUE>` element. The following example specifies the value `sales` for the `Prop_String` property.

```xml
<PROPERTY NAME="Prop_String">
  <VALUE>Sales</VALUE>
</PROPERTY>
```
The following example specifies the Prop_Time property for a web transaction such as deletion. In this case, no <VALUE> element is needed:

```xml
<PROPERTY NAME="Prop_Time" />
```

For a List property, the <PROPERTY> element can contain several <VALUE> elements.

```xml
<PROPERTY NAME="Prop_List">
  <VALUE>1377526368</VALUE>
  <VALUE>1377663124</VALUE>
</PROPERTY>
```

*Composite* properties are like database tables, with several rows and columns. For example, a composite property can contain data from a help desk server listing recent service calls for a host. Retrieved columns would include:

- Date
- Contact
- Severity
- Status
- Description

Composite properties are represented in requests to CounterACT Web Service using the <TABLE_PROPERTY> and <ROW> elements. The <CPROPERTY> and <CVALUE> elements parallel the <PROPERTY> and <VALUE> elements used to describe non-composite properties. The following example describes a record in a composite property.

```xml
<TABLE_PROPERTY NAME="Prop_Composite">
  <ROW>
    <CPROPERTY NAME="Prop_Composite_Sub_Time">
      <CVALUE>1377526368</CVALUE> <!-- Epoch time -->
    </CPROPERTY>
    <CPROPERTY NAME="Prop_Composite_Sub_String">
      <CVALUE>SW_update</CVALUE>
    </CPROPERTY>
  </ROW>
</TABLE_PROPERTY>
```

*Track Changes* properties let you define policy conditions that identify changes in the value of custom properties you define. You can define track changes properties for Single Value, List, or Record Exists properties that you create.

You do not directly populate or modify Track Changes properties using the web service, but their values reflect web service activity. For example, if you define the web service property MoreAccountInfo, the corresponding Track Changes property, MoreAccountInfo Change, is updated each time you submit request messages that modify the base MoreAccountInfo property.
About Aggregate Properties

When you create list or composite properties for use with the CounterACT Web Service, select the **Aggregate new values from each update** option to create a property that retains existing values when the property is updated.

- For a List property, new property values submitted via the web service are appended to the existing list of values.
  - If this option is not enabled when the property is created, an *update* request message overwrites the entire list stored by the property.

- For a Composite property, new property values submitted via the web service are appended as new rows in the table.
  - If this option is not enabled when the property is created, an *update* request message overwrites the entire table stored by the property.

For both aggregate and non-aggregate properties, the *delete* request message deletes the entire list or table stored in the property.

About Data Types

Forescout host properties can contain various types of data. When you define a property, you specify the type of data that the property contains. This determines the available matching options when you use the property in a policy condition. For
example, the Forescout platform offers Segment and IP range options to match IP address values, and Older Than and Before options to match Date values.

The following table lists the data types your custom property can hold, and typical external data sources.

<table>
<thead>
<tr>
<th>Forescout Property Data Type</th>
<th>Typical Expected SQL Data Types</th>
<th>Typical Expected LDAP Syntaxes</th>
<th>Typical Expected External Web Service Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>CHAR, VARCHAR, TEXT, LONGTEXT, TINYTEXT, MEDIUMTEXT</td>
<td>Unicode String Distinguished Name</td>
<td>Plain text</td>
</tr>
<tr>
<td>MAC Address</td>
<td>Standard notation with period (.) or colon (:) separators</td>
<td></td>
<td>Plain text.</td>
</tr>
<tr>
<td>IP Address</td>
<td>Standard IPv4 notation: xxx.xxx.xxx.xxx</td>
<td></td>
<td>Plain text.</td>
</tr>
<tr>
<td>Integer</td>
<td>INTEGER</td>
<td>Integer Octet String</td>
<td>Plain text</td>
</tr>
</tbody>
</table>
| Date                        | NUMBER type with Epoch Time in seconds, or DATE, DATETIME, TIMESTAMP | UTC Coded Timestamp           | Plain Text.                                  
  To let DEX know the correct meaning of the plain text retrieved from the external web service that represents a date/time, select a date format when configuring the property. There are three available formats:
  - **Epoch time**: Also known as Unix/POSIX time.
  - **UTC format in the form of %Y-%m-%dT%H:%M:%SZ**: A date format according to the ISO-8601 with the time zone being UTC (Coordinated Universal Time). An example of date string matching this format is 2017-06-19T15:13:11Z
  - **Customized format**: See Format the Date for details.
| Boolean                     | TRUE, FALSE, BIT, BINARY         | Boolean                       | Plain text                                    |

**Format the Date**

When writing the date in customized format, the following tokens are available:

<p>| %a | Day of the week, abbreviated with only three characters (for example, Mon, Tue, Wed, Thu, Fri, Sat, Sun) |</p>
<table>
<thead>
<tr>
<th>Token</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%A</td>
<td>Day of the week, full name (for example, Monday)</td>
</tr>
<tr>
<td>%b</td>
<td>Month, abbreviated (for example, Jan)</td>
</tr>
<tr>
<td>%B</td>
<td>Month, full name (for example, January)</td>
</tr>
<tr>
<td>%d</td>
<td>Numeric day of the month (1-31). It is acceptable to have a leading zero before a single number, for example, 01.</td>
</tr>
<tr>
<td>%H</td>
<td>Hour, 24 hour clock (0-23)</td>
</tr>
<tr>
<td>%I</td>
<td>Hour, 12 hour clock (1-12)</td>
</tr>
<tr>
<td>%j</td>
<td>Day of the year (1-366). It is acceptable to have a leading zero before a single number, for example, 01.</td>
</tr>
<tr>
<td>%m</td>
<td>Month number (1-12). It is acceptable to have a leading zero before a single number, for example, 01.</td>
</tr>
<tr>
<td>%M</td>
<td>Minute (0-59)</td>
</tr>
<tr>
<td>%p</td>
<td>AM/PM or am/pm</td>
</tr>
<tr>
<td>%s</td>
<td>Seconds since the Epoch</td>
</tr>
<tr>
<td>%S</td>
<td>Seconds (0-59)</td>
</tr>
<tr>
<td>%y</td>
<td>Year (2 digits)</td>
</tr>
<tr>
<td>%Y</td>
<td>Year (4 digits)</td>
</tr>
<tr>
<td>%z</td>
<td>Time zone in format +/-0000</td>
</tr>
</tbody>
</table>

Example:
Consider a date string "10/01/12". You, as a user, are aware of its meaning and what you need to do is give the format of the string to DEX. DEX then understands the meaning of the string as well.

Assuming you interpret the string in this way:
- 10 is for year
- 01 is for month
- 12 is for day

When writing the format, refer to the available tokens. The following three tokens can serve the purpose:

<table>
<thead>
<tr>
<th>Token</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%d</td>
<td>Numeric day of the month (1-31). It is acceptable to have a leading zero before a single number, for example, 01.</td>
</tr>
<tr>
<td>%m</td>
<td>Month number (1-12). It is acceptable to have a leading zero before a single number, for example, 01.</td>
</tr>
<tr>
<td>%y</td>
<td>Year (2 digits)</td>
</tr>
</tbody>
</table>

At this point, the format is very simple to generate by replacing 10 with %y, 01 with %m and 12 with %d, and you get "%y/%m/%d" as the format of the original date string.

There is no need to replace the "/"s (they are just separators that have no meaning with respect to the date).
If the date string contains no time zone information, DEX considers the time zone as GMT+0000.

Appendix 2: Use Advanced JDBC Attributes

You can define additional connection attributes by directly editing the Java Database Connectivity (JDBC) connection string that the Forescout platform uses to define connection to SQL databases.

