Change Notification Feature (CNF) for Administration XML (AXL) Clients

Table of Contents

Abstract 2
Background 2
Solution 2
Details 2
Sample First Request: 2
Sample First Response 3
Sample Subsequent Request 3
Sample Subsequent Response 4
Pagination Example 5
Polling Interval 5
Error Conditions 6

Abstract

Administration XML (AXL) clients are dependent upon Unified CM (CUCM) data to function properly. Today AXL applications need to periodically re-sync to ensure they have the latest information. The AXL Change Notification Feature (CNF) is designed to help keep AXL applications informed when dependent data changes. This document is intended to show AXL Developers how to add the CNF to their applications.

Background

The AXL API's for introduced in release 9.0 for change notification were complex and inefficient; requiring clients to subscribe to and acknowledge changes. As we attempted to expand notification to support all AXL objects, we quickly learned the previous design would not scale and that maintaining separate metadata would be problematic. The AXL Change Notification Feature was re-designed in Unified CM 10.0.

The main goals of the new design were to simplify the process for AXL clients to obtain changes and to minimize the server load. The end result is a single API called listChange that AXL clients can call as often as desired.

Solution

AXL will record all changes for objects exposed within the interface library; no subscriptions are required. Changes are transformed from the database table / columns to the AXL object and stored in a cache. The AXL client obtains the list of changes by calling the listChange interface which returns all of the changes in the cache.

Details

Sample First Request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns="http://www.cisco.com/AXL/API/10.0">
<soapenv:Header/>
        <soapenv:Body>
                <ns:listChange>
                        <objectList>        (optional)
                                <object>Phone</object>
                                <object>User</object>
                        </objectList>
                </ns:listChange>
        </soapenv:Body>
</soapenv:Envelope>
```

The first time the client calls the listChange API, AXL returns all of the change notifications in the cache. AXL clients interested in changes for a specific set of objects should always specify the objectList. Use of the objectList tailors the response and reduces the overall load on the Publisher. There is no limit to the

number of objects which can be specified in the objectList. If the objectList is not specified, AXL will return the list of changes for all objects in the cache.

Sample First Response

Each time a client calls listChange, the response contains two sections. The first section is called queueInfo which contains: firstChangeID, lastChangeID, nextStartChangeID, and queueID.

This sample response shows AXL returned changes 1 to 10,000. The client is expected to ask for change 10,001 the next time it makes a request. Additionally, the queueID must also be submitted. queueID helps the client detect if changes may have been lost between sync intervals. Note that changes 2-10,000 are not shown in this example.

Example:

<queueInfo>
<firstChangeId>1</firstChangeId> **(The first (or oldest) change available in the cache)**
<lastChangeId>10000</lastChangeId> **(The last (or newest) change available in the cache)**
<nextStartChangeId>10001</nextStartChangeId> **(The next change record ID clients should specify in subsequent listChange requests)**
<queueId>093249823498</queueId> **(A unique cache identifier which changes every time the AXL service is restarted)**
</queueInfo>

The second section contains a list of changes.

<change type=' physicalLocation' uuid='8h58f047-7b40-4547-a8c2-fc5b2b668b7f'>
<id>1</id> **(The changeId AXL assigned)**
<action>u</action> **(Indicates the type change: u is update, a is add, r is remove)**
<doGet>false</doGet> **(Indicates when the client should perform a Get operation for the object, typically because the object is new or when multiple attributes of the object were changed)**
<changedTags>
<changedTag name = 'description'>A__V3.Dx.3 w_p6.'</changedTag> **(Indicates what changed)**
</changedTags>
</change>

Sample Subsequent Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns="http://www.cisco.com/AXL/API/10.0">
<soapenv:Header/>
        <soapenv:Body>
                <ns:listChange>
                        <startChangeId queueId='093249823498'>10001</startChangeId>
                        <objectList>          (optional)
                                <object>Phone</object>
                                <object>User</object>
                        </objectList>
                </ns:listChange>
        </soapenv:Body>
</soapenv:Envelope>
```

The first time the client calls the listChange API, the <startChangeId> is not included. However, every subsequent listChange request should include both a startChangeID and queueID. The nextStartChangeId value received in the first response should be submitted as the startChangeID in subsequent requests. Additionally, the queueId received in the first response must also be included to help the client detect if changes may have been lost between sync intervals. Anytime the AXL service is restarted, a new queueID is generated.

