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 Resources page on the ForeScout website for additional technical documentation: https://www.forescout.com/company/resources/
- Have feedback or questions? Write to us at documentation@forescout.com
# Table of Contents

**About the Data Exchange Module** ................................................................. 5  
  About Support for Dual Stack Environments ............................................... 5

**Requirements** ............................................................................................... 5  
  CounterACT Software Requirements ............................................................. 5  
  Connectivity Requirements ......................................................................... 6

**ForeScout Extended Module License Requirements** ................................. 6  
  Per-Appliance Licensing Mode ................................................................. 6  
  Centralized Licensing Mode .................................................................... 8  
  More License Information ....................................................................... 8

**Install the Module** ...................................................................................... 8

**Configure the Module** ............................................................................... 9  
  Define Null Values ................................................................................. 10

**Integration Scenarios** ............................................................................. 11  
  Create a CounterACT Host Property Based on an External Database .......... 11  
  Update an External SQL Database ............................................................. 12  
  Create a CounterACT Host Property based on the CounterACT Web Service .... 13  
  Update a CounterACT Property List through the CounterACT Web Service .......... 13  
  Create a CounterACT Host Property based on an External Web Service .......... 14  
  Update an External Web Service ............................................................ 14  
  Use External Web Server Certificate Validation ............................................. 14

**Working With SQL and LDAP Databases** ................................................. 15  
  LDAP Directory Servers ........................................................................ 15  
  SQL Database Servers ........................................................................ 16  
  Define Connections to External Databases ................................................. 16  
  Test SQL Server Connections ................................................................ 21  
  Define LDAP or SQL Query Statements ................................................... 21  
  Tips for SQL Query Statements .............................................................. 24  
  Define Host Properties Based on Data from External Databases ................. 24  
  Test the Property .................................................................................. 31  
  Mapping External Data to Host Properties ............................................. 31  
  About Query Statements ........................................................................ 32  
  Import and Export Definitions ................................................................. 34  
  About Update Statements ....................................................................... 36  
  Managing Communication with Database Servers ...................................... 36  
  General Tools......................................................................................... 36  
  Tools to Manage Individual Entities ..................................................... 37  
  The JDBC Statistics Log ........................................................................ 37
# Table of Contents

## Working with External Web Services ......................................................... 39
- Define Requests to External Web Services .................................................. 39
- Authentication Tokens ............................................................................... 45
- Define a Cluster of CounterACT Devices .................................................... 50
- Define Host Properties Based on Data from External Web Services .......... 52

## Working with the CounterACT Web Service .................................................. 57
- Define CounterACT Web Service Accounts .................................................. 57
- Define Host Properties From the CounterACT Web Service ....................... 59
- CounterACT Web Service Security Settings ............................................... 64

## Actions and Properties for Data Integration ................................................. 65
- The DEX Update External Database Action .................................................. 65
- The DEX Send Web Service Request Action ............................................... 67

## Open Integration Module Information ......................................................... 71

## Appendix 1: CounterACT Property and Data Types ....................................... 72
- About Aggregate Properties ......................................................................... 74
- About Data Types ....................................................................................... 75
- Formatting the Date .................................................................................... 76

## Appendix 2: Using Advanced JDBC Attributes .............................................. 77

## Appendix 3: Example - Mapping Information from External Servers .......... 78

## Appendix 4: Submitting Data with the CounterACT Web API .......................... 79
- Web Service Implementation Overview ....................................................... 80
- Web Service Interaction ............................................................................. 80
  - Request Messages .................................................................................... 80
  - Response Messages ............................................................................... 81
  - Examples - Submitting Request Messages with Curl ............................... 82
- CounterACT Web Service Transactions ..................................................... 85
  - Update - Write Property Values to CounterACT ........................................ 86
  - Delete - Clear Host Property Values ......................................................... 92
  - Add List Values to CounterACT Property Lists ...................................... 95
  - Delete List Values in CounterACT Property Lists ................................... 98
  - Delete All List Values in CounterACT Property Lists ............................. 101

## Appendix 5: External Web Service Parser Construction .............................. 103

## Additional CounterACT Documentation ..................................................... 107
- Documentation Downloads ......................................................................... 107
- Documentation Portal ............................................................................... 107
- CounterACT Help Tools ............................................................................ 108
About the Data Exchange Module

ForeScout CounterACT® Open Integration Module: Data Exchange lets CounterACT use the following methods to communicate with external entities:

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

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

- This module is part of the ForeScout Open Integration Module. See Open Integration Module Information.

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

About Support for Dual Stack Environments

CounterACT version 8.0 detects endpoints and interacts with network devices based on both IPv4 and IPv6 addresses. However, IPv6 addresses are not yet supported by this component. 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 component.

Requirements

This section describes:

- CounterACT Software Requirements
- Connectivity Requirements
- ForeScout Extended Module License Requirements

CounterACT Software Requirements

The following CounterACT releases can work with this module:

- CounterACT version 8.0 or 8.0.1
- Authentication Module version 1.0 or above with the User Directory Plugin component
- An Open Integration Module license is required
- An active Maintenance Contract for the licensed module is required
Connectivity Requirements
Verify that CounterACT has permission to connect to external servers.

ForeScout Extended Module License Requirements

This ForeScout Extended Module requires a valid license. Licensing requirements differ based on which licensing mode your deployment is operating in:

- Per-Appliance Licensing Mode
- Centralized Licensing Mode

Identifying Your Licensing Mode in the Console

If your Enterprise Manager has a ForeScout CounterACT See license listed in the Console, your deployment is operating in Centralized Licensing Mode. If not, your deployment is operating in Per-Appliance Licensing Mode.

Select Options > Licenses to see whether you have a ForeScout CounterACT See license listed in the table.

Contact your ForeScout representative if you have any questions about identifying your licensing mode.

Per-Appliance Licensing Mode

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

When the demo period expires, you will be required to purchase a permanent module license. In order to continue working with the module, you must purchase the license.
Demo license extension requests and permanent license requests are made from the CounterACT 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 CounterACT 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 CounterACT. 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 CounterACT, 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.
Centralized Licensing Mode

When you set up your CounterACT deployment, you must activate a license file containing valid licenses for each feature you want to work with in your deployment, including Extended Modules. After the initial license file has been activated, you can update the file to add additional Extended Module licenses or change endpoint capacity for existing Extended Modules. For more information on obtaining Extended Module licenses, contact your ForeScout 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 Extended Module license has an associated capacity, indicating the number of endpoints the license can handle. The capacity of each Extended Module license varies by module, but does not exceed the capacity of the See license.

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

More License Information

Refer to the CounterACT Administration Guide for information on Extended Module licenses. You can also contact your ForeScout representative or license@forescout.com for more information.

Install the Module

Perform the following procedure to install the module.

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 - Centralized Licensing Mode

   To find out which licensing mode your deployment is working with, see Identifying Your Licensing Mode in the Console.

2. Download the module .fpi file.
3. Save the file to the machine where the CounterACT Console is installed.
4. Log into the CounterACT 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 will not proceed if you do not agree to the license agreement.

   - *The installation will begin 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.*

### Identifying Your Licensing Mode in the Console

If your Enterprise Manager has a *ForeScout CounterACT See* license listed in the Console, your deployment is operating in Centralized Licensing Mode. If not, your deployment is operating in Per-Appliance Licensing Mode.

Select **Options > Licenses** to see whether you have a *ForeScout CounterACT See* license listed in the table.