To use advanced JDBC attributes:

1. In the Data Exchange configuration pane, select the SQL/LDAP tab and select the DB Servers tab. Then select Add.

2. In the General pane of the Add/Edit SQL Server wizard, select Advanced. The Advanced server settings dialog box opens.

   The JDBC protocol represents the database connection as a URL string with attributes. The default URL string displayed reflects the database driver you selected and includes placeholder tags for the configured fields.

3. Edit the URL string to add optional configuration attributes. For example, the following string adds the SSL attribute to a connection that uses the PostgreSQL database driver.

   jdbc:postgresql://{hostname}:{port}/{dbname}?user={user}&password={passwd}&ssl=true

   Add only attributes supported by the database driver you specify. The Forescout platform does not validate these optional attributes.
4. Select **OK** to save the URL string and return to the wizard. Complete the server definition and apply changes.

**Appendix 3: Map Information from External Servers Example**

The following example demonstrates the creation of custom host properties based on an external SQL database. The external database includes the GUESTS_BY_MAC table, with the following columns:

- MAC – The Guest MAC-Address
- ID – The Guest ID
- APPROVE – Is the MAC approved as a guest (True/False)
- EXPIRATION_DATE – A date for expiration date

<table>
<thead>
<tr>
<th>MAC</th>
<th>ID</th>
<th>APPROVE</th>
<th>EXPIRATION_DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>001122aabbc</td>
<td>1001</td>
<td>TRUE</td>
<td>31/12/2012</td>
</tr>
</tbody>
</table>

1. Create the following query to retrieve data from the GUESTS_BY_MAC table:

   ```sql
   SELECT ID, APPROVE, EXPIRATION_DATE FROM dbo.GUESTS_BY_MAC WHERE mac='{mac}'
   ```

   The MAC Address tag `{mac}` is inserted in the query syntax to ensure that the query retrieves information for a specific host.

2. Create the following properties that contain data from columns in the GUESTS_BY_MAC table:

   - **GUESTS_BY_MAC – MAC Address Exists**
     This is a Result Exists property that indicates whether the MAC address exists in the GUESTS_BY_MAC table.
   - **GUESTS_BY_MAC – ID**
     This Single Value property contains the value of the ID column, formatted as a String data type.
   - **GUESTS_BY_MAC – APPROVE**
     This Single Value property holds a Boolean value based on the APPROVE column.
   - **GUESTS_BY_MAC – EXPIRATION_DATE**
     This Single Value property holds a Date data value based on the EXPIRATION_DATE column.

3. The **GUESTS_BY_MAC – ID** property you created from the GUESTS_BY_MAC table can, in turn, be used as a key to retrieve data from related data tables. For the data table GUESTS_BY_ID, the table columns are:

   - ID – The Guest ID
   - USER – The Guest user name
You can create a new query that uses the `GUESTS_BY_MAC – ID Column` property as the key field. Select **Add Tags** and add the property tag for the `GUESTS_BY_MAC – ID` property.

The query definition is:

```
SELECT USER FROM dbo.GUESTS_BY_ID WHERE ID='{eds_GUESTS_BY_MAC_ID}'
```

Define a property named `GUESTS_BY_ID – User Column` to store the retrieved value.

4. Create policies using the properties you defined. Policies are evaluated for a host based on values the Forescout platform retrieves for that host from the external database.

### Appendix 4: Submit Data with the Forescout Web API

The CounterACT Web Service lets external entities communicate with the Forescout platform using simple, yet powerful web service requests based on HTTP interaction.

The CounterACT Web Service lets you communicate with the Forescout platform using web service requests. You create web service requests that reference Forescout host properties and Forescout Property List values. These messages contain property information in an XML data body. The CounterACT Web Service parses the data to update Forescout host properties and Property List values.

- Refer to "Web Service Transactions for Data Retrieval" in the Forescout eyeExtend Connect Module: Web API Plugin Configuration Guide.

See the following topics:

- [Web Service Implementation Overview](#)
- [Web Service Interaction](#)
- [CounterACT Web Service Transactions](#)

### Web Service Implementation Overview

Typically, web service integration includes the following steps:

- For updating host properties: Plan and create custom host properties based on the structure of the data to be submitted to the Forescout platform. For details, see [Define Host Properties from the CounterACT Web Service](#).

- Properties and accounts are related and should be planned with the third-party user who will interact with the web service. The Forescout administrator can define several accounts for the web service. Each account allows modification of a specified set of host properties.
For updating Forescout Property List values: Plan and create custom Property Lists based on the structure of the data that will be submitted to the Forescout platform. For more information, refer to "Defining and Managing Lists" in the Forescout Administration Guide.

Each CounterACT Web Service account allows modification of all the Forescout Property Lists.

Create third-party login credentials for the web service. For details, see Define CounterACT Web Service Accounts.

Third-party users must receive accounts that let them modify the custom properties relevant to their integration.

Third-party users of the web service define and test the commands or routines that generate and submit messages to the web service. They can refer to this appendix for descriptions of the fields and attributes used in a typical web service request.

Some configuration steps must be performed on the Forescout platform to support integration. Work with your Forescout administrator on these steps.

**Web Service Interaction**

This topic provides an overview of the HTTP request and response messages exchanged by external platforms and the CounterACT Web Service.

**Request Messages**

All CounterACT Web Service requests use the same URI. The requested action is declared in the data section of the request. This means that the HTTP header is identical for all web service transactions.

The URI has the following format:

https://<EM.IP>/FSAPI/niCore/<Purpose>

where:

<EM.IP> is the IP address of the Forescout Enterprise Manager in the network. Verify that this target is accessible from any clients that send requests to the CounterACT Web Service.

<Purpose> is the purpose of the request. There are two options: Hosts for updating host properties and Lists for updating Forescout Property List values.

The following example shows the format of a typical request header submitted to the CounterACT Web Service. The Enterprise Manager IP address is 10.0.0.1.

```
POST https://10.0.0.1/fsapi/niCore/Hosts HTTP/1.1
Host: 10.0.0.1
Authorization: Basic aGFtZWVkQGZvcmVzY291dC5jb206aGFtZW291dC5jb20=
Accept: application/xml
Content-type: application/xml
```

Note the following:

- All requests to the CounterACT Web Service use secured HTTP (HTTPS).
- All requests to the CounterACT Web Service use the HTTP POST request method.
- All requests to the CounterACT Web Service use standard Basic authorization. Use the credentials provided to you by the Forescout administrator.
- Endpoint/Property List data is submitted to the CounterACT Web Service in XML format, therefore the following header is specified:
  
  ```
  Content-type: application/xml
  ```
- The CounterACT Web Service uses basic HTTP authorization.

The body of the request message represents task information in XML format. The following example calls the Update task, and writes a new value to the host property `Prop_String` for the endpoint with IP address `10.0.0.101`.

```xml
<FSAPI TYPE="request" API_VERSION="1.0">
  <TRANSACTION TYPE="update">
    <OPTIONS CREATE_NEW_HOST="true"/>
    <HOST_KEY NAME="ip" VALUE="10.0.0.101"/>
    <PROPERTIES>
      <PROPERTY NAME="Prop_String">
        <VALUE>Sales</VALUE>
      </PROPERTY>
    </PROPERTIES>
  </TRANSACTION>
</FSAPI>
```

The elements `<FSAPI>` `<TRANSACTION>` `<OPTIONS>` specify the requested task, target endpoint, and other processing options.

The nested elements `<HOST_KEY>` `<PROPERTIES>` `<PROPERTY NAME>` `<VALUE>` relate to the host properties modified by the task.

**Response Messages**

The CounterACT Web Service replies to request messages with a response message. The header of the message reflects the original request message header.

