Cisco Networkers 2008

Extending Cisco Unified Communications Manager using the Administrative XML/SOAP (AXL) API

Special thanks to author

Johannes Krohn

Abstract

The SOAP based Administrative XML (AXL) interface of the Cisco Unified Communications Manager provides a unique and flexible basis to create customer specific tools to allow for more efficient deployment, management and operation of Unified Communications deployments. This session will give a quick overview of the fundamentals of the AXL interface, including very basic example scripts that use the AXL interface to carry out management tasks. The ultimate goal of this session is to drive the adoption of the AXL interface and create the confidence that given a set of examples and a basic framework everyone can create useful scripts to solve some of the challenges faced in day-2-day operations.

Pre-requisites

Readers should have a solid understanding of Cisco Unified Communications Manager configuration and operations.

Agenda

- Concepts
- AXL APIs
- Documentation
- How to enable AXL
- Troubleshooting
- AXL Messages
- AXL Versioning
- Database Access
- Scripting / Automation
- Summary

Concepts

XML, WSDL, SOAP etc.

XML

- eXtensible Markup Language
- W3C recommendation
- restricted form of SGML (Standard Generalized Markup Language, ISO 8879)
- general purpose markup language
- extensible; individual tags can be defined
- W3C specifies grammar and parsing requirements
- encode documents and serialize data
- XML 1.0 (4th edition), 16 August 2006 http://www.w3.org/TR/2006/REC-xml-20060816/

XML, well-formed document

- document conforms to all syntax rules
- e.g. opening/closing tag for elements
- not validated against schema
- example:

```
<person>
      <lastname>Krohn</lastname>
      <givenname>Johannes</givenname>
</person>
```

XML, special characters

- some characters can´t be used in XML
- solution: escape or numerical representation
- escape:

  & & ampersand

  < < less than

  > > greater than

  ' ´ apostrophe

  " “ quotation mark

  example: <company>AT&T<company>

- numerical representation of Unicode codepoint

  example: ™= ™

XML, semantics

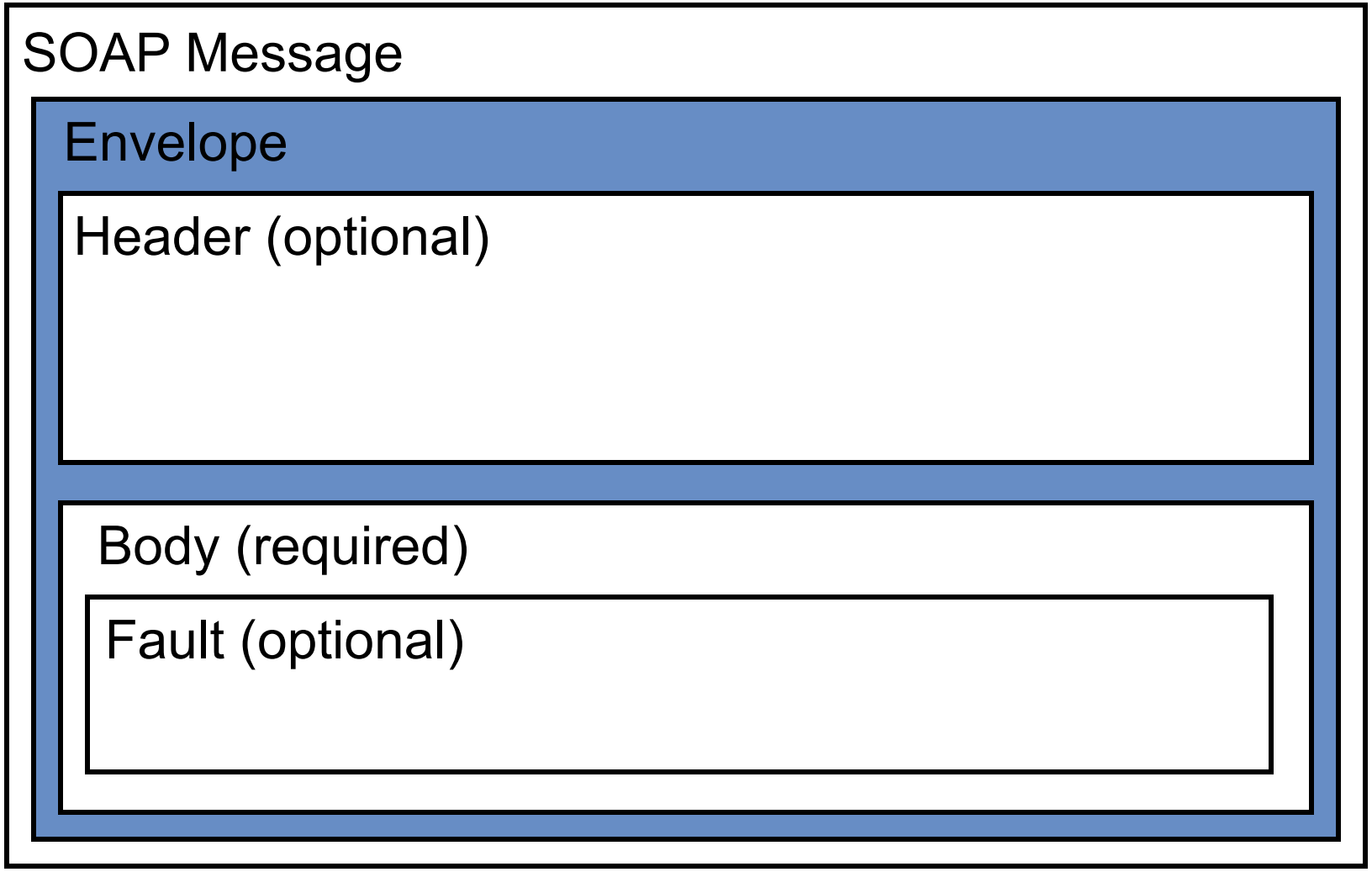
- names, hierarchy, meaning of elements and attributes defined by schema
- XML Schema defined by W3C
- Primer: http://www.w3.org/TR/xmlschema-0/
- Example:

```
<xsd:element name="person" type="PersonType"/>
<xsd:complexType name="PersonType">
     <xsd:sequence>
            <xsd:element name="lastname" type="xsd:string"/>
            <xsd:element name="givenname" type="xsd:string"/>
     </xsd:sequence>
</xsd:complexType>
```