![Options and Licenses](image)

Contact your ForeScout representative if you have any questions about identifying your licensing mode.

### Configure the Module

To access the module configuration pane, select **Options** in the Console. In the Options tree, select **Data Exchange**. The Data Exchange configuration pane displays.
The module supports interaction with several types of external data sources, which are completely independent of each other. For example, configuration settings allow you to 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 Integration Scenarios for configuration procedures required for working with data sources.

The DEX Module automatically reboots when you define new data sources or host properties, or change any configuration settings.

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

Define Null Values

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

In the Data Exchange pane, select the General Settings tab. The Null Values section lets you define how CounterACT represents empty data fields when it maps received data to corresponding host properties.
Integration Scenarios

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

- Working with external SQL and LDAP databases:
  - Create a CounterACT Host Property Based on an External Database
  - Update an External SQL Database

- Working with External Web Services:
  - Update a CounterACT Property List through the CounterACT Web Service
  - Create a CounterACT 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 CounterACT Host Property based on the CounterACT Web Service
  - CounterACT Web Service Security Settings

Create a CounterACT Host Property Based on an External Database

In this scenario, a custom host property is created in CounterACT. CounterACT 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 CounterACT property.
Implemented by:

- CounterACT administrator
- External server administrator

**To retrieve data from external databases:**

1. Define a user account for CounterACT on each external DB server you want to work with.
2. Install the module. See *Install the Module*.
3. Configure the module's connection with external DB servers. See *Define Connections to External Servers*.
4. Define a custom CounterACT property that holds retrieved fields or attributes.
   - Define queries that retrieve data from external servers. See *Define LDAP or SQL Query Statements*.
   - Map retrieved data to custom properties. See *Define Host Properties Based on External Server Data*.
5. Include the new property in CounterACT policies.

CounterACT periodically queries the external DB server, and updates the host property based on retrieved data.

For general information, see *Working 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 CounterACT policy detects endpoint Anti-virus compliance status.

Implemented by:

- CounterACT administrator
- External server administrator

**To update external SQL databases:**

1. Define a user account for CounterACT 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 module connection with external databases.
3. Use the *DEX Update External Databases* action in a policy. CounterACT updates records in external SQL databases for hosts that match policy conditions.

See *The DEX Update External Database Action* and *About Update Statements*. 
Create a CounterACT Host Property based on the CounterACT Web Service

In this scenario, a custom host property is created in CounterACT. 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 CounterACT.

Implemented by:

- CounterACT administrator
- External application/service programmer

To implement this scenario, perform the following steps:

1. Configure and test web service connectivity to CounterACT.
2. Define access credentials for the CounterACT Web Service. See Working with the CounterACT Web Service for more information.
3. Create a new host property in CounterACT. See Define Host Properties For Web Service Interaction and Appendix 1: CounterACT Property and Data Types for more information.
4. Program the web service call on the external application. See Appendix 4: Submitting Data with the CounterACT Web API for more information.
5. Test the interaction.

Update a CounterACT Property List through the CounterACT Web Service

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

Implemented by:

- CounterACT administrator
- External application/service programmer

To update a CounterACT property list:

1. Configure and test web service connectivity to CounterACT.
2. Define access credentials for the CounterACT Web Service. See Working with the CounterACT Web Service for more information.
3. Create a new Property List in CounterACT. For more information, refer to the Defining and Managing Lists section in the CounterACT Console User Manual.
4. Program the web service call on the external application. See Appendix 4: Submitting Data with the CounterACT Web API for more information.
5. Test the interaction.
Create a CounterACT Host Property based on an External Web Service

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

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

To implement this scenario, perform the following steps:

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

2. Define a custom CounterACT 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.
   - Map retrieved data to custom properties. See Define Host Properties Based on Data from External Web Services.

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

For general information, see Working 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:
- CounterACT administrator
- External service administrator

To update external SQL databases:

1. Define a user account for CounterACT 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. CounterACT sends the web request to the service for hosts that match policy conditions.
   See The DEX Send Web Service Request Action.

Use External Web Server Certificate Validation

You can configure the DEX Plugin 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 the DEX Plugin will always set up the connection.

Implemented by:

- CounterACT 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, determine if you want the check box to the Validate Web Server Certificate field to be checked or unchecked.

   By default, the box is un-checked, and it means the DEX Plugin will always setup the connection. It does not matter whether the server certificate is a common commercial Certificate Authority issued certificate or not.

   If you check the box, you are adding certificate validation.
   - If the server certificate is a common commercial Certificate Authority issued certificate, the DEX Plugin will set up the connection to the server.
   - If the server certificate is not a common commercial Certificate Authority issued certificate, the DEX Plugin will not set up the connection to the server.

**Working With SQL and LDAP Databases**

The Data Exchange Module 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 in this module.
### 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
  - SQL Server 2012
- Oracle versions 10 and above
- PostgreSQL

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

- **Define Connections to External Servers** - You must configure CounterACT communication with external servers to use the features of this module.

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

- **Define Host Properties Based on External Server Data** - Define new CounterACT 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 Mapping External Data to Host Properties for details of exporting configured query, update, and property definitions.

### Define Connections to External Databases

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

This module works with LDAP directory servers and SQL database servers.

- Define 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 connection to SQL databases using the following procedure.

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

1. In the Data Exchange configuration pane, select the SQL/LDAP Databases tab and select the DB Servers tab. Select Add. The Add SQL Server wizard opens.
Define the following database server parameters in the General pane.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server Name</td>
<td>Enter a name for this connection.</td>
</tr>
<tr>
<td>Description</td>
<td>Enter a description of this connection.</td>
</tr>
<tr>
<td>IP Address</td>
<td>Enter the server IP address.</td>
</tr>
<tr>
<td>Port</td>
<td>Enter the server port.</td>
</tr>
<tr>
<td>Database Instance Name</td>
<td>Specify a database on the server.</td>
</tr>
<tr>
<td>Database Driver</td>
<td>Select a database driver from the Database Driver dropdown list.</td>
</tr>
<tr>
<td>Database Username</td>
<td>Enter the name of a user who can access the database in the Database Username field. The module does not support users defined as &quot;sys&quot; users (system administrators). If your environment uses NTLMv2 authentication, use the standard domain\user format.</td>
</tr>
<tr>
<td>Database Password</td>
<td>Enter the password of the user specified in the previous step in the Database Password field.</td>
</tr>
<tr>
<td>Verify Password</td>
<td>Enter the password of the user specified in the previous step in the Verify Password field.</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.

2. (Optional) To define additional connection attributes in the Java Database Connectivity (JDBC) connection string, select Advanced. The Advanced server settings dialog box opens.
The JDBC protocol represents the database connection as a URL string with attributes. The default string displayed reflects the database driver you selected, and includes placeholder tags for the configuration fields you specified earlier.