The body of the message has the following XML structure.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<FSAPI TYPE="response" API_VERSION="1.0">
  <STATUS>
    <CODE>FSAPI_OK</CODE>
    <MESSAGE>Successfully updated 1 properties for host ip=10.0.0.101</MESSAGE>
  </STATUS>
</FSAPI>
```

The `<STATUS>` element is an envelope for feedback information about submitted request messages.

The `<CODE>` element contains feedback codes that correspond to HTTP status codes.

<table>
<thead>
<tr>
<th>CODE: Valid Values</th>
<th>HTTP Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSAPI_OK</td>
<td>200</td>
<td>Requested tasks were completed successfully.</td>
</tr>
</tbody>
</table>
### CODE: Valid Values

<table>
<thead>
<tr>
<th>CODE</th>
<th>HTTP Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSAPI_BAD_SECRET</td>
<td>401</td>
<td>Authorization failed.</td>
</tr>
<tr>
<td>FSAPI_BAD_XML_SYNTAX</td>
<td>400</td>
<td>XML error in the body of the request.</td>
</tr>
<tr>
<td>FSAPI_BAD_REQUEST_STRUCTURE</td>
<td>400</td>
<td>The body of the request contained XML that did not correspond to expected data structure.</td>
</tr>
<tr>
<td>FSAPI_BAD_REQUEST_DATA</td>
<td>400</td>
<td>Data in a <code>&lt;VALUE&gt;</code> element could not be parsed.</td>
</tr>
<tr>
<td>FSAPI_NOT_IMPLEMENTED</td>
<td>501</td>
<td>The request specified a transaction type or feature that is not yet supported by the web service.</td>
</tr>
</tbody>
</table>

### Examples: Submit Request Messages with Curl

The examples in this guide use the popular curl utility to generate and submit request messages to the CounterACT Web Service. If curl is installed in your environment, you can copy, paste, and edit these examples.

**Example 1: To Update Host Properties**

The following example shows a curl command that can be used to create and submit the request message described above:

```bash
curl -u "\{username\}:\{account\}::{password}" -k \
-H "Content-Type:application/xml" -d @/tmp/update.xml \
-X POST https://\{EM.IP\}/fsapi/niCore/Hosts
```

Note the following:

- The entire command is a single line. Backslash characters `\` indicate line continuation, as in most Unix environments.
- Replace the `{username}` `{account}` and `{password}` placeholders with the web service account credentials you received from the Forescout administrator.
- Replace the `{EM.IP}` placeholder with the IP address of the Forescout Enterprise Manager that hosts the web service. In this guide, the placeholder value `10.0.0.1` is used.
- The previously generated `update.xml` file contains the message body. This file is appended to the HTTP POST message.
- The curl `-k` option allows connections to SSL sites without certificates.

The full interaction is shown below. The `update.xml` file is listed.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<FSAPI TYPE="request" API_VERSION="1.0">
  <TRANSACTION TYPE="update">
    <OPTIONS CREATE_NEW_HOST="true"/>
    <HOST_KEY NAME="ip" VALUE="10.0.0.101"/>
    <PROPERTIES>
      <PROPERTY NAME="Prop_String">
        <VALUE>Sales</VALUE>
      </PROPERTY>
    </PROPERTIES>
  </TRANSACTION>
</FSAPI>
```
The `update.xml` file is added to the body of an HTTPS POST message, which is submitted to the web service. The following statement uses the curl utility to generate and submit the message.

```
curl -u "user@account:password" -k -H "Content-Type:application/xml" 
    -d @/tmp/update.xml 
    -X POST https://10.0.0.1/fsapi/niCore/Hosts
```

The following response message indicates that the request succeeded.

```
<?xml version="1.0" encoding="UTF-8"?>
<FSAPI TYPE="response" API_VERSION="1.0">
    <STATUS>
        <CODE>FSAPI_OK</CODE>
        <MESSAGE>Successfully updated 1 properties for host ip=10.0.0.101</MESSAGE>
    </STATUS>
</FSAPI>
```

The updated value of the `Prop_String` property is displayed in the Home view of the Console.

---

**Example 2: To Update Forescout Property List Values**

The following example shows a curl command that can be used to create and submit the request message described in the previous section:
curl -u "\{username\}@\{account\}:\{password\}" -k \n  -H "Content-Type:application/xml" -d @/tmp/update.xml \n  -X POST https://\{EM.IP\}/fsapi/niCore/Lists

Note the following:

- The entire command is a single line. Backslash characters \ indicate line continuation, as in most Unix environments.
- Replace the \{username\} \{account\} and \{password\} placeholders with the web service account credentials you received from the Forescout administrator.
- Replace the \{EM.IP\} placeholder with the IP address of the Forescout Enterprise Manager that hosts the web service. In this guide, the placeholder value 10.0.0.1 is used.
- The previously generated update.xml file contains the message body. This file is appended to the HTTP POST message.
- The full curl –k option allows connections to SSL sites without certificates.

The updated lists, sales_employee_id and support_employee_id are displayed in Options > Lists.
CounterACT Web Service Transactions

Transactions are the tasks you perform using web service message requests. This topic lists the transactions provided by the CounterACT Web Service and describes how they are implemented.

Different API versions have different sets of transaction types. The request XML file should specify the correct API_VERSION to make the transaction work. Typically, when working with Forescout Property Lists, the API_VERSION should be 2.0.

<table>
<thead>
<tr>
<th>Transaction Type</th>
<th>API Version</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>update</td>
<td>√</td>
</tr>
<tr>
<td>delete</td>
<td>√</td>
</tr>
<tr>
<td>add_list_values</td>
<td></td>
</tr>
<tr>
<td>delete_list_values</td>
<td></td>
</tr>
<tr>
<td>delete_all_list_values</td>
<td></td>
</tr>
</tbody>
</table>

Update: Write Property Values to the Forescout Platform

The request message for the Update transaction specifies the following information:

- The target host, identified by its IP address.
- Host properties and corresponding values for this host.

The CounterACT Web Service updates the properties with the supplied values.

*The specified host properties must be defined in the Forescout platform before you submit the request message.*

In the CounterACT Web Service, the requested action is declared in the <TRANSACTION> element in the data section of the request. For update tasks, the Transaction Type attribute has the value update. This essential setting differentiates this request message from other tasks.

The data section of the message should reflect the structure of the referenced host property. See Appendix 1: Forescout Property and Data Types for details.

**Example: Basic Update**

The following example shows the basic request message to update a host property with a new value. The update.xml file is listed.

```xml
#!/tmp/update.xml
<?xml version="1.0" encoding="UTF-8"?>
<FSAPI TYPE="request" API_VERSION="1.0">
    <TRANSACTION TYPE="update">
        <OPTIONS CREATE_NEW_HOST="true"/>
        <HOST_KEY NAME="ip" VALUE="10.0.0.101"/>
        <PROPERTIES>
            <PROPERTY NAME="Prop_String">
                <VALUE>Sales</VALUE>
            </PROPERTY>
        </PROPERTIES>
    </TRANSACTION>
</FSAPI>
```
The following statement uses the `curl` utility to create and submit an HTTPS request message with the XML file in the message body.

```
curl -u "user@account:password" -k -H "Content-Type: application/xml" \
    -d @/tmp/update.xml \
    -X POST https://10.0.0.1/fsapi/niCore/Hosts
```

**XML Schema for the Update Transaction**

This section describes the following elements that can be used when you construct an Update request message.