SOAP

- formerly known as “Simple Object Access Protocol”
- W3C specification: http://www.w3.org/TR/soap/
- exchange of structured and typed information based on XML; XML infoset
- SOAP spec. defines
  - SOAP message format
  - How to send and receive messages
  - Data encoding
- can be used for remote procedure calls (RPC)

SOAP Message



RPC requirements

- RPC requires:
  - address of SOAP node
  - procedure/method name
  - identities/values of arguments
  - output parameters, return values
- interface/service definition not part of SOAP

WSDL

- Web Services Definition Language
- W3C: http://www.w3.org/TR/wsdl20/
- XML based format (grammar) to describe Web Services
- defines four pieces of data:

  publicly available methods; interface description, formats

  data type information for requests and responses

  binding; which transport protocol

  address information; where to find the service

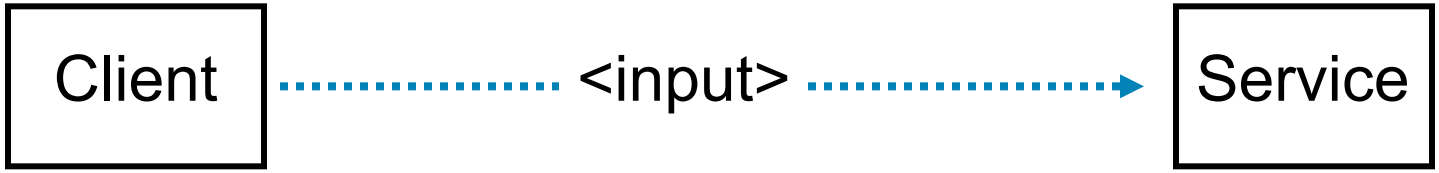
WSDL service description elements

definitions: root element

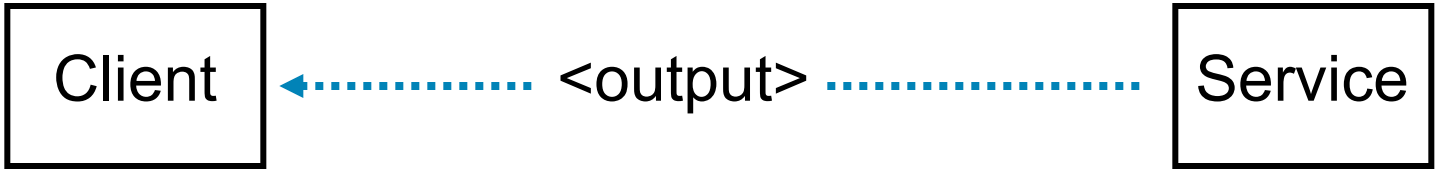
- types: which datatypes are exchanged?
- message: which messages are exchanged?
- portType: what operations/functions exist?
- binding: message exchange; soap specifics
- service: location of the service

WSDL operation patterns

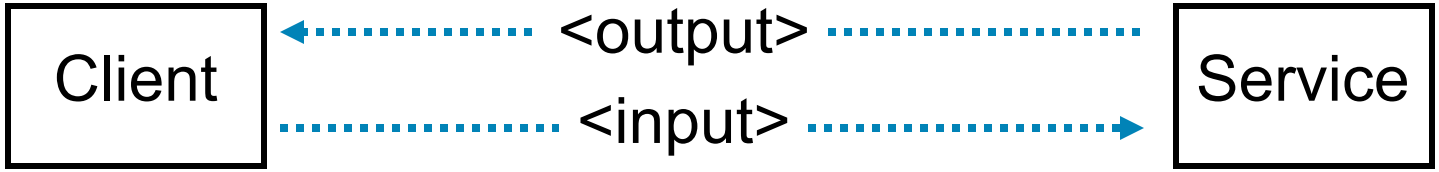
one-way



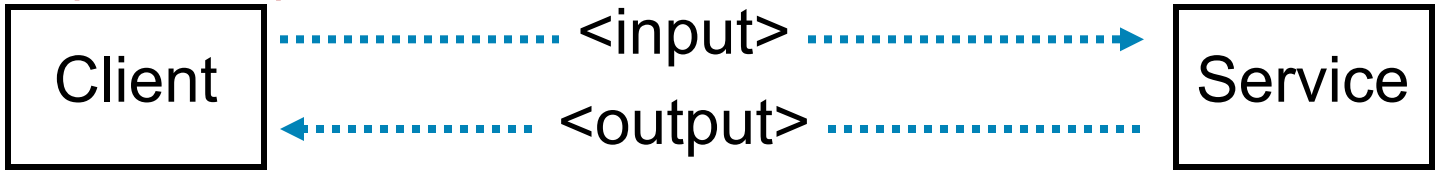
notification



solicit/response



request/response



AXL APIs

AXL

- AXL = AVVID XML Layer
- RPCs to the Unified Communications Manager
- Interface defined using WSDL
- SOAP message exchange via HTTP(s)