The default URL string used by CounterACT 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
```

![Advanced server settings dialog](image)

- **Add only attributes supported by the database driver you specify.**
  - **CounterACT 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. CounterACT uses the customized string to connect to this SQL server.

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

  **3.** Select **Next**. The Traffic Thresholds pane displays.
Use the following parameters to control traffic to the database server:

<table>
<thead>
<tr>
<th><strong>Maximum Requests</strong></th>
<th>Specify a maximum number of requests that the module sends to the server in the selected time period. If the module generates more requests, they are queued to a buffer.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Maximum Buffered Requests</strong></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 Managing Communication with Database Servers.

4. Select Next. The Replica pane displays. Define organizational replica servers that are used if the server defined in the General pane fails. Adding replica servers is optional.
5. (Optional) To add a replica server, select **Add**. The Add Replica dialog box opens.

6. Specify the following values:

<table>
<thead>
<tr>
<th>Field</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>

7. Select **OK**. The server you defined is added to the replica server table. Add additional replica servers as required.
8. Select **Finish**. The database and server information is added to the table in the DB Server tab.

9. (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 **SQL Servers** tab.
2. Select the servers you want to test.
3. Select **Test**. The Configuration Test dialog box opens.
4. Each CounterACT device attempts to log in to each server. Results are listed in the dialog box.

**Define LDAP or SQL Query Statements**

A **Query Statement** is a set of SQL statements or User Directory filter strings that CounterACT 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**. The General pane of the Add Query Statement wizard opens.
3. Specify the following general information about the query statement:

<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>In the Type drop-down list, specify whether the query statement is directed to an LDAP server or a SQL server. This value cannot be edited later.</td>
</tr>
</tbody>
</table>

4. Select **Next**. The Content pane displays.

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

6. To insert property tags that resolve to host property values, select **Add Tags**. The Tags dialog box opens.
7. Scroll through the list or use the Tag Filter field to search for a specific tag.
8. Select the tag and then select OK.
9. In the Content pane, select Next. The Cache pane displays.

10. The Cache period field defines how long the values returned by this query are valid. The cache period is how long data coming from the server is kept in the DEX Module’s memory. Within the cache period, queries are answered using the value stored in the DEX Module’s memory. Queries are not sent to the external server.

11. Select Finish. The query statement is added to the Query Statements list.
12. In the Data Exchange 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 will be 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 CounterACT 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');
```

**Define Host Properties Based on Data from External Databases**

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

**To create a property:**

1. In the Data Exchange pane, select the SQL/LDAP Databases tab and then select the Properties tab.

2. Select Add. The General pane of the Add Property from Database wizard opens.
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 appears 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. CounterACT 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. Whatever selection you make displays in the DB Queries field at the bottom of the pane.</td>
</tr>
<tr>
<td>Query Statement</td>
<td>Depending upon what is selected in this field, the Query Statement will populate and the details of the statement displays in the DB Queries field.</td>
</tr>
</tbody>
</table>

4. Select Next. The Map Query Fields pane displays with Single Property selected by default.
5. Select the type of property you want to create.
   - **Single property** contains one retrieved value. 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. In order to activate this property, you must have the Server field in the General tab selected to a SQL server. To create a Composite property, go to step 8.
   - **Record Exists** properties indicate 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:
   - In the **Data Type** drop-down list, select the type of data the property contains. See Appendix 1: CounterACT 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 the **Aggregate new values from each update** 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.

---

8. To define a Composite property:

   a. Select **Add**. The Sub-property dialog box opens.
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>In the drop-down list, select the type of data the property contains. See Appendix 1: CounterACT Property and Data Types for details.</td>
</tr>
<tr>
<td>Create Inventory Key</td>
<td>Select the Create Inventory Key option to let administrators list hosts based on this field in the Asset Inventory view. When this option is selected, the property appears as an index key in the Views pane of the Asset Inventory view.</td>
</tr>
</tbody>
</table>

c. Select OK. The field appears in the composite property table.

d. Repeat these steps to define additional fields of the composite property.

e. 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.

9. To define the Record Exists property:
Select the **Record Exists** radio button. Use this option if you only want to know of the existence of the data and not about the content of the data.

10. Select **Next**. The Display/Track pane displays.

Select the following options to specify where the property is displayed in the CounterACT Console.

| **Display Property in Inventory View** | Select this box to show the property in the Inventory. |
| **Description** | This field activates after selecting Display Property in Inventory View. Enter the title of the Inventory item for this property. |
| **Display Property in Host Profiles Pane of Home View** | Select this box to list this property in the Profiles tab of the Home View. |
| Display Property in the Assets Portal | Select this box to display this property in the Assets Portal. |
| Enable Track Changes | (For Single, Composite and List properties only) Select this box 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. |
| Name | This field activates after selecting Enable Track Changes. Enter a name of the Track Change for this property or accept the default name that is pulled from the Property Name field. |
| Description | This field activates after selecting Enable Track Changes. Enter a description describing the change in the selected property or accept the default description. |

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

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

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

To create a new property based upon an existing property:

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

   The **Property Name** field contains the following dummy value based on the existing property:

   
   ```
   Copy of <existing_property_name>
   ```
The Property Tag field contains a unique string generated by CounterACT. Edit these values to create unique identifiers for the new property.


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

## Test the Property

The property test verifies the definition of the property and the query that uses it. Because the query uses CounterACT 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. Select an item and then select Test. The Tag Values dialog box opens.

2. The names of the fields within 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, CounterACT submits a test query statement referenced by the new host property to the external server.

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

## Mapping External Data to Host Properties

To map information submitted through the CounterACT Web Service to host properties, see Create a CounterACT 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 irresolvable.
To populate a list property, use a query that retrieves a several rows from a column of a SQL database, or an LDAP attribute with multiple values.

CounterACT 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, CounterACT evaluates the corresponding property as *irresolvable*. Set the **Maximum Values Retrieved** setting in the General Settings tab to control this limit. See [Managing Communication with Database Servers](#) for more details.

### About Query Statements

A *Query Statement* is a set of SQL statements or User Directory filter strings that CounterACT 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 CounterACT properties. You define a query statement that returns several related data fields, and map them to different CounterACT properties.

#### One Query Statement, Many Endpoints

CounterACT 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 CounterACT *property tag* that specifies host-specific key values. Property tags are placeholder parameters that represent endpoint property values. When CounterACT 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, CounterACT 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, CounterACT submits a query for each host, replacing the {hostname} property tag with the host name of the host:

```
(name=QA_TOSIBA_1)
(name=SALES_LAPTOP2132)
```

Data is retrieved for each host, and can be mapped to the properties of that host.
For more information about Property Tags, see the CounterACT Administration Guide.

You cannot use tags that resolve to lists as index keys.

Because the MAC address is often used as an index key, this module provides two additional 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.

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.

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

Import and Export Definitions

In some cases it is 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.

CounterACT 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 CounterACT.

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 Services tab.
2. Within the selected tab, select the desired sub-tab.
3. (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. Select **Export**. The Import/Export Password dialog box opens.

5. To encrypt data, enter the **Password**. **Confirm the Password** and then select **OK**.

6. The Export Table dialog box opens.

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

8. Select **OK**. 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 Services** 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 the source xml file you want to import. 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.*
**About Update Statements**

An *Update Statement* is a set of SQL statements that CounterACT 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 CounterACT 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, CounterACT submits multiple statements to the external database. For each host that matches the policy conditions, CounterACT 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.

For more information, see [Update an External SQL Database](#).

**Managing Communication with Database Servers**

Policies that interact with external database and directory servers can generate significant communications between CounterACT and external servers. The module 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 CounterACT options such as policy recheck schedules to control the volume of communication with external servers.

- For tools to control all module 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 this module. These general tools are available on the General Settings tab of the Data Exchange pane.

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

  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 CounterACT accepts in response to a query statement. When the query statement retrieves a column or attribute with more values than this maximum, CounterACT evaluates the corresponding property as **Unresolvable**.
  - The **Timeout** setting determines how long CounterACT waits after submitting a request to the external server. If the external server does not respond within the timeout period, CounterACT evaluates related host properties as Irresolvable. If the **Log JDBC Statistics** option is selected, an error is recorded for this session in the log.