- **Forescout API Element: Web Service Envelope**
- **TRANSACTION Element: Specify a Web Service Task**
- **OPTIONS Element: Task Options**
- **HOST_KEY Element: Identify the Target Endpoint**
- **PROPERTIES Element: Host Property Data Section**
- **PROPERTY Element: Specify a Single Value or List Host Property**
- **TABLE_PROPERTY, CPROPERTY, and ROW Elements: Specify a Composite Property**
- **VALUE and CVALUE Element: Host Property Values**

**Forescout API Element: Web Service Envelope**

The `<FSAPI>` element identifies the XML payload as CounterACT Web Service request content, and provides basic information about the request content.

Usage:

```xml
<FSAPI TYPE="request" API_VERSION="1.0">
```

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE</td>
<td>The type of web service message.</td>
<td>Request: web service request message</td>
</tr>
<tr>
<td>API_VERSION</td>
<td>The version of the API that is used to process the request.</td>
<td>1.0 (default)</td>
</tr>
</tbody>
</table>

**TRANSACTION Element: Specify a Web Service Task**

The `<TRANSACTION>` element specifies the action you want to apply to a specific host property. This element is the external envelope of the request message body, enclosing all other elements.

A single request message contains only one transaction element. This means it can perform one type of task using data relevant to a single endpoint.

You can submit several similar tasks for an endpoint in a single request message. For example, a single transaction can update several properties for one endpoint.

The `TYPE` attribute determines the action requested by the message. For the Update task, this attribute should be set to `update`, as in the following example:

```xml
<TRANSACTION TYPE="update">
```
OPTIONS Element: Task Options

Use the `<OPTIONS>` element to enable processing options for the transaction. Options are listed as attributes of the element, and can be independently enabled and disabled.

Usage:

```xml
<OPTIONS CREATE_NEW_HOST="true"/>
```

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE_NEW_HOST</td>
<td>If the <code>&lt;HOST_KEY&gt;</code> element specifies an endpoint not known to the Forescout platform, create a new endpoint based on the <code>&lt;HOST_KEY&gt;</code> value.</td>
<td>True/False Default: False</td>
</tr>
</tbody>
</table>

HOST_KEY Element: Identify the Target Endpoint

The `<HOST_KEY>` element specifies the endpoint to which the task is applied. The endpoint can be specified using its IP address.

Use one of the following statements:

```xml
<HOST_KEY NAME="ip" VALUE="10.0.0.101"/>
```

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>Type of identifier used to specify the endpoint.</td>
<td>IP: Host is identified by its IP address. The VALUE attribute must contain a valid IP address.</td>
</tr>
<tr>
<td>VALUE</td>
<td>Actual identifier value used to specify the endpoint.</td>
<td>IP address in dot-separated format: 10.0.0.101</td>
</tr>
</tbody>
</table>

PROPERTIES Element: Host Property Data Section

The `<PROPERTIES>` element is an envelope for the detailed host property information included in the message. In this section of the message, you specify the host property values that will be modified for the endpoint.

The request can contain only one `<PROPERTIES>` element, which must contain at least one `<PROPERTY>` or `<TABLE_PROPERTY>` child element.

When one request modifies several host properties for an endpoint, the `<PROPERTIES>` element contains several child `<PROPERTY>` or `<TABLE_PROPERTY>` elements. In the following example, the `<PROPERTIES>` element contains a composite property and a Single Value string property.

```xml
<PROPERTIES>
  <TABLE_PROPERTY NAME="Prop_Composite">
    . . .
  </TABLE_PROPERTY>
  <PROPERTY NAME="Prop_String">
    . . .
  </PROPERTY>
</PROPERTIES>
```
PROPERTY Element: Specify a Single Value or List Host Property

Use the `<PROPERTY>` element to specify a property to be modified. The `<PROPERTY>` element contains all modification information for a host property containing a single data value or a list.

For a Single Value property, the `<PROPERTY>` element contains a single `<VALUE>` element. If more than one `<VALUE>` element is specified, the last value is used for the Update task.

```xml
<PROPERTY NAME="Prop_Time">
  <VALUE>1377526368</VALUE> <!-- Epoch time -->
</PROPERTY>
```

For a list property, the `<PROPERTY>` element can contain several `<VALUE>` elements.

```xml
<PROPERTY NAME="Prop_List">
  <VALUE>1377526368</VALUE> <!-- Epoch time -->
  <VALUE>1377668342</VALUE> <!-- Epoch time -->
</PROPERTY>
```

- If the list property was defined as an aggregate property, existing values are retained when the property is updated. Property values submitted in the request message are appended as new values to the list.
- If the list property is not an aggregate property, the values in the request message completely replace the list stored by the property.

See Appendix 1: Forescout Property and Data Types for details.

TABLE_PROPERTY, CPROPERTY, and ROW Elements: Specify a Composite Property

When you work with a composite property containing multiple fields, the `<TABLE_PROPERTY>` element contains several `<ROW>` elements, which contain multiple `<CPROPERTY>` elements. Each `<CPROPERTY>` element contains modification information for a single field of the composite property.

In the example shown, each row of the composite property represents a service desk record. The `Prop_Composite_Sub_Time` property contains the date and time of the service call and the `Prop_Composite_Sub_String` property identifies the type of service provided.

```xml
<TABLE_PROPERTY NAME="Prop_Composite">
  <ROW>
    <CPROPERTY NAME="Prop_Composite_Sub_Time">
      <CVALUE>1377526368</CVALUE> <!-- Epoch time -->
    </CPROPERTY>
    <CPROPERTY NAME="Prop_Composite_Sub_String">
      <CVALUE>SW_update</CVALUE>
    </CPROPERTY>
  </ROW>
  <ROW>
    <CPROPERTY NAME="Prop_Composite_Sub_Time">
      <CVALUE>1377668342</CVALUE> <!-- Epoch time -->
    </CPROPERTY>
    <CPROPERTY NAME="Prop_Composite_Sub_String">
      <CVALUE>HW_maint</CVALUE>
    </CPROPERTY>
  </ROW>
</TABLE_PROPERTY>
```
The `<TABLE_PROPERTY>` element can contain several `<ROW>` elements.

- If the property was defined as an aggregate property, existing rows are retained when the property is updated. Rows submitted in the request message are appended as new rows to the table.
- If the property is not an aggregate property, the rows in the request message completely replace the table stored by the property.

See [Appendix 1: Forescout Property and Data Types](#) for details.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>The name of the host property or sub-property to be modified. <strong>This attribute is required.</strong></td>
<td>The name of a custom host property, or a sub-property of a custom composite property. This property must already exist in the Forescout platform before a request is submitted to the web service.</td>
</tr>
</tbody>
</table>

**VALUE and CVALUE Element: Host Property Values**

The `<VALUE>` element contains the value of a host property. Depending on the type of property you are modifying, the `<PROPERTY>` element can contain one or more `<VALUE>` elements.

Similarly, the `<CVALUE>` element contains the value of an individual field in a composite property.

Data must conform to valid formats accepted by the Forescout platform. The following example shows an EPOCH timestamp value, a format accepted by the Forescout platform.

```
<VALUE>1377526368</VALUE>
```

Similarly, null values should correspond to configured values in the Forescout platform. See [Define Null Values](#) for details.

**Update Transaction Examples**

All CounterACT Web Service requests use the same URI. The requested action is declared in the data section of the request. This means that the HTTP header is identical for all web service transactions.