AXL configuration API

- read/modify configuration database
- methods

  list*

  add*

  update*

  get*

  remove*
- includes methods for direct database access
- Service port: https://<server>:8443/axl/

Other AXL interfaces

- Perfmon service WSDL:
  `https://<server>:8443/perfmonservice/services/PerfmonPort?wsdl`
- Real-time information service WSDL:
  `https://<server>:8443/realtimeservice/services/RisPort?wsdl`
- Log collection service WSDL:
  `https://<server>:8443/logcollectionservice/services/LogCollectionPort?wsdl`
- DIME get file service WSDL:
  `https://<server>:8443/logcollectionservice/services/DimeGetFileService?wsdl`
- Control Center services WSDL:
  `https://192.168.121.6:8443/controlcenterservice/services/ControlCenterServicesPort?wsdl`
- SOAP Monitor WSDL:
  `https://192.168.121.6:8443/realtimeservice/services/SOAPMonitorService?wsdl`
- CDR on demand WSDL:
  `https://192.168.121.6:8443/CDRonDemandService/services/CDRonDemand?wsdl`

Documentation

Developer Support

- http://developer.cisco.com/web/axl/home
- “Documents”
- you will get
  - Data Dictionary
  - XML Developers Guide
  - AXL Interface Specification (HTML)

Documentation on Cisco.com

- http://www.cisco.com/en/US/products/sw/voicesw/ps556/products_programming_reference_guides_list.html
- Products, Communications Manager, Configure, Programming Guides
- you will get

  XML Developer Guide

  JTAPI Developer Guide

  TAPI Developer Guide

  Data Dictionary

  ...

View documents by topics: Choose Topic

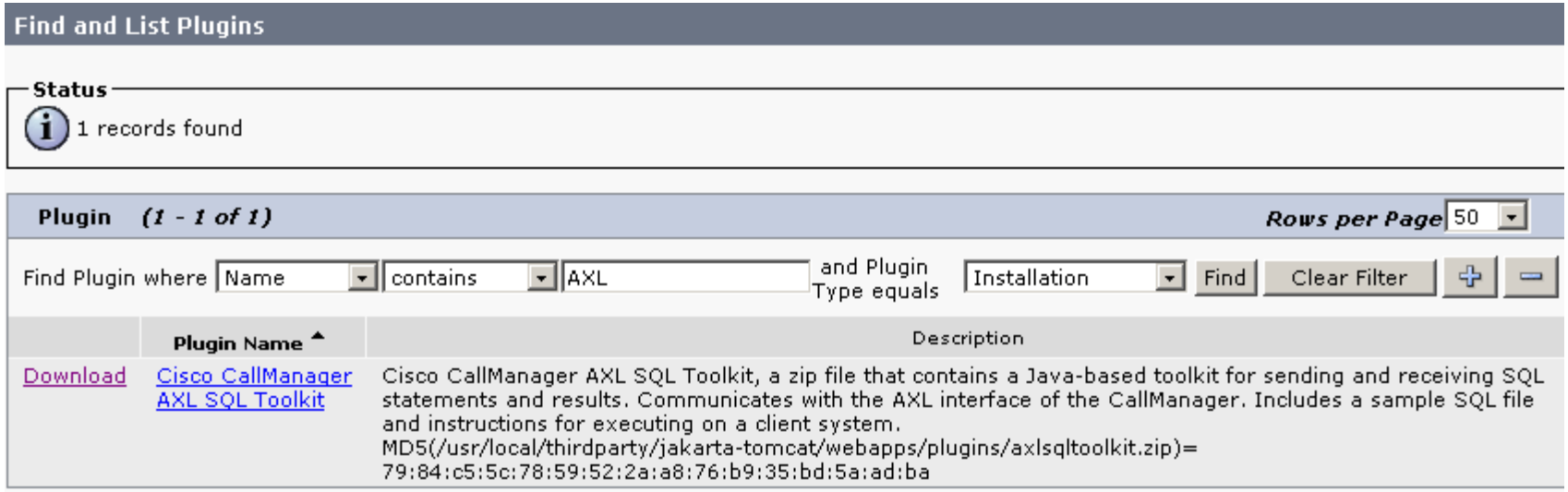
Cisco Unified Communications Manager Version 6.0

Cisco Unified CallManager Data Dictionary, Release 6.0(1) (PDF - 10 MB)
Cisco Unified Communications Manager XML Developers Guide for Release 6.0(1)
Cisco Unified Communications Manager JTAPI Developers Guide
Cisco Unified Communications Manager TAPI Developers Guide
Cisco JTAPI JavaDoc -- zip archive (550 KB)
Cisco Unified IP Phone Services Application Development Notes, Release 6.0(1)
SIP Line Messaging Guide (Standard) for Release 6.0(1) (PDF - 3 MB)

AXL documentation on the server

- Cisco Unified CM AXL SQL Toolkit is available in the Plugin list
- contains complete schema definition:

  AXLAPI.wsdl, AXLEnums.xsd, axlmessage.xsd, axlsoap.xsd, axl.xsd



How to enable AXL

Service Activation

In Communications Manager Serviceability

Database and Admin Services

	Service Name	Activation Status
☑	Cisco AXL Web Service	Activated
☑	Cisco Bulk Provisioning Service	Activated
☐	Cisco TAPS Service	Deactivated