![Database Query Related Settings](image)

**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 CounterACT requests to the SQL server. See **Define Connections to External Servers**.

- Similar settings are available for LDAP servers. See the **User Directory Plugin Guide** 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 CounterACT sends the query to the external server. See **Define LDAP or SQL Query Statements**.

**The JDBC Statistics Log**

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

CounterACT 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 this interaction. CounterACT submitted the query request to resolve the value of this property. For update interactions this field is undefined.
- The time the request was sent to the external server, in Unix epoch time format.
- The time that the complete results were received from the external servers, in Unix epoch time format.
- The number of rows/values involved 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. For update interactions, this field is undefined.
- 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 sample shows typical log entries.

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

```
UPDATE
"insert statement"
"undef"
"1,358,332,479,503"
"1,358,332,479,512"
"9"
"1"
"undef"
"insert into statutable values ('160.0.0.2', 'no_antisirus')"
"root@jdbc:mysql://<Server_IP>:<Port>/<User_Name>"

QUERY
"eds ip"
"value exist"
"1,358,332,518,504"
"1,358,332,518,505"
"1"
"0"
"0"
"select FULLNAME from userstable where ip='160.0.0.2'
root@jdbc:mysql://<Server_IP>:<Port>/<User_Name>
```
Working with External Web Services

Data exchange between CounterACT 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 CounterACT 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 CounterACT, external platforms can submit REST messages to the CounterACT web service. See Working with the CounterACT Web Service.

Depending on the desired 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

Use the procedure described here to define an HTTP request message that CounterACT submits to an external web service. To allow DEX to 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. CounterACT 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 CounterACT sends to the web service, and specify which CounterACT Appliance(s) contact the web service.

The maximum number of HTTP request messages allowed to be configured is 10 and a cluster of the same HTTP request message is accounted as 1.

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.
3. Enter the following information:

<table>
<thead>
<tr>
<th>Request Name</th>
<th>The name of this request. Example: My Request</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>A brief description of the purpose of this request, and its target service.</td>
</tr>
</tbody>
</table>
| HTTP Method  | Select the HTTP method for retrieving data from an external web service.  
  - GET - retrieves information from the external web service. Using the same parameters gives the same results.  
  - POST - creates or updates the external web service. |
<table>
<thead>
<tr>
<th>URL (with request parameters)</th>
<th>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 Add Tags 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></th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP Message Headers</td>
<td>HTTP headers included in the message. Example message header: Authorization: Bearer {eds_ext_web_auth_token} Content-type: Application/json</td>
</tr>
<tr>
<td>HTTP Request Body</td>
<td>Enter the actual HTTP request data. The default limit is set to 5 KB. Example request body: { “time”:{eds_time_epoch}, “mac”:{mac}, “hostname”:{nbtab} }</td>
</tr>
<tr>
<td>No Record Returns HTTP Status</td>
<td>The HTTP status code that this service returns when no record matches the query parameters.</td>
</tr>
<tr>
<td>Tags</td>
<td>To insert runtime variables, place your cursor in the appropriate text field and then select Add Tags. These variables are replaced with: 1. Actual endpoint property values. See the CounterACT Administration Guide for details. 2. Non-endpoint-related information. These special tags consists of: – {eds_time_epoch}: Current time in epoch format – {eds_time_utc}: Current time in UTC format – {eds_ext_web_auth_token}: the relevant authorization token that refers to the authorization token selected in step 4. 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 place this tag since the http request simply uses the cookie for authorization in this case. For example tags, see the above field definitions.</td>
</tr>
</tbody>
</table>

4. Select **Next**. The Authentication pane of the wizard appears.
5. Specify the following information:
   
   In the Authentication section:

<table>
<thead>
<tr>
<th>Authentication</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No Authentication</strong></td>
<td>Select if you do not want to use a web service username/password or an Authorization Token.</td>
</tr>
<tr>
<td><strong>Use Basic Authentication Header</strong></td>
<td>This box is checked by default. De-select this box if you do not want to utilize 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</strong> <strong>Verify Password</strong></td>
<td>The password for authorizing credentials for the request message is sent as part of the message header. Enter the password and confirm the password.</td>
</tr>
</tbody>
</table>
   | **Use Authorization Token** | 1. Select the radio button to use an Authorization Token.  
                                 2. Select from the drop-down the token name. This Authorization Token is from the External Web Services tab, Authorization Token sub-tab. |
Validate Web Server Certificate

By default, the box is un-checked, and it means the DEX Plugin will always setup the connection. It does not matter whether the server certificate is a common commercial Certificate Authority issued certificate or not.

If you check the box, you are adding certificate validation.

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

In the Proxy section:

<table>
<thead>
<tr>
<th>Use Proxy</th>
<th>Select this box to access the web service via a proxy.</th>
</tr>
</thead>
<tbody>
<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 that the HTTP request will be sent to.</td>
</tr>
<tr>
<td>Proxy Username</td>
<td>Enter the username for authorizing CounterACT to access the proxy server.</td>
</tr>
<tr>
<td>Proxy Password</td>
<td>Enter the password for authorizing CounterACT to access the proxy server. Confirm the password.</td>
</tr>
</tbody>
</table>

6. Select Next. The **CounterACT Devices** pane of the Wizard appears.
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.

In some cases 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 CounterACT and the web service.

In these cases, select **Use a Connecting CounterACT Device.** In the **Connecting CounterACT Device** drop-down, 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 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.**

7. Select **Next.** The **Traffic Thresholds** pane of the wizard appears.

8. Specify the following information:
| **Maximum Requests Device Submits per** | Specifies the maximum 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** | Specifies how many request messages are queued when the CounterACT device is already handling the maximum number of concurrent requests. |

9. Select **Finish**. The request is added to the table in the Web Service Requests tab, see below for an example.

### Authentication Tokens

You can configure DEX to integrate with the APIs of third-party solutions that have implemented JSON web tokens or other standard tokens. This allows you to create an integration to the 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 Token** tab.
2. Select **Add**. The Add Authorization Token wizard opens.

3. Enter your configurations.
<table>
<thead>
<tr>
<th><strong>Name</strong></th>
<th>Name of your Authentication Token.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>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 that 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>
</tbody>
</table>
| **Method** | Select the HTTP method for retrieving data from an external web service.  
- GET - retrieves information from the external web service. Using the same parameters gives the same results.  
- POST - creates or updates the external web service. |
| **URL** | Enter a URL to submit a request for a JSON Web Token or other standard token to be generated. An example login URL could be:  
https://{company IP}/companyapi/login |
| **Headers** | HTTP headers included in the message. |
| **Body** | Enter the actual HTTP request data. |
| **Use Basic Authentication Header** | This box is checked by default. De-select this box if you not want to utilize the basic authentication header. |
| **Web Service Username** | Enter the username for authorizing credentials for the request message. This is submitted as part of the message header. |
| **Web Service Password / Verify Password** | The password for authorizing credentials for the request message is submitted as part of the message header. Enter the password and confirm the password. |
| **Field** | Select the field that the HTTP Response should look up for the authorization token.  
- Headers - The authorization token is in one of the headers  
- Entire Content - The authorization token is the entire content  
- Content: XML - The authorization token is in the content whose format is XML  
- Content: JSON - The authorization token is in the content whose format is JSON  
- In Cookie - The Authorization token is in the form of a http cookie |
Keyword/Parser | Keyword or parser needed to extract token from the http response based on the Field.
---|---
- Headers - The name of one header that indicates the authorization token
- Entire Content - (no need to have parser)
- Content: XML - Parsing pattern of XML Path. For more information about XMP Path, see Appendix 5: External Web Service Parser Construction.
- Content: JSON - Parsing pattern of JSON Path. For more information about JSON Path, see Appendix 5: External Web Service Parser Construction.
- In Cookie - no need to have parser

The proxy configuration that DEX uses for the token refreshment call depends on the external web service request that uses the token. If there are multiple external web service requests that use the same token but different proxies, upon refreshing the token, the proxy for the token refreshment call will be based on the most recent external web service request that uses 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 Token** tab.
2. Select **Import**. The Import dialog box opens.