**Update Multiple Properties**

In the following example, several properties are updated. The message body is saved to the `update_multiple.xml` file.

```xml
# /tmp/update_multiple.xml
<?xml version="1.0" encoding="UTF-8"?>
<FSAPI TYPE="request" API_VERSION="1.0">
  <TRANSACTION TYPE="update">
    <OPTIONS CREATE_NEW_HOST="true"/>
    <HOST_KEY NAME="ip" VALUE="10.0.0.101"/>
    <PROPERTIES>
      <PROPERTY NAME="Prop_String">
        <VALUE>Login</VALUE>
      </PROPERTY>
      <PROPERTY NAME="Prop_Time">
        <CVALUE>1377526368</CVALUE>  <!-- Epoch time -->
      </PROPERTY>
    </PROPERTIES>
  </TRANSACTION>
</FSAPI>
```
The following statement uses the curl utility to create and submit an HTTPS request message with the XML file in the message body.

```bash
curl -u "user@account:password" -k -H "Content-Type:application/xml" \
-d @/tmp/update_multiple.xml \
-X POST https://10.0.0.1/fsapi/niCore/Hosts
```

**Update a Composite Property**

In the following example several sub-properties of a composite property are updated. The message body is saved to the `update_composite.xml` file.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<FSAPI TYPE="request" API_VERSION="1.0">
  <TRANSACTION TYPE="update">
    <OPTIONS CREATE_NEW_HOST="true"/>
    <HOST_KEY NAME="ip" VALUE="10.0.0.101"/>
    <PROPERTIES>
      <TABLE_PROPERTY NAME="Prop_Composite">
        <ROW>
          <CPROPERTY NAME="Prop_Composite_Sub_Time">
            <CVALUE>1377526368</CVALUE>  <!-- Epoch time -->
          </CPROPERTY>
          <CPROPERTY NAME="Prop_Composite_Sub_String">
            <CVALUE>SW_update</CVALUE>
          </CPROPERTY>
        </ROW>
        <ROW>
          <CPROPERTY NAME="Prop_Composite_Sub_Time">
            <CVALUE>1377668342</CVALUE>  <!-- Epoch time -->
          </CPROPERTY>
          <CPROPERTY NAME="Prop_Composite_Sub_String">
            <CVALUE>HW_maint</CVALUE>
          </CPROPERTY>
        </ROW>
      </TABLE_PROPERTY>
    </PROPERTIES>
  </TRANSACTION>
</FSAPI>
```

The following statement uses the curl utility to create and submit an HTTPS request message with the XML file in the message body.

```bash
curl -u "user@account:password" -k -H "Content-Type:application/xml" \
-d @/tmp/update_composite.xml  \
-X POST https://10.0.0.1/fsapi/niCore/Hosts
```

**Create New Host on Update**

In the following example, the `CREATE_NEW_HOST` option is enabled. If the endpoint with IP address 10.0.0.101 is not known to the Forescout platform, it is added.

The message body is saved to the `update_create_new.xml` file.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<FSAPI TYPE="request" API_VERSION="1.0">
  <TRANSACTION TYPE="update">
    <OPTIONS CREATE_NEW_HOST="true"/>
    <HOST_KEY NAME="ip" VALUE="10.0.0.101"/>
    <PROPERTIES>
      <TABLE_PROPERTY NAME="Prop_Composite">
        <ROW>
          <CPROPERTY NAME="Prop_Composite_Sub_Time">
            <CVALUE>1377526368</CVALUE>  <!-- Epoch time -->
          </CPROPERTY>
          <CPROPERTY NAME="Prop_Composite_Sub_String">
            <CVALUE>SW_update</CVALUE>
          </CPROPERTY>
        </ROW>
        <ROW>
          <CPROPERTY NAME="Prop_Composite_Sub_Time">
            <CVALUE>1377668342</CVALUE>  <!-- Epoch time -->
          </CPROPERTY>
          <CPROPERTY NAME="Prop_Composite_Sub_String">
            <CVALUE>HW_maint</CVALUE>
          </CPROPERTY>
        </ROW>
    </TABLE_PROPERTY>
  </TRANSACTION>
</FSAPI>
```

The following statement uses the curl utility to create and submit an HTTPS request message with the XML file in the message body.

```bash
curl -u "user@account:password" -k -H "Content-Type:application/xml"  \
-d @/tmp/update_create_new.xml  \
-X POST https://10.0.0.1/fsapi/niCore/Hosts
```
The following statement uses the curl utility to create and submit an HTTPS request message with the XML file in the message body.

```
curl -u "user@account:password" -k -H "Content-Type:application/xml" \
    -d @/tmp/update_create_new.xml \
    -X POST https://10.0.0.1/fsapi/niCore/Hosts
```

The following response message indicates that the request succeeded.

```
<?xml version="1.0" encoding="UTF-8"?>
<FSAPI TYPE="response" API_VERSION="1.0">
  <STATUS>
    <CODE>FSAPI_OK</CODE>
    <MESSAGE>Successfully updated 1 properties for new host ip=10.0.0.101</MESSAGE>
  </STATUS>
</FSAPI>
```

By default, the value of the CREATE_NEW_HOST attribute is false. If the message body omits the attribute (as shown in the following example), this default value is used.

```
# /tmp/update_create_new.xml
<?xml version="1.0" encoding="UTF-8"?>
<FSAPI TYPE="request" API_VERSION="1.0">
  <TRANSACTION TYPE="update">
    <HOST_KEY NAME="ip" VALUE="10.0.0.102"/>
    <PROPERTIES>
      <PROPERTY NAME="Prop_String">
        <VALUE>Sales</VALUE>
      </PROPERTY>
    </PROPERTIES>
  </TRANSACTION>
</FSAPI>
```

In this case, if the endpoint with IP address 10.0.0.102 is not known to the Forescout platform, the following error is returned.

```
<?xml version="1.0" encoding="UTF-8"?>
<FSAPI TYPE="response" API_VERSION="1.0">
  <STATUS>
    <CODE>FSAPI_BAD_REQUEST_DATA</CODE>
    <MESSAGE>/Prop_String: Cannot create host ip=10.0.0.102; resubmit with CREATE_NEW_HOST="true"</MESSAGE>
  </STATUS>
</FSAPI>
```
Delete: Clear Host Property Values

The request message for the Delete task specifies host properties for an endpoint. The CounterACT Web Service clears the values of these properties.

- The target host property must be defined in the Forescout platform before you submit the request message.

In the CounterACT Web Service the requested action is declared in the `<TRANSACTION>` element in the data section of the request. For update tasks, the Transaction Type attribute has the value Delete. This essential setting differentiates this request message from other tasks.

The data section of the message should reflect the structure of the referenced host property. See Appendix 1: Forescout Property and Data Types for details.

Basic Delete Example

The following example shows the basic request message to clear a host property. The existing value of the Manager host property is deleted for endpoint 10.0.0.201. The message body is saved to the delete_prop_value.xml file.

```xml
# /tmp/delete_prop_value.xml
<?xml version="1.0" encoding="UTF-8"?>
<FSAPI TYPE="request" API_VERSION="1.0">
  <TRANSACTION TYPE="delete">
    <HOST_KEY NAME="ip" VALUE="10.0.0.101" />
    <PROPERTIES>
      <PROPERTY NAME="Prop_String" />
    </PROPERTIES>
  </TRANSACTION>
</FSAPI>
```

The following statement uses the curl utility to create and submit an HTTPS request message with the XML file in the message body.

```
curl -u "user@account:password" -k -H "Content-Type:application/xml" \
  -d @/tmp/delete_prop_value.xml \
  -X POST https://10.0.0.1/fsapi/niCore/Hosts
```

XML Schema for the Delete Task

This section describes the following elements that can be used when you construct a Delete request message.

- Forescout API Element
- TRANSACTION Element: Specify a Web Service Task
- HOST_KEY Element
- PROPERTIES Element
- PROPERTY Element
- TABLE_PROPERTY Element

Forescout API Element

The `<FSAPI>` element identifies the XML payload as CounterACT Web Service request content. For usage details, see Forescout API Element: Web Service Envelope.
**TRANSACTION Element: Specify a Web Service Task**

The `<TRANSACTION>` element specifies the action you want to apply to a specific host property. This element is the external envelope of the request message body, enclosing all other elements.