CDR Services

	Service Name	Activation Status
☑	Cisco SOAP - CDRonDemand Service	Deactivated
☐	Cisco CAR Scheduler	Deactivated
☐	Cisco CAR Web Service	Deactivated

AXL Authorization

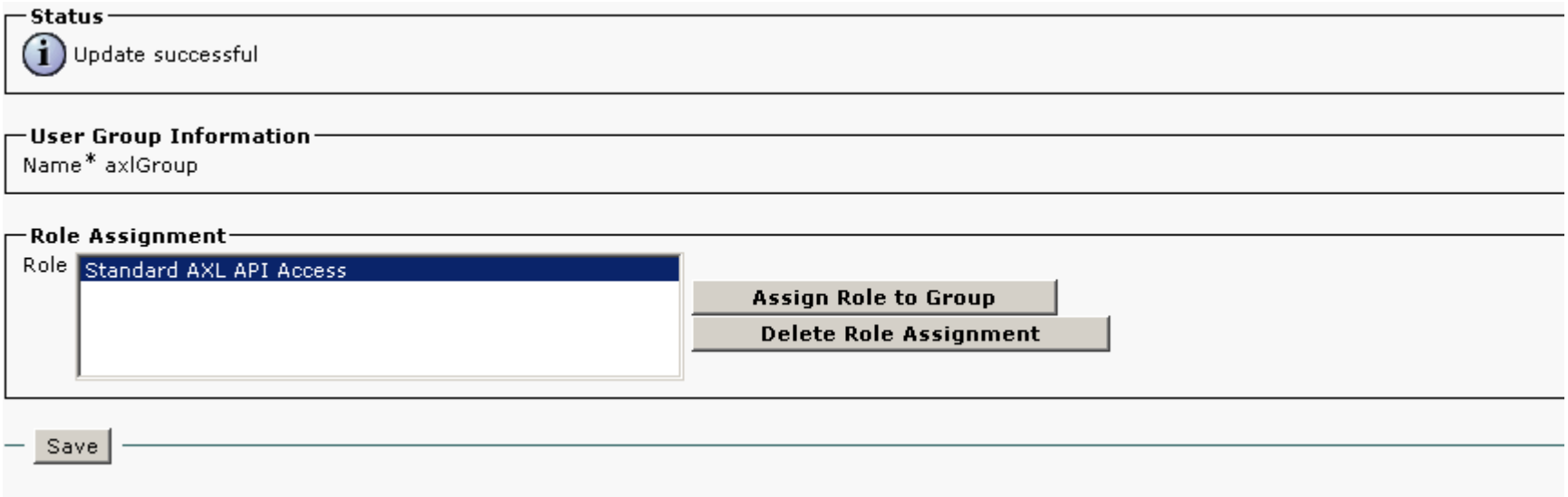
- All AXL requests have to be authorized
- AXL requests are authorized using HTTPS basic authorization
- Authorization header with <user>:<password> in BASE64 coding

```
Authorization: Basic YXhsVXNlcjpjaXNjbw==
```

- BASE64 coding can EASILY be decoded
- Authorization secure only because HTTPS is used
- Don´t use “CCMAdministrator”!
- Dedicated application user should be used instead
- AXL API access is a dedicated role in Communications Manager
- User for AXL API access can/should be limited to AXL API access only

Dedicated user for AXL access

- Create special application user for AXL access
- Create User Group for AXL access
- Put AXL user in this user group
- User Group needs Role „Standard AXL API Access“



Rate Control

- All the performance and Real-time monitoring queries should be polled at the rate that should not affect the Call processing performance
- Admin can configure the system level rate that is acceptable in Call Manager environment
- If incoming request rate are exceeded then request will be dropped and slow down responses are sent to appropriate clients making the request

Select Server and Service

Server* 192.168.121.6 (Active)

Service* Cisco Database Layer Monitor (Active)

All parameters apply only to the current server except parameters that are in the Clusterwide group(s).

Cisco Database Layer Monitor (Active) Parameters on server 192.168.121.6 (Active)

Parameter Name	Parameter Value	Suggested Value
Clusterwide Parameters (Parameters that apply to all servers)		
Device Name Validation *	True	True
Maintenance Time *	24	24
Maintenance Window *	2	2
Maximum AXL Writes Allowed per Minute *	50	50

Rate Control

- if configured rate is exceeded the server will send a HTTPS 503 Service unavailable response
- These requests are not throttled:

  executeSQLQuery

  doDeviceReset

  all „get“ requests

  all „list“ requests
- executeSQLUpdate is throttled
- Beware: Excessive use of the API might have negative impact on the call control performance

Troubleshooting

Quick Functionality Check

- Go to the AXL API URL via a web browser
- For instance, enter https://cm1:8443/axl/ in the address text box
- When prompted for user name and password, use the standard administrator login, or use the configured AXL user
- Look for a plain page that states the AXL listener is working and accepting requests, but only communicates via POST