![Import Dialog Box](image)

3. Select **Browse** to and select an XML file that will run on web services.

4. Select **OK**.
5. The file is imported and displays in the Authentication Token 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 Token tab.
2. Select Export. The Import / Export Password dialog box opens.

3. Enter the Password and Confirm Password, then select OK.
4. The Export Table dialog box opens.

5. Select Browse to and select a location for the XML file. Select Open.
6. Select OK. In the Export Table dialog box, select OK.
7. CounterACT may ask to create a directory on the computer. Select Yes.

8. The XML file is successfully exported.
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 clusters of CounterACT devices to handle web service request messages.

Before you follow the procedure described here, you must define a default CounterACT Connecting Device for the request message, as described in Define Requests to External Web Services. Devices that are not assigned to a cluster 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 opens. This dialog 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 you will use to specify the CounterACT device that handle the request.

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

<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 responses it receives from the external web service.</th>
</tr>
</thead>
</table>
| Available Devices Selected Devices | To specify the CounterACT devices that submit this request message through the specified connecting device:  
  1. Select an item in the Available Devices list.  
  2. Select **Add**. The selected device appears in the Selected Devices list.  
  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. |

5. Select **OK**. The new request message appears in the Web Service Requests tab.
Define Host Properties Based on Data from External Web Services

Properties store information that CounterACT discovers for each endpoint. Use this procedure to create new CounterACT properties to hold data retrieved from external services. This makes retrieved data available for use in CounterACT 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 also Appendix 1: CounterACT Property and Data Types.

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 new property, select Add. The General pane of the Add Property from External Web Service wizard opens.
   - To create a property based on an existing property, select the already defined property, and then select Duplicate. CounterACT creates a copy of the property in the Properties tab. Select the duplicate, and then select Edit.

3. Specify the following information:

<table>
<thead>
<tr>
<th>Property Name</th>
<th>The name of the property.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Tag</td>
<td>The internal runtime tag of the property. CounterACT references the property using this unique identification string. Do not use spaces in the string.</td>
</tr>
<tr>
<td>(ASCII only)</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>A brief description of the purpose of this property, and its source web service.</td>
</tr>
</tbody>
</table>
Web Service Request | The name of the web service request that retrieves the data contained in this property.

4. Select **Next**. The Map Data pane displays.

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**, specify the following information:

<table>
<thead>
<tr>
<th><strong>Data Type</strong></th>
<th>The type of data the property contains. See <a href="#">Appendix 1: CounterACT Property and Data Types</a> for details.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parse Data Using</strong></td>
<td>The type of format of the expression used to extract data from the payload returned by the external web service. Choices are XML Path, JSON Path, and Regular Expression. See the <strong>Parsing Pattern</strong> field or <a href="#">Appendix 5: External Web Service Parser Construction</a> for details.</td>
</tr>
</tbody>
</table>
7. To define a **Composite Property**:

a. Specify the following information:

<table>
<thead>
<tr>
<th>Parsing Pattern</th>
<th>The expression used to extract data.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• For XML Path, the Parsing Pattern could look like: /response/result/serial_number/text()</td>
</tr>
<tr>
<td></td>
<td>• For JSON Path, the Parsing Pattern could look like: $..serial_number</td>
</tr>
<tr>
<td></td>
<td>• Regular Expression, the Parsing Pattern could look like: Begin(.*?)End</td>
</tr>
<tr>
<td>Aggregate new values from each update</td>
<td>(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.</td>
</tr>
</tbody>
</table>

See [Appendix 5: External Web Service Parser Construction](#) for details.

b. Select Add. The Sub-property dialog box opens.

c. Specify the following information:

| Field Name | The name of the Sub-property. |
Description | A description of the data contained in this Sub-property.
---|---
Parsing Pattern | The expression used to extract data. See Appendix 5: External Web Service Parser Construction for details.
Data Type | The type of data the property contains. Choices are XML Path and JSON Path. See Appendix 1: CounterACT Property and Data Types for details.
Create Inventory Key | When this option is selected, the property appears as an index key in the Views pane of the Inventory view.

d. Select OK. The field appears 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 Exist Property:
   a. Specify the following information:

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

9. Select Next. The Display/Track pane displays.

10. Specify where the property will be displayed in the CounterACT Console:

   | Display Property in Inventory View | Show this property in the Inventory.
<table>
<thead>
<tr>
<th>Description</th>
<th>This field activates after selecting Display Property in Inventory View. Enter the title of the Inventory item for this property.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Display Property in Host Profiles Pane of Home View</strong></td>
<td>Select this box 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 box to display this property in the Assets Portal.</td>
</tr>
<tr>
<td><strong>Enable Track Changes</strong></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>
<tr>
<td><strong>Name</strong></td>
<td>This field activates after selecting Enable Track Changes. 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 activates after selecting Enable Track Changes. Enter a description describing the change in the selected property or accept the default description.</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 appear in the Properties tree under the **DEX: External Web Service** folder. Related change tracking properties appear in the Track Changes folder.

- **CounterACT only retrieves data and updates values for properties that are referenced by an active policy.**
Working with the CounterACT Web Service

Data exchange between CounterACT 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, CounterACT can initiate messaging to external platforms. See Working with External Web Services.

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

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

Define CounterACT Web Service Accounts

This module supports the CounterACT Web Service, which lets external platforms and services interact with CounterACT 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.

For more information about the CounterACT Web Service, see Appendix 4: Submitting Data with the CounterACT Web API.

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 Services tab, and then select the Accounts tab.
2. Select **Add**. The Add Account dialog box opens.

a. Specify the following information:

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

3. Select **OK**. The account appears 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`. For the example shown below, the login target is `Servicedesk@Corporate Data Web Server`
Define Host Properties From the CounterACT Web Service

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

For more information about the CounterACT Web Service, see Appendix 4: Submitting Data with the CounterACT Web API and the ForeScout CounterACT Open Integration Module: Web API Configuration Guide.