A single request message contains only one transaction element. This means it can perform one type of task using data relevant to a single endpoint.

You can submit several similar tasks for an endpoint in a single request message. For example, a single transaction can delete several properties for a single endpoint.

The **TYPE** attribute determines the action requested by the message. For the Delete task, this attribute should be set to **delete**, as in the following example:

```xml
<TRANSACTION TYPE="delete">
```

**HOST_KEY Element**

- The `<HOST_KEY>` element specifies the endpoint to which the task is applied. For usage details, see **HOST_KEY Element: Identify the Target Endpoint**.

**PROPERTIES Element**

The `<PROPERTIES>` element is an envelope for the detailed host property information included in the message. For usage details, see **PROPERTIES Element: Host Property Data Section**.

**PROPERTY Element**

Use the `<PROPERTY>` element to specify a Single Value or list property to be cleared. The **delete** transaction deletes all values in a list property.

Usage:

```xml
<PROPERTY NAME="prop_name" />
```

clears all values of the **prop_name** host property for the specified endpoint.

**TABLE_PROPERTY Element**

When you work with a composite property containing multiple fields, the `<TABLE_PROPERTY>` element is used to specify the property to be cleared. The **delete** transaction deletes the entire table stored in the specified composite property.

Usage:

```xml
<TABLE_PROPERTY NAME="cprop_name" />
```

clears all values of the **cprop_name** property for the specified endpoint.

**Delete Transaction Examples**

All CounterACT Web Service requests use the same URI and HTTP header for all transactions. The requested action is declared in the data section of the request.

**Delete Multiple Properties**

In the following example, several properties are deleted. The message body is saved to the `delete_multiple.xml` file.

```xml
# /tmp/delete_multiple.xml
<?xml version="1.0" encoding="UTF-8"?>
<FSAPI TYPE="request" API_VERSION="1.0">
    <TRANSACTION TYPE="delete">
```

The following statement uses the curl utility to create and submit an HTTPS request message with the XML file in the message body.

curl -u "user@account:password" -k -H "Content-Type:application/xml" \
-d @/tmp/delete_multiple.xml \
-X POST https://10.0.0.1/fsapi/niCore/Hosts

**Add List Values to Forescout Property Lists**

The request message for the `Add_list_value` transaction specifies the following information:

- The target Forescout Property Lists are identified by their names.
- Values to be added to the Forescout Property Lists.

The CounterACT Web Service updates the Forescout Property Lists with the supplied values.

> The specified Forescout Property Lists must be defined in the Forescout platform before you submit the request message.

In the CounterACT Web Service, the requested action is declared in the `<TRANSACTION>` element in the data section of the request. For the `Add_list_values` task, the Transaction Type attribute has the value `add_list_values`. This essential setting differentiates this request message from other tasks.

The data section of the message should contain the name of the Forescout Property List and the values to be added to it. Multiple Forescout Property Lists can be included in one message.

**Example: Basic Add_list_values**

The following example shows the basic request message to add values to Forescout Property Lists. The `update.xml` file is listed.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<FSAPI TYPE="request" API_VERSION="2.0">
  <TRANSACTION TYPE="add_list_values">
    <LISTS>
      <LIST NAME="sales_employee_id">
        <VALUE>A001</VALUE>
        <VALUE>A002</VALUE>
      </LIST>
      <LIST NAME="support_employee_id">
        <VALUE>B001</VALUE>
        <VALUE>B002</VALUE>
      </LIST>
    </LISTS>
  </TRANSACTION>
</FSAPI>
```
The following statement uses the curl utility to create and submit an HTTPS request message with the XML file in the message body.

```bash
curl -u "user@account:password" -k -H "Content-Type: application/xml" \
-d @/tmp/update.xml \
-X POST https://10.0.0.1/fsapi/niCore/Lists
```

**XML Schema for the Update Transaction**

This section describes the following elements that can be used when you construct an Update request message.

- **Forescout API Element: Web Service Envelope**
- **TRANSACTION Element: Specify a Web Service Task**
- **LISTS Element: Forescout Property Lists Data Section**
- **LIST Element: Specify a Single Forescout Property List**
- **VALUE Element: Forescout Property List Value**

**Forescout API Element: Web Service Envelope**

The `<FSAPI>` element identifies the XML payload as CounterACT Web Service request content, and provides basic information about the request content.

Usage:

```xml
<FSAPI TYPE="request" API_VERSION="2.0">
```

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE</td>
<td>The type of web service message.</td>
<td>Request: web service request message</td>
</tr>
<tr>
<td>API_VERSION</td>
<td>The version of the API that is used to process the request.</td>
<td>2.0</td>
</tr>
</tbody>
</table>

**TRANSACTION Element: Specify a Web Service Task**

The `<TRANSACTION>` element specifies the action you want to apply to specific Forescout Property Lists. This element is the external envelope of the request message body, enclosing all other elements.

A single request message contains only one transaction element. This means it can perform one type of task using data relevant to the Forescout Property Lists.

A single transaction can add values to multiple Forescout Property Lists.

The `TYPE` attribute determines the action requested by the message. For the Add_list_values task, this attribute should be set to `add_list_values`, as in the following example:

```xml
<TRANSACTION TYPE="add_list_values ">
```

**LISTS Element: Forescout Property Lists Data Section**

The `<LISTS>` element is an envelope for the detailed Forescout Property List information included in the message. In this section of the message, you specify the values that are added to the Forescout Property List.
The request can contain only one `<LISTS>` element, which must contain at least one `<LIST>` child element.

When one request updates several Forescout Property Lists, the `<LISTS>` element contains several child `<LIST>` elements. In the following example, the `<LISTS>` element contains two Forescout Property Lists to be updated.

```
<LISTS>
  <LIST NAME="list1">
    ...  
  </LIST>
  <LIST NAME="list2">
    ...  
  </LIST>
</LISTS>
```

**LIST Element: Specify a Single Forescout Property List**

Use the `<LIST>` element to specify a Forescout Property List to be updated. The `NAME` attribute is the identifier of the specified Forescout Property List. The `<LIST>` element can contain multiple `<VALUE>` element. In the following example, the `<LIST>` element contains two values to be added to the Forescout Property List.

```
<LIST NAME="list1">
  <VALUE>value1</VALUE>
  <VALUE>value2</VALUE>
</LIST>
```

In a Forescout Property List, each value is unique. This means if a value to be added to the Forescout Property List already exists in the list, it will not be added.

For more information about Forescout Property List, refer to “Defining and Managing Lists” in the *Forescout Administration Guide*.

**VALUE Element: Forescout Property List Value**

The `<VALUE>` element contains the value of a Forescout Property List. The `<LIST>` element can contain one or more `<VALUE>` elements.

**Example: add_list_values Transaction**

All CounterACT Web Service requests use the same URI. The requested action is declared in the data section of the request. This means that the HTTP header is identical for all web service transactions.

In the following example, several Forescout Property Lists are updated. The message body is saved to the `update.xml` file, which is listed.