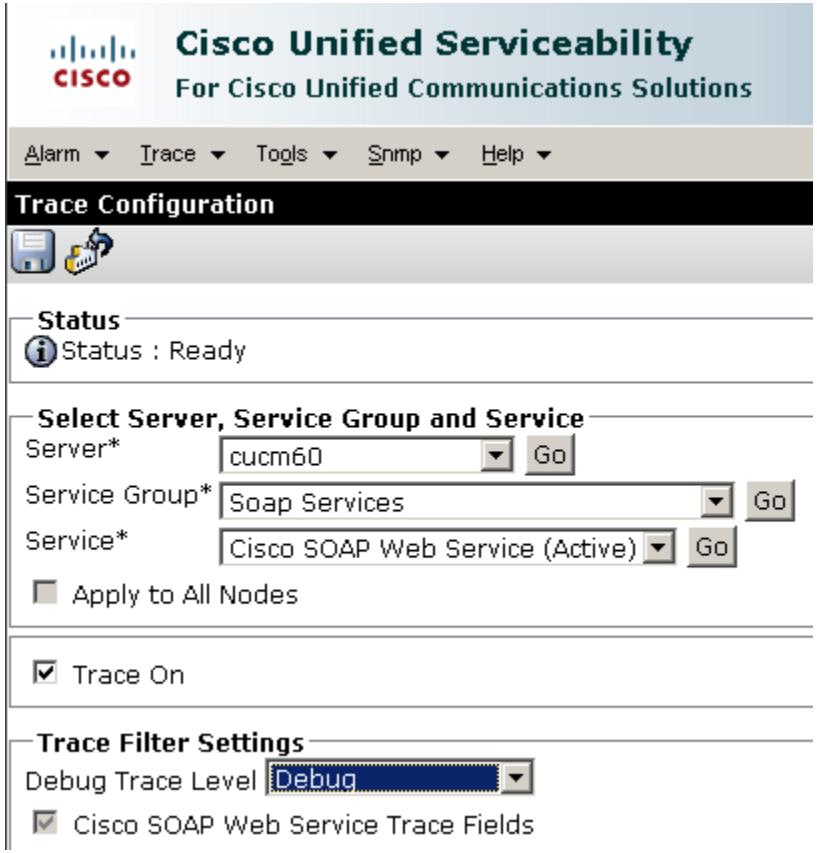
Cisco CallManager: AXL Web Service

The AXL Web Service is working and accepting requests. Use HTTP POST to send a request.

- This verifies functionality and user access

Enable SOAP Traces

- detailed SOAP traces can be enabled in Cisco Unified Serviceability settings



Analyze SOAP logs

- Use Real-Time Monitoring Tool to access SOAP log
- log contains incoming SOAP requests and outgoing responses

```
INFO  [http-8443-Processor24] axl.AxlListener - Received request 1195307341505 from
admin at IP 192.168.121.1
INFO  [http-8443-Processor24] axl.AxlListener - <SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"><SOAP-ENV:Body><axl:getCCMVersion
xmlns:axl="http://www.cisco.com/AXL/1.0"></axl:getCCMVersion></SOAP-ENV:Body></SOAP-
ENV:Envelope>
INFO  [http-8443-Processor24] axl.Handler - Handler initializing
INFO  [http-8443-Processor24] axl.AxlListener - <?xml version="1.0"
        encoding="UTF-8"?><SOAP-ENV:Envelope
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <SOAP-ENV:Header/>
        <SOAP-ENV:Body><axl:getCCMVersionResponse
                xmlns:axl="http://www.cisco.com/AXL/API/1.0"
                xmlns:xsi="http://www.cisco.com/AXL/API/1.0">
        <return><componentVersion>
        <version>6.0.1.2107(1)</version></componentVersion></return>
        </axl:getCCMVersionResponse>
        </SOAP-ENV:Body></SOAP-ENV:Envelope>
INFO  [http-8443-Processor24] axl.AxlListener - Request 1195307341505 was process in
791ms
```

AXL messages

AXL messages

- AXL message is a SOAP message
- contains:

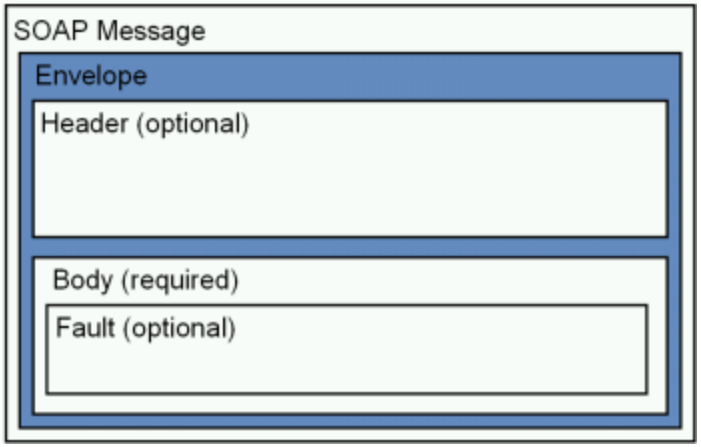
  header (HTTPS)

  SOAP envelope

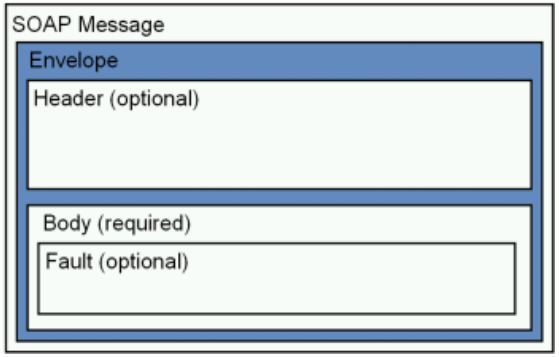
  (SOAP header)

  SOAP body

  (SOAP fault)