Sample Subsequent Response

This sample response shows AXL returned changes from 10,001 to 20,000 for queueID 093249823498. The client should ask for change 20,001 with the same queueID the next time it makes a request. Note that changes 10,005-20,000 are not shown in this example.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns="http://www.cisco.com/AXL/API/10.0">

<soapenv:Body>
        <ns:listChangeResponse>
                <queueInfo>
                        <firstChangeId>10001</firstChangeId>
                        <lastChangeId>20000</lastChangeId>
                        <nextStartChangeId>20001</nextStartChangeId>
                        <queueId>093249823498<queueId>
                </queueInfo>
                <changes>
                        <change type='Phone' uuid='4c48f047-7b40-4547-a8c2-fc5b2b668bda'>
                                <id>10001</id>
                                <action>a</action>
                                <doGet>true</doGet>
                        </change>
...
                        <change type='PhysicalLocation' uuid='8h58f047-7b40-4547-a8c2-fc5b2b668b7f'>
                                <id>10002</id>
                                <action>u</action>
                                <doGet>false</doGet>
                                <changedTags>
                                        <changedTag name = 'description'>A__V3.Dx.3 w_p6.</changedTag>
                                </changedTags>
                        </change>
...
                        <change type='PhysicalLocation' uuid='8h58f047-7b40-4547-a8c2-fc5b2b668b7f'>
                                <id>10003</id>
                                <action>r</action>
                                <doGet>false</doGet>
                        </change>
                        <change type='Phone' uuid='8g48f047-7b40-4547-a8c2-fc5b2b668b8d'>
                                <id>10004</id>
                                <action>u</action>
                                <doGet>false</doGet>
                                <changedTags>
                                        <changedTag name = 'name'>SEP00000001</changedTag>
                                        <changedTag name = 'description'>A__V.x.3 w_p6.</changedTag>
                                        <changedTag name = 'versionStamp'>815abf2-1c0e-4'</changedTag>
                                </changedTags>
```

```
                    </change>
                </changes>
            </ns:listChangeResponse>
</soapenv:Body>
</soapenv:Envelope>
```

The <doGet> tag indicates when the client needs to perform a get <object> operation, typically when the object is newly added or when too many attributes for the object changed. In this example the client should perform a getPhone request for uuid 4c48f047-7b40-4547-a8c2-fc5b2b668bda'.

Note: There may be multiple results for the same object. In this example the 'PhysicalLocation' attribute for phone with uuid '8h58f047-7b40-4547-a8c2-fc5b2b668b7f' was updated, then removed.

Pagination Example

The client should always ask for the nextStartChangeId whether or not the request was paged, so the client doesn't really need to detect if the request was paged. If lastChangeId >= to nextStartChangeId then the request was paged. The client can immediately request the nextStartChangeId.

```
<queueInfo>
        <firstChangeId>100</firstChangeId>
        <lastChangeId>10000</lastChangeId>
        <nextStartChangeId>1001</nextStartChangeId>
        <queueId>093249823498<queueId>
</queueInfo>
```

Polling Interval

The new AXL CNF has no restrictions on polling intervals. Requests are still subject to existing AXL throttles which limit each response to 8MB and all concurrent responses to 16MB.

The change queue cache is stored in memory and is limited to 100,000 changes. The cache can fill quickly depending on the types of changes performed. For example, if an XSI (IP Phone) Service has been configured for 10,000 phones and the service is deleted, the cache will include one entry representing the deletion of the service plus 10,000 phone updates indicating the service was removed from each device.

For a typical cluster, Cisco recommends AXL clients request listChange every 1-15 minutes. On a very busy system, AXL clients should request changes often (between 1-5 minutes) to ensure all changes are communicated while still in the cache. If the client waits too long and the changeID requested is no longer in the queue, AXL will return an error.

Error Conditions

Given the following queueInfo, here are some common error conditions:

```
<queueInfo>
        <firstChangeId>154</firstChangeId>
        <lastChangeId>10000</lastChangeId>
        <nextStartChangeId>10001</nextStartChangeId>
        <queueId>093249823498<queueId>
</queueInfo>
```

1) Asking for a change that is no longer in the queue:

```
<ns:listChange>
        <startChangeId queueId='093249823498'>153</startChangeId>
</ns:listChange>
```

2) Asking for a change that is greater than the nextStartChangeId in the queue:

```
<ns:listChange>
        <startChangeId queueId='093249823498'>10002</startChangeId>
</ns:listChange>
```

3) Using an invalid or old queueId:

```
<ns:listChange>
        <startChangeId queueId='foobar'>10001</startChangeId>
</ns:listChange>
```