```
# /tmp/update.xml
<?xml version="1.0" encoding="UTF-8"?>
<FSAPI TYPE="request" API_VERSION="2.0">
  <TRANSACTION TYPE="add_list_values">
    <LISTS>
      <LIST NAME="sales_employee_id">
        <VALUE>A001</VALUE>
        <VALUE>A002</VALUE>
      </LIST>
      <LIST NAME="support_employee_id">
        <VALUE>B001</VALUE>
        <VALUE>B002</VALUE>
      </LIST>
    </LISTS>
  </TRANSACTION>
</FSAPI>
```
The following statement uses the curl utility to create and submit an HTTPS request message with the XML file in the message body.

curl -u "user@account:password" -k -H "Content-Type:application/xml" -d @/tmp/update.xml -X POST https://10.0.0.1/fsapi/niCore/Lists

**Delete List Values in Forescout Property Lists**

The request message for the `delete_list_values` transaction specifies the following information:

- The target Forescout Property Lists are identified by their names.
- Values to be deleted in the Forescout Property Lists.

The CounterACT Web Service deleted the specified values in the specified Forescout Property Lists.

> The specified Forescout Property Lists must be defined in the Forescout platform before you submit the request message.

In the CounterACT Web Service the requested action is declared in the `<TRANSACTION>` element in the data section of the request. For the `delete_list_values` task, the Transaction Type attribute has the value `delete_list_values`. This essential setting differentiates this request message from other tasks.

The data section of the message should contain the name of the Forescout Property List and the values to be deleted in it. Multiple Forescout Property Lists can be included in one message.

**Example: Basic Delete_list_values**

The following example shows the basic request message to delete values in Forescout Property Lists. The `update.xml` file is listed.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<FSAPI TYPE="request" API_VERSION="2.0">
  <TRANSACTION TYPE="delete_list_values">
    <LISTS>
      <LIST NAME="sales_employee_id">
        <VALUE>A001</VALUE>
        <VALUE>A002</VALUE>
      </LIST>
      <LIST NAME="support_employee_id">
        <VALUE>B001</VALUE>
        <VALUE>B002</VALUE>
      </LIST>
    </LISTS>
  </TRANSACTION>
</FSAPI>
```
The following statement uses the curl utility to create and submit an HTTPS request message with the XML file in the message body.

```
curl -u "user@account:password" -k -H "Content-Type:application/xml" \ 
-d @/tmp/update.xml \ 
-X POST https://10.0.0.1/fsapi/niCore/Lists
```

**XML Schema for the Update Transaction**

This section describes in detail the following elements that can be used when you construct an Update request message.

- **Forescout API Element: Web Service Envelope**
- **TRANSACTION Element: Specify a Web Service Task**
- **LISTS Element: Forescout Property Lists Data Section**
- **LIST Element: Specify a Single Forescout Property List**
- **VALUE Element: Forescout Property List Value**

**Forescout API Element: Web Service Envelope**

The `<FSAPI>` element identifies the XML payload as CounterACT Web Service request content and provides basic information about the request content.

Usage:

```xml
<FSAPI TYPE="request" API_VERSION="2.0">
```

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE</td>
<td>The type of web service message.</td>
<td>Request: web service request message</td>
</tr>
<tr>
<td>API_VERSION</td>
<td>The version of the API that is used to process the request.</td>
<td>2.0</td>
</tr>
</tbody>
</table>

**TRANSACTION Element: Specify a Web Service Task**

The `<TRANSACTION>` element specifies the action you want to apply to specific Forescout Property Lists. This element is the external envelope of the request message body, enclosing all other elements.

A single request message contains only one transaction element. This means it can perform one type of task using data relevant to the Forescout Property Lists.

A single transaction can delete values in multiple Forescout Property Lists.

The TYPE attribute determines the action requested by the message. For the Delete_list_values task, this attribute should be set to `delete_list_values`, as in the following example:

```xml
<TRANSACTION TYPE="delete_list_values">
```

**LISTS Element: Forescout Property Lists Data Section**

The `<LISTS>` element is an envelope for the detailed Forescout Property List information included in the message. In this section of the message, you specify the values that will be added to the Forescout Property List.

The request can contain only one `<LISTS>` element, which must contain at least one `<LIST>` child element.
When one request updates several Forescout Property Lists, the `<LISTS>` element contains several child `<LIST>` elements. In the following example, the `<LISTS>` element contains two Forescout Property Lists to be updated.

```xml
<LISTS>
    <LIST NAME="list1">
        ...
    </LIST>
    <LIST NAME="list2">
        ...
    </LIST>
</LISTS>
```

**LIST Element: Specify a Single Forescout Property List**

Use the `<LIST>` element to specify a Forescout Property List to be updated. The `NAME` attribute is the identifier of the specified Forescout Property List. The `<LIST>` element can contain multiple `<VALUE>` elements. In the following example, the `<LIST>` element contains two values to be deleted in the Forescout Property List.

```xml
<LIST NAME="list1">
    <VALUE>value1</VALUE>
    <VALUE>value2</VALUE>
</LIST>
```

The delete action proceeds even if the value to be deleted does not exist in the Forescout Property List.

For more information about Forescout Property List, refer to “Defining and Managing Lists” in the *Forescout Administration Guide*.

**VALUE Element: Forescout Property List Value**

The `<VALUE>` element contains the value of a Forescout Property List. The `<LIST>` element can contain one or more `<VALUE>` elements.

**Example: Delete_list_values Transaction**

All CounterACT Web Service requests use the same URI. The requested action is declared in the data section of the request. This means that the HTTP header is identical for all web service transactions.

In the following example, several Forescout Property Lists are updated. The message body is saved to the `update.xml` file, which is listed.

```xml
# /tmp/update.xml
<?xml version="1.0" encoding="UTF-8"?>
<FSAPI TYPE="request" API_VERSION="2.0">
    <TRANSACTION TYPE="delete_list_values">
        <LISTS>
            <LIST NAME="sales_employee_id">
                <VALUE>A001</VALUE>
                <VALUE>A002</VALUE>
            </LIST>
            <LIST NAME="support_employee_id">
                <VALUE>B001</VALUE>
                <VALUE>B002</VALUE>
            </LIST>
        </LISTS>
    </TRANSACTION>
</FSAPI>
```
The following statement uses the curl utility to create and submit an HTTPS request message with the XML file in the message body.

```
curl -u "user@account:password" -k -H "Content-Type:application/xml" \
-d @/tmp/update.xml \
-X POST https://10.0.0.1/fsapi/niCore/Lists
```

### Delete All List Values in Forescout Property Lists

The request message for the `Delete_all_list_values` transaction specifies the following information:

- The target Forescout Property Lists are identified by their names.

The CounterACT Web Service deletes all the values in the specified Forescout Property Lists.

> The specified Forescout Property Lists must be defined in the Forescout platform before you can submit the request message.

In the CounterACT Web Service, the requested action is declared in the `<TRANSACTION>` element in the data section of the request. For the `Delete_all_list_values` task, the Transaction Type attribute has the value `delete_all_list_values`. This essential setting differentiates this request message from other tasks.

The data section of the message should contain the name of the Forescout Property List. Multiple Forescout Property Lists can be included in one message.

**Example: Basic Delete_all_list_values**

The following example shows the basic request message to delete values in Forescout Property Lists. The update.xml file is listed.

```
# /tmp/update.xml
<?xml version="1.0" encoding="UTF-8"?>
<FSAPI TYPE="request" API_VERSION="2.0">
  <TRANSACTION TYPE="delete_all_list_values">
    <LISTS>
      <LIST NAME="list1"/>
      <LIST NAME="list2"/>
    </LISTS>
  </TRANSACTION>
</FSAPI>
```

The following statement uses the curl utility to create and submit an HTTPS request message with the XML file in the message body.

```
curl -u "user@account:password" -k -H "Content-Type:application/xml" \
-d @/tmp/update.xml \
-X POST https://10.0.0.1/fsapi/niCore/Lists
```

### XML Schema for the Update Transaction

This section describes in detail the following elements that can be used when you construct an Update request message.
- **Forescout API Element: Web Service Envelope**
- **TRANSACTION Element: Specify a Web Service Task**
- **LISTS Element: Forescout Property Lists Data Section**
- **LIST Element: Specify a Single Forescout Property List**

**Forescout API Element: Web Service Envelope**

The `<FSAPI>` element identifies the XML payload as CounterACT Web Service request content, and provides basic information about the request content.