AXL message format



Header

```
POST: 8443/axl/
Host: axl.myhost.com:8443
Accept: text/*
Authorization: Basic bGFycnk6Y3VybHkgYW5kIG1vZQ==
Content-type: text/xml
SOAPAction: "CUCM:DB ver=6.1"
Content-length: 613
```

Basic Authorization

Content Type

Versioning

Envelope

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV=
                      "http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

Body

```
<SOAP-ENV:Body>
<axl:getPhone xmlns:axl="http://www.cisco.com/AXL/1.0"
xsi:schemaLocation="http://www.cisco.com/AXL/1.0
                    http://ccmserver/schema/axlsoap.xsd"
                    sequence="1234">
<phoneName>SEP222222222245</phoneName>
</axl:getPhone>
</SOAP-ENV:Body>
```

```
</SOAP-ENV:Envelope>
```

SOAP Messages

- Graphical representation of schema available as part of Developer Documentation
- Contains all elements, complex and simple types
- For every request there are two elements: the actual request and a response; example addAARGroup and addAARGroupResponse

Schema **axlsoap.xsd**

schema location: **C:\Documents and Settings\adjoshi.APAC\Desktop\SPEC\axlsoap.xsd**
targetNamespace: **http://www.cisco.com/AXL/API/1.0**

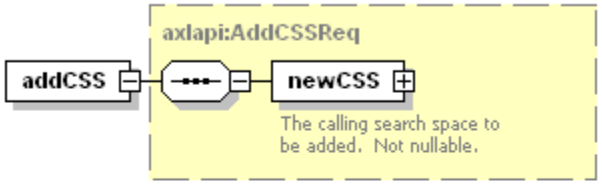
Elements	Complex types
addAARGroup	addAARGroupReq
addAARGroupResponse	AddApplicationToSoftkeyTemplateReq
addApplicationToSoftkeyTemplate	AddAttendantConsoleHuntGroupReq
addApplicationToSoftkeyTemplateResponse	AddAttendantConsoleUserReq
addAttendantConsoleHuntGroup	AddCallerFilterListReq
addAttendantConsoleHuntGroupResponse	AddCallManagerGroupReq

Legend

	Element Name	Description
addCallManager	Complex Element	Can have childs and attributes. Solid border = mandatory
name	Simple Element	No childs, only attributes. Solid border= mandatory
description	Optional	Optional. Any type of element can be optional
	Sequence	All children must appear in order
	Choice	Only one of the children can appear

Example: addCSSReq

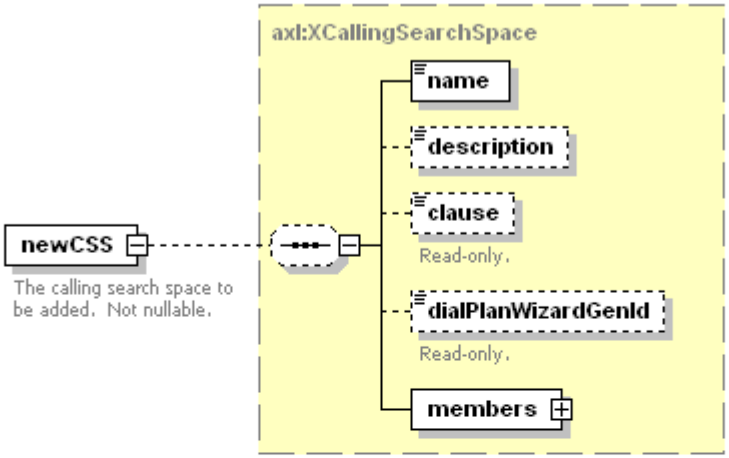
element **addCSS**

diagram	
namespace	http://www.cisco.com/AXL/API/1.0
type	**axlapi:AddCSSReq**
children	**newCSS**
attributes	Name: sequence; Type: xsd:unsignedLong; Use: optional; Default:
source	`<xsd:element name="addCSS" type="axlapi:AddCSSReq"/>`

```
<axl:addCSS>
</axl:addCSS>
```

Example: addCSSReq

element **AddCSSReq/newCSS**

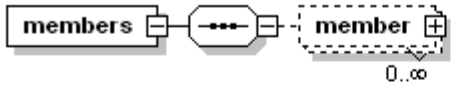
diagram	
type	**axl:XCallingSearchSpace**
children	**name description clause dialPlanWizardGenId members**
attributes	Name Type Use Default Fixed Annotation uuid axl:XUUID
annotation	documentation The calling search space to be added. Not nullable.

source	`<xsd:element name="newCSS" type="axl:XCallingSearchSpace" nillable="false">`
	`<xsd:annotation>`
	`<xsd:documentation>`The calling search space to be added. Not nullable.`</xsd:documentation>`
	`</xsd:annotation>`
	`</xsd:element>`

```
<axl:addCSS>
<name>test</name>
<members>
</members>
</axl:addCSS>
```