Before you perform this procedure, see Appendix 1: CounterACT Property and Data Types for detailed information about CounterACT 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. The General pane of the Add Property from CounterACT Web Service wizard opens.
3. Specify the following information:

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

4. Select Next. The Map Data pane displays.
5. Select the type of host property you want to create.

<table>
<thead>
<tr>
<th><strong>Single Value Property</strong></th>
<th>Contains one retrieved value. To define a single value property, select from the <strong>Data Type</strong> drop-down list the type of data the property contains. See <a href="#">Appendix 1: CounterACT Property and Data Types</a> for details. Continue to step 6.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>List Property</strong></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:
---|---
a. Select **Add**. The Add Sub-property dialog box opens.
b. In the **Field Name** and **Description** fields, enter the name of this field of the composite property, and a short description of its content.
c. In the **Data Type** drop-down list, select the type of data the property contains. See Appendix 1: CounterACT Property and Data Types for details.
d. 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 appears as an index key in the Views pane of the Inventory view.
e. Select **OK**. The field appears in the composite property table.
f. Repeat these steps to define additional fields of the composite property.
g. 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**. The Display/Track pane displays.
Select the following options to specify where the property is displayed at the Console.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display Property in Inventory View</td>
<td>Show this property in the Inventory. In the Description field, 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>List this property in the Profiles tab of the Home view.</td>
</tr>
<tr>
<td>Display Property in the Assets Portal</td>
<td>Show 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>

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

When you create policy conditions, your properties appear in the Properties tree under the **DEX: CounterACT Web Service** folder. Related change tracking properties appear 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. Follow the procedure described here to allow users to contact the CounterACT web service.

In addition, login protection settings are provided.

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. The Add IP Range dialog box opens.
   b. Select either All IPs or enter an IP Range.
   c. Select OK. The IP range appears in the IP Address Range list.

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

**Actions and Properties for Data Integration**

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

**The 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 CounterACT policy detects endpoint Antivirus compliance status.

When you use the **DEX Update External Database** action in a policy, CounterACT 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. CounterACT 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. CounterACT 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 the **DEX Update External Database** action.
4. In the Parameters tab, configure the following:

<table>
<thead>
<tr>
<th><strong>DEX DB Server</strong></th>
<th>Using the drop-down, select the target external server. The update statement is submitted to this server.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Statement</strong></td>
<td>The SQL statement that CounterACT submits to the external database. &lt;br&gt; Select <strong>Add Tags</strong> to insert CounterACT property tags that resolve to host property values. &lt;br&gt; - For the index key of the query, you cannot use tags that resolve to a list.</td>
</tr>
<tr>
<td><strong>Simulate</strong></td>
<td>Checking this box generates SQL statements without submitting them to the external database.</td>
</tr>
</tbody>
</table>

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

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

6. In the Views tree of the Home view, select **Policy** and navigate 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 appears, 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 CounterACT tags resolved to host property values as you 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 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, CounterACT updates values in the corresponding record of the external database according to the statement you defined for the action.

The 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, CounterACT 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 the DEX Send Web Service Request action.
2. In the Parameters tab, perform the following:

<table>
<thead>
<tr>
<th>Action Identifier</th>
<th>Identifies requests submitted when this action is applied to an endpoint.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use predefined Web Service Request to fill in parameters</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. In the Request Name drop-down, select a pre-defined web service request. Parameters reflect settings of the web request.</td>
</tr>
<tr>
<td>Define a new message for this action</td>
<td>Define an HTTP request message from scratch.</td>
</tr>
</tbody>
</table>

3. In the HTTP Request section specify components of the HTTP message in the following fields:

<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>Select the appropriate HTTP message method.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request URL</td>
<td>Enter the URL of the target web service.</td>
</tr>
<tr>
<td>HTTP Request Headers(raw)</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.</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>3. <em>Only add tags that reference Single Value properties. You cannot add tags that refer to list or composite properties.</em></td>
</tr>
<tr>
<td><strong>Upload Body as File</strong></td>
<td>Select this option to append the contents of the Request Body field to the request message as a text file.</td>
</tr>
</tbody>
</table>
| **Tags**              | To insert runtime variables, place your cursor in the appropriate text field and then select Add Tags. These variables are replaced with:  
  1. Actual endpoint property values. See the CounterACT Administration Guide for details.  
  2. Non-endpoint-related information. These special tags consists of:  
     - `{eds_time_epoch}`: Current time in epoch format  
     - `{eds_time_utc}`: Current time in UTC format  
     - `{eds_ext_web_auth_token}`: the relevant authorization token that refers to the authorization token selected in step 4. 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 place this tag since the http request simply uses the cookie for authorization in this case. |

4. In the Authentication section:

<table>
<thead>
<tr>
<th><strong>No Authentication</strong></th>
<th>Select 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 box if you want to utilize 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>The password for authorizing credentials for the request message is submitted as part of the message header. Enter the password and confirm the password.</td>
</tr>
</tbody>
</table>
5. In the **Proxy section**:

<table>
<thead>
<tr>
<th><strong>Use HTTP Proxy</strong></th>
<th>Select this box to access the web service via a proxy.</th>
</tr>
</thead>
<tbody>
<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>Enter the port number on the proxy server that the HTTP request will be sent to.</td>
</tr>
<tr>
<td><strong>Proxy Username</strong></td>
<td>Enter the username for authorizing CounterACT to access the proxy server.</td>
</tr>
<tr>
<td><strong>Proxy Password</strong></td>
<td>Enter the password for authorizing CounterACT to access the proxy server. Confirm the password.</td>
</tr>
</tbody>
</table>

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

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

7. View 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. CounterACT interprets all HTTP 2xx return codes as action success. All other HTTP return codes are interpreted as action failure.
Open Integration Module Information

The Open Integration Module contains the Data Exchange Module, which supports bidirectional query and update interactions with external SQL, Oracle, and LDAP servers.

In addition, it contains other components that support the CounterACT Web Service, which lets external entities communicate with CounterACT.

Information can be retrieved from CounterACT using simple, yet powerful web service requests based on HTTP interaction.

Information can be submitted to CounterACT using web service requests with an XML data body. The CounterACT Web Service parses the data to update CounterACT host properties.
Appendix 1: CounterACT Property and Data Types

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

You can create the following types of properties:

**Single Value Property** contains one value. For example:

- A string property that contains the GUID of the endpoint

**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.

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

```
<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 `<PROPERTY>` and `<VALUE>` elements parallel the `<PROPERTY>` and `<VALUE>` elements used to describe non-composite properties. The following example describes a record in a composite property.

```
<TABLE_PROPERTY NAME="Prop_Composite">
   <ROW>
      <PROPERTY NAME="Prop_Composite_Sub_Time">
         <VALUE>1377526368</VALUE>  <!-- Epoch time -->
      </PROPERTY>
      <PROPERTY NAME="Prop_Composite_Sub_String">
         <VALUE>SW_update</VALUE>
      </PROPERTY>
   </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.

- Application Added/Removed
- Service Banner Change
- Device interfaces on the host

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.
For more information about Track Changes properties, see the CounterACT Administration Guide.

**About Aggregate Properties**

When you create list or composite properties for use with the CounterACT Web Service, select the **Is Aggregator** 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 completely 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 completely 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

CounterACT 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 matching options that CounterACT provides when you use the property in a policy condition. For example: CounterACT 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>CounterACT 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>
<tr>
<td>Date</td>
<td>NUMBER type with Epoch Time in seconds, or DATE, DATETIME, TIMESTAMP</td>
<td>UTC Coded Timestamp</td>
<td>Plain Text</td>
</tr>
<tr>
<td>Boolean</td>
<td>TRUE, FALSE, BIT, BINARY</td>
<td>Boolean</td>
<td>Plain text</td>
</tr>
</tbody>
</table>

In order to let the DEX Module know the correct meaning of the plain text retrieved from the external web service that represents a date/time, please select a date format when configuring the property. There are three options for the format:

- **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** - refer to Formatting the Date for how to write a customized format.
# Formatting the Date

When writing the date in customized format, refer to the tokens table below.