Usage:

```xml
<FSAPI TYPE="request" API_VERSION="2.0">
```

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE</td>
<td>The type of web service message.</td>
<td>Request: web service request message</td>
</tr>
<tr>
<td>API_VERSION</td>
<td>The version of the API that is used to process the request.</td>
<td>2.0</td>
</tr>
</tbody>
</table>

**TRANSACTION Element: Specify a Web Service Task**

The `<TRANSACTION>` element specifies the action you want to apply to specific Forescout Property Lists. This element is the external envelope of the request message body, enclosing all other elements.

A single request message contains only one transaction element. This means it can perform one type of task using data relevant to the Forescout Property Lists.

A single transaction can delete all the values for multiple Forescout Property Lists.

The `TYPE` attribute determines the action requested by the message. For the `Delete_all_list_values` task, this attribute should be set to `delete_all_list_values`, as in the following example:

```xml
<TRANSACTION TYPE="delete_all_list_values">
```

**LISTS Element: Forescout Property Lists Data Section**

The `<LISTS>` element is an envelope for the detailed Forescout Property List information included in the message. In this section of the message, you specify the Forescout Property Lists that will have all their values deleted.

The request can contain only one `<LISTS>` element, which must contain at least one `<LIST>` child element.

When one request updates several Forescout Property Lists, the `<LISTS>` element contains several child `<LIST>` elements. In the following example, the `<LISTS>` element contains two Forescout Property Lists to be updated.

```xml
<LISTS>
  <LIST NAME="list1"/>
  <LIST NAME="list2"/>
</LISTS>
```
LIST Element: Specify a Single Forescout Property List

Use the `<LIST>` element to specify a Forescout Property List to be updated. The NAME attribute is the identifier of the specified Forescout Property List. It can have no child element.

```xml
<LIST NAME="list1"/>
```

The delete action proceeds even if the value to be deleted does not exist in the Forescout Property List.

For more information about Forescout Property List, refer to “Defining and Managing Lists” in the Forescout Administration Guide.

Example: Delete_all_list_values Transaction

All CounterACT Web Service requests use the same URI. The requested action is declared in the data section of the request. This means that the HTTP header is identical for all web service transactions.

In the following example, several Forescout Property Lists are updated. The message body is saved to the `update.xml` file, which is listed.

```xml
# /tmp/update.xml
<?xml version="1.0" encoding="UTF-8"?>
<FSAPI TYPE="request" API_VERSION="2.0">
  <TRANSACTION TYPE="delete_all_list_values">
    <LISTS>
      <LIST NAME="sales_employee_id"/>
      <LIST NAME="support_employee_id"/>
    </LISTS>
  </TRANSACTION>
</FSAPI>
```

The following statement uses the curl utility to create and submit an HTTPS request message with the XML file in the message body.

```bash
curl -u "user@account:password" -k -H "Content-Type:application/xml" \
-d @/tmp/update.xml \
-X POST https://10.0.0.1/fsapi/niCore/Lists
```

Appendix 5: External Web Service Parser

DEX lets users configure a parsing pattern to map HTTP response content (typically in JSON/XML format) to a property as part of External Web Service Property definition. DEX supports three types of parser: JSON Path, XML Path, and REGEX.

Exchanged data typically uses an XML or JSON data structure. When you submit a request message to retrieve data, the returned payload is parsed to yield Forescout property values. When you submit a request message with data to an external service, the message header should conform to the required structure.

In addition to requests initiated by the Forescout platform, external platforms can submit REST messages to the CounterACT web service. See Work with the CounterACT Web Service for details.
The purpose of the table below is to provide a basic impression of how the parsing patterns map the HTTP response content to the External Web Service properties. The parsing patterns in the table are presented in the simplest way.

In order to write more complex and powerful parsing patterns, you should have a good understanding of the JSON Path/XML Path/REGEX. Note the following:

- REGEX can only be used to parse data for Single Value or Record Exists properties.
- REGEX can be used to parse data returned in JSON or XML format. However, JSON Path and XML Path are more efficient, and it is highly recommended to use one of these parsing methods instead of REGEX.

<table>
<thead>
<tr>
<th>HTTP Response Content Type</th>
<th>Forescout property type</th>
<th>Example of HTTP Response Content (bold is the value of the Forescout property)</th>
<th>Parse Data Using</th>
<th>Parsing Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSON Single</td>
<td>Single</td>
<td>{</td>
<td></td>
<td>$.result.update_by</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;result&quot; : {</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;update_by&quot; : &quot;abc123&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;update_ip&quot; : &quot;10.10.10.10&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>},</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;error&quot; : &quot;none&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>List</td>
<td>{</td>
<td></td>
<td>$.softwares[ ].name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;softwares&quot; : [</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{&quot;name&quot; : &quot;AAA&quot;, &quot;vendor&quot; : &quot;B&quot;},</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{&quot;name&quot; : &quot;CCC&quot;, &quot;vendor&quot; : &quot;D&quot;}</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Composite</td>
<td>{</td>
<td></td>
<td>$.result.update_by</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;result&quot; : {</td>
<td></td>
<td>Sub property1:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;update_by&quot; : &quot;abc123&quot;,</td>
<td></td>
<td>$.result.update_by</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;update_ip&quot; : &quot;10.10.10.10&quot;</td>
<td></td>
<td>Sub property2:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>},</td>
<td></td>
<td>$.error</td>
</tr>
<tr>
<td></td>
<td>Record Exist</td>
<td>{</td>
<td></td>
<td>$.result.update_ip</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;result&quot; : {</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;update_by&quot; : &quot;abc123&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;update_ip&quot; : &quot;10.10.10.10&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>},</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;error&quot; : &quot;none&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTTP Response Content Type</td>
<td>Forescout Property Type</td>
<td>Example of HTTP Response Content (bold is the value of the Forescout property)</td>
<td>Parse Data Using</td>
<td>Parsing Pattern</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------------</td>
<td>--------------------------------------------------------------------------------</td>
<td>------------------</td>
<td>-----------------</td>
</tr>
</tbody>
</table>
| XML                         | Single                 | <response> 
  <result> 
  <update_by>abc123<update_by> 
  <update_ip>10.10.2.3<update_ip> 
  </result> 
  <error>none</error> 
  <response> | XML Path               | /response/result/update_by /text() |
| XML                         | List                   | <softwares> 
  <software vendor="CompanyA"> 
   <name>AAA</name> 
  </software> 
  <software vendor="CompanyA"> 
   <name>BBB</name> 
  </software> 
  <software vendor="CompanyB"> 
   <name>CCC</name> 
  </software> 
 </softwares> | XML Path | /softwares/software[@vendor='CompanyA']/name/text() |
| XML                         | Composite              | <response> 
  <result> 
  <update_by>abc123<update_by> 
  <update_ip>10.10.2.3<update_ip> 
  </result> 
  <error>none</error> 
  <response> | XML Path               | Sub property1: 
  /response/result/update_by /text() 
  Sub property2: 
  /response/error/text() |
| XML                         | Record Exist           | <response> 
  <result> 
  <update_by>abc123<update_by> 
  <update_ip>10.10.2.3<update_ip> 
  </result> 
  <error>none</error> 
  <response> | XML Path               | /response/result/update_by /text() |
| Other                       | Single / Record Exist  | update_by=abc123,update_ip=10.1.1.1 | REGEX           | update_by=(.*) 
  update_ip=(.*),upda 
  te_ip=(.*),upda 
  te_ip=(.*) |