Example: addCSSReq

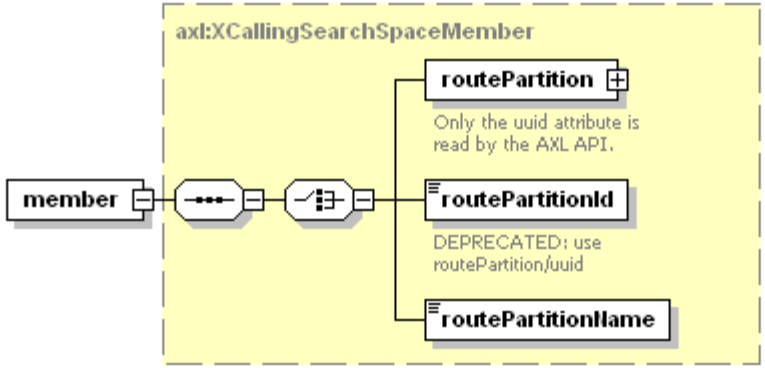
element **axl:XCallingSearchSpace/members**

diagram	
children	**member**
source	<xsd:element name="members"> <xsd:complexType> <xsd:sequence> <xsd:element name="member" type="axl:XCallingSearchSpaceMember" minOccurs="0" maxOccurs="unbounded"/> </xsd:sequence> </xsd:complexType> </xsd:element>

```
<axl:addCSS>
<name>test</name>
<members>
  <member>
  </member>
  <member>
  </member>
</members>
</axl:addCSS>
```

Example: addCSSReq

element **axl:XCallingSearchSpace/members/member**

diagram	
type	**axl:XCallingSearchSpaceMember**
children	**routePartition routePartitionId routePartitionName**
attributes	Name: index; Type: xsd:positiveInteger; Use; Default; Fixed; Annotation
source	`<xsd:element name="member" type="axl:XCallingSearchSpaceMember" minOccurs="0" maxOccurs="unbounded"/>`

```
<axl:addCSS>
<name>test</name>
<members>
  <member>
    <index>1</index>
    <routePartitionName>
      Partition1
    </routePartitionName>
  </member>
  <member>
    <index>2</index>
    <routePartitionName>
      Partition2
    </routePartitionName>
  </member>
</members>
</axl:addCSS>
```

Example: addCSSReq

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body>
<axl:addCSS xmlns:axl="http://www.cisco.com/AXL/1.0">
    xsi:schemaLocation="http://www.cisco.com/AXL/1.0 http://ccmserver/schema/axlsoap.xsd" sequence="1234">
    <name>test</name>
    <members>
        <member>
            <index>1</index>
            <routePartitionName>Partition1</routePartitionName>
        </member>
        <member>
            <index>2</index>
            <routePartitionName>Partition2</routePartitionName>
        </member>
    </members>
</axl:addCSS>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

AXL versioning

AXL versioning

- changes in Communications Manager functionality drive AXL schema changes
- schema change might require changes in components using the API
- Solution: starting with release 6.1 every communications manager will support several versions of the API
- every release will still support the releases of the previous major release
- requestor defines version to use via the “SOAPAction” header; e.g.:
  `SOAPAction: "CUCM:DB ver=6.0"`
- SOAPAction header currently is optional; if missing request will be treated as a 6.0 request
- SOAPAction header will be mandatory in future releases
- Access to not supported version will lead to:
  `599, „The version you specified is not available. Available versions are 6.0, 6.1 and 1.0”`

Database Access

Database Dictionary

- configuration in Communications Manager is stored in relational database
- dictionary documents all existing tables in Communications Manager Database
  - field types
  - database constraints
  - relations
- Common Table Relationships
- Schema changes in recent releases

Table relations (1)

- pkid is the primary key ID. It is always of type GUID.
- Fields that begin with the letters "fk" represent foreign keys into another table. The name of the field following the "fk" prefix up to but not including an underscore character is the name of the related table. The field in related table is always pkid. and is a GUID.
- Examples in table device:

  device.fkenduser → enduser.pkid

  device.fkenduser_mobility → enduser.pkid

  device.fkcallingsearchspace → callingsearchspace.pkid

  device.fkcallingsearchspace_aar → callingsearchspace.pkid

Table relations (2)

- Fields that begin with the letters "ik" represent internal keys into the same table.
- Example in table device:

  device.ikdevice_primaryphone → device.pkid

Table relations (3)

- Fields that begin with a "tk" represent an enumerated type. This field is related to a table whose name begins with "Type" and ends with the name of the field following the prefix up to but not including an underscore character. The field in the related table is always "enum" and is an integer.
- Examples in table device

  tkclass → typeclass.enum

  tkdeviceprotocol → typedeviceprotocol.enum

  tkmodel → typemodel.enum

  tkproduct → typeproduct.enum

SQL

- language for retrieval and management of data stored in relational database management systems
- originally called SEQUEL (structured english query language)
- standardized by ISO/IEC

SQL Statements on the CLI

- SQL statements can be executed on the CLI using „run sql“
- “&” can´t be used on the CLI
- Can be used to test SQL statements to be used in scripts
- Example:

```
admin:run sql select enum,name from typemodel where
tksubclass=1

enum  name

===== ====================================

20    SCCP Phone

134   Remote Destination Profile

30027 Analog Phone

30028 ISDN BRI Phone

2     Cisco 12 SP+

3     Cisco 12 SP

...
```

Example: dialplan

- All DNs and patterns are stored in table numplan

```
admin:run sql select dnorpattern, fkroutepartition, tkpatternusage
from numplan

dnorpattern       fkroutepartition                     tkpatternusage
================= ==================================== ==============
8496.XXXX         c86ff8a5-61d5-abaf-e56c-c79a2e366636 5
8496.XXXX         4a7e83fc-64f9-f7e9-098c-95aefb3de9b6 5
000496196773.XXXX c86ff8a5-61d5-abaf-e56c-c79a2e366636 5
000496196773.XXXX 4a7e83fc-64f9-f7e9-098c-95aefb3de9b6 5
006196773.XXXX    c86ff8a5-61d5-abaf-e56c-c79a2e366636 5
006196773.XXXX    4a7e83fc-64f9-f7e9-098c-95aefb3de9b6 5
9765              f5da0288-c206-a2b8-2584-c07530e2bf2d 2
```