<table>
<thead>
<tr>
<th>Token</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%a</td>
<td>Day of the week, abbreviated with only three characters (e.g. Mon, Tue, Wed, Thu, Fri, Sat, Sun)</td>
</tr>
<tr>
<td>%A</td>
<td>Day of the week, full name (e.g. Monday)</td>
</tr>
<tr>
<td>%b</td>
<td>Month, abbreviated (e.g. Jan)</td>
</tr>
<tr>
<td>%B</td>
<td>Month, full name (e.g. 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, i.e., 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, i.e., 01.</td>
</tr>
<tr>
<td>%m</td>
<td>Month number (1-12). It is acceptable to have a leading zero before a single number, i.e., 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 module user, are aware of its meaning and what you need to do is give the format of the string to the module. The module will then understand 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, you refer to the tokens table above and find these three tokens that can serve the purpose:

<table>
<thead>
<tr>
<th>%d</th>
<th>Numeric day of the month (1-31). It is acceptable to have a leading zero before a single number, i.e., 01.</th>
</tr>
</thead>
<tbody>
<tr>
<td>%m</td>
<td>Month number (1-12). It is acceptable to have a leading zero before a single number, i.e., 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 since they are just separators that have no meaning with respect to the date.

» If the date string contains no time zone information, the DEX Module will consider the time zone as GMT+0000.

**Appendix 2: Using Advanced JDBC Attributes**

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

**To use advanced JDBC attributes:**

1. 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 configuration fields you specified earlier.

2. 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. CounterACT does not validate these optional attributes.
3. Select **OK** to save the URL string and return to the wizard. Complete server definition and apply changes.

### Appendix 3: Example - Mapping Information from External Servers

The following example demonstrates 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>001122aabbcc</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

<table>
<thead>
<tr>
<th>ID</th>
<th>USER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>Alice</td>
</tr>
</tbody>
</table>

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 CounterACT retrieves for that host from the external database.

### Appendix 4: Submitting Data with the CounterACT Web API

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

The web service lets you communicate with CounterACT using web service requests. You create web service requests that reference CounterACT host properties and CounterACT Property List values. These messages contain property information in an XML data body. The CounterACT Web Service parses the data to update CounterACT host properties and CounterACT Property List values.
For information about retrieving information from CounterACT using the web service, refer to the ForeScout CounterACT Open Integration Module: Web API Configuration Guide.

Web Service Implementation Overview

Typically, web service integration follows this general procedure. Some configuration steps must be performed on CounterACT to support integration. Work with the CounterACT administrator on these steps.

- For updating host properties: plan and create custom host properties based on the structure of the data that will be submitted to CounterACT. For more information, see Define Host Properties For Web Service Interaction.
- Properties and accounts are related, and should be planned with the third-party user who will interact with the Web service. The CounterACT administrator can define several accounts for the web service. Each account allows modification of a specified set of host properties.
- For updating CounterACT Property List values: plan and create custom CounterACT Property Lists based on the structure of the data that will be submitted to CounterACT. For more information, refer to the Defining and Managing Lists section in the CounterACT Console User Manual.
- Each CounterACT Web Service account allows modification of all the CounterACT Property Lists.
- Create third-party login credentials for the web service. For more information, see Define Web Service Accounts.
- Third party users must receive account(s) that allow them to 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.

Web Service Interaction

This section provides a general 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 CounterACT Enterprise Manager in the network. Verify that this target is accessible from any clients that will send requests to the CounterACT Web Service.

<Purpose> is the purpose of the request. There are two options for it: Hosts for updating host properties and Lists for updating CounterACT 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 aGFtZWVkQGZvcmVzY29idC5jb206aGFtZW291dC5jb20=
Accept: application/xml
Content-type: application/xml
```

Note that:
- All requests to the CounterACT Web Service use secured HTTP (HTTPS)
- All requests to the 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 CounterACT administrator.
- Endpoint/Property List data is submitted to the ForeScout 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.

```
<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 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>
<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 &lt;VALUE&gt; 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 - Submitting 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 that:

- The entire command is a single line. Backslash characters \ indicate line continuation, as in most Unix environments.
- Replace `{username} {account} and `{password}` placeholders with the web service account credentials you received from the CounterACT administrator.
- The `{EM.IP}` placeholder should be replaced with the IP address of the CounterACT 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 certs.

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

```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
<?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 CounterACT Console.
Example 2: To Update CounterACT Property List Values

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

```
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 that:

- The entire command is a single line. Backslash characters \ indicate line continuation, as in most Unix environments.
- Replace \{username\} \{account\} and \{password\} placeholders with the web service account credentials you received from the CounterACT administrator.
- The \{EM.IP\} placeholder should be replaced with the IP address of the CounterACT 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 certs.

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

```xml
<?xml version="1.0" encoding="UTF-8"?>
<FSAPI TYPE="request" API_VERSION="2.0">
  <TRANSACTION TYPE="add_list_values">
    <LISTS>
```

# /tmp/update.xml
<?xml version="1.0" encoding="UTF-8"?>
<FSAPI TYPE="request" API_VERSION="2.0">
  <TRANSACTION TYPE="add_list_values">
    <LISTS>
```
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/Lists
```

The following response message indicates that the request succeeded.

```
<?xml version="1.0" encoding="UTF-8"?>
<FSAPI TYPE="response" API_VERSION="2.0">
  <STATUS>
    <CODE>FSAPI_OK</CODE>
    <MESSAGE>Successfully added values to the [2] lists.</MESSAGE>
  </STATUS>
</FSAPI>
```

The updated lists, *sales_employee_id* and *support_employee_id* are displayed in the Options -> Lists.

### CounterACT Web Service Transactions

Transactions are the tasks you perform using web service message requests. This section lists the transactions provided by the CounterACT Web Service, and gives implementation details for each one.

Different API versions have different sets of traction types. The request xml file should specify the correct *API_VERSION* to make the transaction work. Typically, when working with CounterACT 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 CounterACT

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

- The target host is 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 CounterACT 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. Refer to Appendix 1: CounterACT Property and Data Types.

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 below.

```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 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
- 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:

```
<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:

```
<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/disabled.

Usage:

```
<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 CounterACT, 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:

```
<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 which host property values 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: CounterACT Property and Data Types](#) for more information.

**TABLE_PROPERTY, CPROPERTY, and ROW elements – Specify a Composite Property**

When you work with 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.

```
<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: CounterACT Property and Data Types](#) for more information.

<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 CounterACT 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 CounterACT. The following example shows an EPOCH timestamp value – a format accepted by CounterACT.

```xml
<VVALUE>1377526368</VVALUE>
```

Similarly, null values should correspond to configured values in CounterACT. See Define Null Values for more information.

**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, which is listed below.

```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.

```
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, listed below.

```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"/>
        </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.

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

Create New Host Upon 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 CounterACT, it is added to CounterACT.

The message body is saved to the update_create_new.xml file, which is listed below.

```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_create_new.xml \
-X POST https://10.0.0.1/fsapi/niCore/Hosts

The following response message indicates that the request succeeded.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<FSAPI TYPE="response" API_VERSION="1.0">
<STATUS>
<CODE>FSAPI_OK</CODE>
<MESAGE>Successfully updated 1 properties for new host
```
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 - the default value false 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 CounterACT, 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 CounterACT 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. Refer to Appendix 1: CounterACT Property and Data Types.

**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, which is listed below.

```
# /tmp/delete_prop_value.xml
<?xml version="1.0" encoding="UTF-8"?>
```
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 in detail 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:

<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, which is listed below.

```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_1" />
      <PROPERTY NAME="Prop_String_2" />
      <PROPERTY NAME="Prop_Boolean" />
    </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_multiple.xml \
  -X POST https://10.0.0.1/fsapi/nicore/Hosts
```
Add List Values to CounterACT Property Lists

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

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

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

> The specified CounterACT Property Lists must be defined in CounterACT 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 CounterACT Property List and the values to be added to it. It is allowed to have multiple CounterACT Property Lists in one message.

**Example - Basic Add_list_values**

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

```
# /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
```

**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 – CounterACT Property Lists Data Section**
- **LIST element – Specify a Single CounterACT Property List**
- **VALUE element – CounterACT 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:**

```
<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><strong>TYPE</strong></td>
<td>The type of web service message.</td>
<td>Request – web service request message.</td>
</tr>
<tr>
<td><strong>API_VERSION</strong></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 CounterACT 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 CounterACT Property Lists.

A single transaction can add values to multiple CounterACT 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:

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

**LISTS element – CounterACT Property Lists Data Section**

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

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

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

```
<Lists>
    <List>
        <List NAME="list1">
            ...
        </List>
    <List>
        <List NAME="list2">
            ...
        </List>
```
**LIST element – Specify a Single CounterACT Property List**

Use the `<LIST>` element to specify a CounterACT Property List to be updated. The `NAME` attribute is the identifier of the specified CounterACT 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 CounterACT Property List.

```
<List NAME="list1">
  <Value>value1</Value>
  <Value>value2</Value>
</List>
```

- In a CounterACT Property List, each value is unique. This means if a value to be added to the CounterACT Property List already exists in the list, it won’t get added.

For more information about CounterACT Property List, refer to the Defining and Managing Lists section in the *CounterACT Console User Manual*.

**VALUE element – CounterACT Property List Value**

The `<VALUE>` element contains the value of a CounterACT 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 CounterACT Property Lists are updated. The message body is saved to the `update.xml` file, which is listed below.

```
# /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 CounterACT Property Lists

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

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

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

> The specified CounterACT Property Lists must be defined in CounterACT 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 CounterACT Property List and the values to be deleted in it. It is allowed to have multiple CounterACT Property Lists in one message.

**Example - Basic Delete_list_values**

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

```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
```

**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 – CounterACT Property Lists Data Section
- LIST element – Specify a Single CounterACT Property List
- VALUE element – CounterACT 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 CounterACT 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 CounterACT Property Lists.

A single transaction can delete values in multiple CounterACT 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 – CounterACT Property Lists Data Section

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

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

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

```xml
<LISTS>
    <LIST NAME="list1">
        ...
    </LIST>
    <LIST NAME="list2">
        ...
    </LIST>
</LISTS>
```
LIST element – Specify a Single CounterACT Property List

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

```
<List NAME="list1">
  <Value>value1</Value>
  <Value>value2</Value>
</List>
```

- The delete action will still proceed if the value to be deleted doesn’t exist in the CounterACT Property List.

For more information about CounterACT Property List, refer to the Defining and Managing Lists section in the CounterACT Console User Manual.

VALUE element – CounterACT Property List Value

The `<VALUE>` element contains the value of a CounterACT 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 CounterACT Property Lists are updated. The message body is saved to the `update.xml` file, which is listed below.

```
# /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 CounterACT Property Lists

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

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

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

- **The specified CounterACT Property Lists must be defined in CounterACT 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 CounterACT Property List. It is allowed to have multiple CounterACT Property Lists in one message.

**Example - Basic Delete_all_list_values**

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

```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 – CounterACT Property Lists Data Section
- LIST element – Specify a Single CounterACT 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 CounterACT 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 CounterACT Property Lists.

A single transaction can delete all the values for multiple CounterACT 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 – CounterACT Property Lists Data Section

The `<LISTS>` element is an envelope for the detailed CounterACT Property List information included in the message. In this section of the message, you specify which CounterACT Property Lists 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 CounterACT Property Lists, the `<LISTS>` element contains several child `<LIST>` elements. In the following example, the `<LISTS>` element contains two CounterACT Property Lists to be updated.

```xml
<Lists>
  <List>
    <List NAME="list1"/>
    <List NAME="list2"/>
  </List>
</Lists>
```

List element – Specify a Single CounterACT Property List

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

```xml
<List NAME="list1"/>
```
The delete action will still proceed if the value to be deleted doesn’t exist in the CounterACT Property List.

For more information about CounterACT Property List, refer to the Defining and Managing Lists section in the CounterACT Console User Manual.

**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 CounterACT Property Lists are updated. The message body is saved to the `update.xml` file, which is listed below.

```
# /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.

```
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 Construction**

DEX module allows users to configure a parsing pattern to map HTTP response content (typically in JSON/XML format) to property as part of External Web Service Property definition. The module 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 CounterACT 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 CounterACT, external platforms can submit REST messages to the CounterACT web service. See Working with the CounterACT Web Service

The purpose of the table below is to provide the 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 given in the simplest way. In order to write more
complex and powerful parsing patterns, the user is expected to have a good understanding of the JSON Path/XML Path/REGEX. Note that:

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

For information about other CounterACT features and modules, refer to the following resources:

- Documentation Downloads
- Documentation Portal
- CounterACT Help Tools

Documentation Downloads

Documentation downloads can be accessed from one of two ForeScout portals, depending on which licensing mode your deployment is using.

- **Per-Appliance Licensing Mode** - Product Updates Portal
- **Centralized Licensing Mode** - Customer Portal

Software downloads are also available from these portals.

To learn which licensing mode your deployment is using, see [Identifying Your Licensing Mode in the Console](#).

Product Updates Portal

The Product Updates Portal provides links to CounterACT version releases, Base and Content Modules, and Extended Modules, as well as related documentation. The portal also provides a variety of additional documentation.

**To access the Product Updates Portal:**

2. Select the CounterACT version you want to discover.

Customer Portal

The Downloads page on the ForeScout Customer Portal provides links to purchased CounterACT version releases, Base and Content Modules, and Extended Modules, as well as related documentation. Software and related documentation will only appear on the Downloads page if you have a license entitlement for the software. The Documentation page on the portal provides a variety of additional documentation.

**To access documentation on the ForeScout Customer Portal:**

2. Select Downloads or Documentation.

Documentation Portal

The ForeScout Documentation Portal is a searchable, web-based library containing information about CounterACT tools, features, functionality and integrations.
If your deployment is using Centralized Licensing Mode, you may not have credentials to access this portal.

**To access the Documentation Portal:**

2. Use your customer support credentials to log in.
3. Select the CounterACT version you want to discover.

**CounterACT Help Tools**

Access information directly from the CounterACT Console.

*Console Help Buttons*

Use context sensitive Help buttons to quickly access information about the tasks and topics you are working with.

*CounterACT Administration Guide*

Select CounterACT Help from the Help menu.

*Plugin Help Files*

1. After the plugin is installed, select Options from the Tools menu and then select Modules.
2. Select the plugin and then select Help.

*Documentation Portal*

Select Documentation Portal from the Help menu.

**Identifying Your Licensing Mode in the Console**

If your Enterprise Manager has a ForeScout CounterACT See license listed in the Console, your deployment is operating in Centralized Licensing Mode. If not, your deployment is operating in Per-Appliance Licensing Mode.

Select Options > Licenses to see whether you have a ForeScout CounterACT See license listed in the table.

Contact your ForeScout representative if you have any questions about identifying your licensing mode.
Legal Notice

© 2018 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.

2018-07-12 16:19