- Which partition? What type of pattern?
- Let's look in tables routepartition and typepatternusage

Example: dialplan

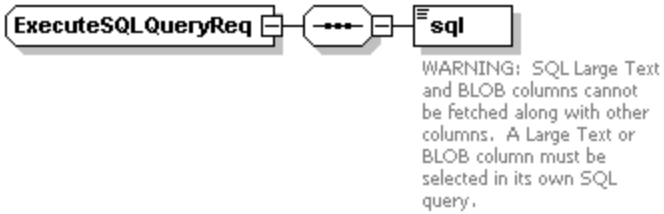
```
admin:run sql select dnorpattern, routepartition.name,
typepatternusage.name from numplan,routepartition,typepatternusage
where fkroutepartition=routepartition.pkid and
tkpatternusage=typepatternusage.enum

dnorpattern       name        name
================= =========== ======
9765              allPhones   Device
9766              allPhones   Device
9767              allPhones   Device
9768              allPhones   Device
9769              allPhones   Device
9780              allPhones   Device
9781              allPhones   Device
9782              allPhones   Device
9783              allPhones   Device
9784              allPhones   Device
9785              allPhones   Device
9786              allPhones   Device
8496.XXXX         100sites    Route
000496196773.XXXX 100sites    Route
006196773.XXXX    100sites    Route
8496.XXXX         100sitesNOv Route
000496196773.XXXX 100sitesNOv Route
006196773.XXXX    100sitesNOv Route
```

- Assignment of DNs to devices is in table devicenumplanmap

Database access via AXL

- AXL provides methods to execute SQL queries and updates:
- executeSQLQuery (SELECT)
- executeSQLUpdate (INSERT, UPDATE, DELETE)
- both methods take a SQL command as argument

executeSQLQuery



- Result is a sequence of rows
- each row has a number of sub-elements, one per column of the resulting table

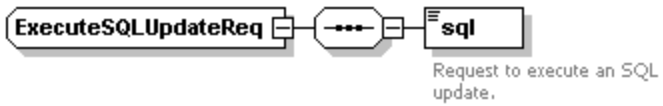
```
<SOAP-ENV:Envelope ...>
<SOAP-ENV:Header/>
<SOAP-ENV:Body><axl:executeSQLQueryResponse ...>
<return>
     <row>
            <pkid>8555d448-5818-8494-e16a-de099e9a403c</pkid>
            <realm>jkrohn</realm>
            <userid>jkrohn</userid>
            <passwordreverse>...</passwordreverse>
     </row>
</return>
</axl:executeSQLQueryResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

- sequence can be empty!

```
<SOAP-ENV:Body><axl:executeSQLQueryResponse ...>
<return/>
</axl:executeSQLQueryResponse>
</SOAP-ENV:Body>
```

!!!

executeSQLUpdate



- Writing to the database can destroy database integrity and thus compromise core functionality!
- Very limited to no integrity checks!
- delete means deleted ☺
- result is an element indicating the number of rows updated

```
<SOAP-ENV:Envelope ...>
<SOAP-ENV:Header/>
     <SOAP-ENV:Body>
            <axl:executeSQLUpdateResponse ...>
                   <return>
                          <rowsUpdated>1</rowsUpdated>
                   </return>
            </axl:executeSQLUpdateResponse>
     </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Scripting / Automation

Automation

- AXL only provides simple configuration API
- Algorithms bring „intelligence“ to automation
- Automation requires API (AXL) plus algorithms
- Any platform be used to define the algorithms: C++, C#, Java, Phython, Perl, ...

Perl

- Practical Extracting and Report Language
- Pathologically Eclectic Rubbish Lister ☺
- scripting language
- published 1987
- GPL
- Perl is only used as an **EXAMPLE**; you can use ANY other language
- „Perl is the only language that looks the same before and after RSA encyption“ ☺

Perl for Windows

- ActivePerl is a Perl port for Windows
  http://www.activestate.com/products/activeperl/
- for AXL we need SSL support (HTTPS!)
- SSL-Support for ActivePerl:

  for SSL support an additional package has to be installed using perl package manager (ppm).

  Execute this on the CLI (DOS prompt):
  ppm install http://theoryx5.uwinnipeg.ca/ppms/Crypt-SSLeay.ppd

Summary

Next steps

- CCS-2007, “Adding addtl. value to UCM 6 based on it´s open standards approach”
- scripts using the AXL can be used to
  - set/change COS settings
  - provision users and phones
  - provision dial plans
  - check dial plan consistency
  - provision services
- adding a web frontend enables self service portals
- web services frameworks (e.g. AXIS) allow for automatic creation of java classes to access the API
- ...
- examples from this session available at http://www.employees.org/~jkrohn/BRKUCT-2014.zip

Summary

AXL API is a (complex) powerful API to extend Cisco Unified Communications Manager

Use of AXL optimizes day-2-day operations by automating repeated tasks

Operations costs can be reduced significantly

References

References

- Programming Perl, 3rd Edition
  by Tom Christiansen, Jon Orwant, Larry Wall
  Publisher: O'Reilly
  Pub Date: July 2000
  ISBN: 0-596-00027-8

- Learning SQL
  by Alan Beaulieu
  Publisher: O‘Reilly
  Pub Date: August 2005
  ISBN: 0-596-00727-2

- Web Services Essentials
  by Ethan Ceramo
  Publisher: O‘Reilly
  Pub Date: February 2002
  ISBN: 0-596-00224-6