cisco Live!

ALL IN

#CiscoLive

The bridge to possible

Successful Migrations from Unified CM to Webex Calling

Johannes Krohn, Principal Technical Marketing Engineer

BRKCOL-2481a

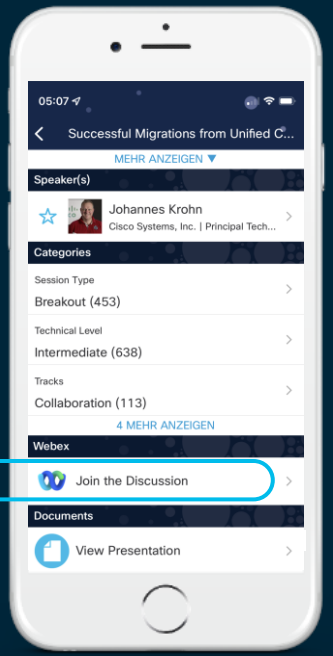
Cisco Webex App

Questions?

Use Cisco Webex App to chat
with the speaker after the session

How

1. Find this session in the Cisco Live Mobile App
2. Click “Join the Discussion”
3. Install the Webex App or go directly to the Webex space
4. Enter messages/questions in the Webex space



https://ciscolive.ciscoevents.com/ciscolivebot/#BRKCOL-2481a

Webex spaces will be moderated by the speaker until June 17, 2022.

Agenda

- General Process
- Discover
- Design
- Deploy
- Migrate

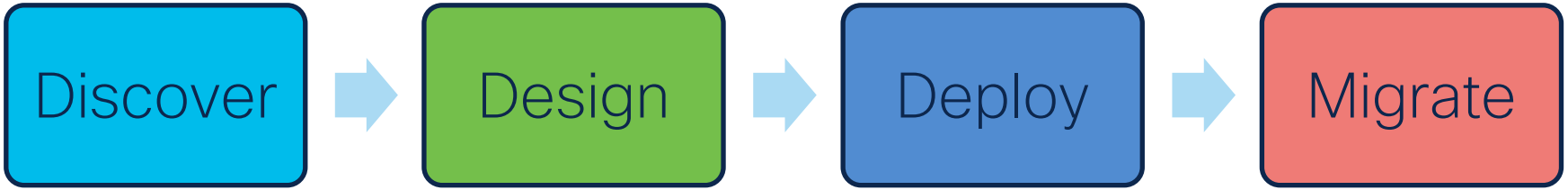
BRKCOL-2481a

BRKCOL-2481b

... with special focus on programmability using Python

Webex Calling Migration

General Process



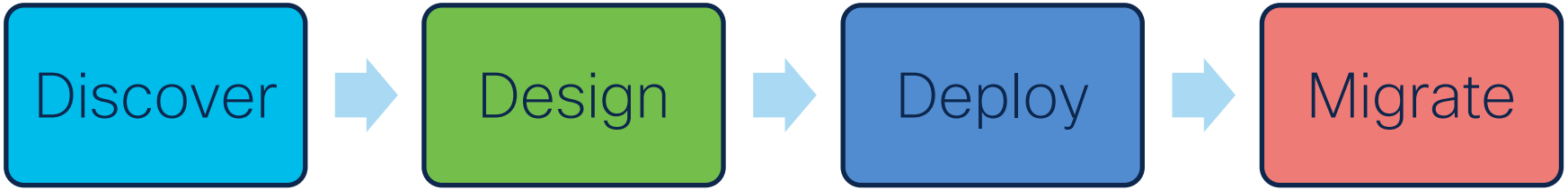
- Requirements
- Config assessment
- Inventory
  - users, devices, locations, …
- Feature utilization
- Integrations
- Validate network requirements

- Network requirements
- Feature mapping
- Migration batches
- Integrations
- Dial plan

- Infrastructure setup
- Base configuration
- Interworking setup
- Licensing

- Users
- Devices
- Features
- PSTN porting
- Acceptance test

General Process



- Requirements
- Config assessment
- Inventory
  - users, devices, locations, …
- Feature utilization
- Integrations
- Validate network requirements

- Network requirements
- Feature mapping
- Migration batches
- Integrations
- Dial plan

- Infrastructure setup
- Base configuration
- Interworking setup
- Licensing

- Users
- Devices
- Features
- PSTN porting
- Acceptance test

Discover

Analytics

- Gather insights of existing installation
- Cloud Connected UC
  - Call volume
  - Registered endpoints
  - (CAC) locations
  - Trunk utilization
- RTMT
- ...

Unified CM Data Extraction Options

- AXL – Administrative XML
  - SOAP based provisioning API
- BAT – Bulk Administration Tool
  - CSV based
- Config Export
  - Single file Unified CM config export

AXL

- The **Administrative XML Web Service (AXL)** is an XML/SOAP based interface that provides a mechanism for inserting, retrieving, updating and removing data from the Unified Communication configuration database.
- https://developer.cisco.com/site/axl/
- Thick AXL – API defines specific objects that can be created, removed, queried, or updated
- Thin AXL – Provides a mechanism to perform direct SQL queries / updates

AXL Challenge: Interface, Object Deserialization

- SOAP defines interface signature (endpoint, parameters, return) in WSDL (Web Service Definition Language) files
- Idea: automatic interface and API layer creation based on WSDL
- Reality
  - Trying to avoid interface creation
  - Manual SOAP message templates
  - Tools like SoapUI simplify this.

```
<operation name="addPhone">
    <soap:operation soapAction="CUCM:DB ver=11.5 addPhone" style="document"/>
    <input>
        <soap:body use="literal"/>
    </input>
    <output>
        <soap:body use="literal"/>
    </output>
    <fault name="fault">
        <soap:fault name="fault" use="literal"/>
    </fault>
</operation>

<operation name="addPhone">
    <input message="s0:addPhoneIn"/>
    <output message="s0:addPhoneOut"/>
    <fault name="fault" message="s0:AXLError"/>
</operation>
```

Solution: Zeep – Python SOAP Client

- “A fast and modern Python SOAP client”
- Python module to easily consume SOAP APIs
- “Zeep” (Dutch) = SOAP
- Consumes AXL WSDL and creates the Python interfaces

```python
# Create the Zeep service binding to AXL at the specified CUCM
service = client.create_service( '{http://www.cisco.com/AXLAPIService/}AXLAPIBinding',
                                 f'https://{os.getenv("CUCM_ADDRESS")}:8443/axl/' )

# Create a simple phone
# Of note, this appears to be the minimum set of elements required
# by the schema/Zeep
phone = {
        'name': 'CSFTESTPHONE',
        'product': 'Cisco Unified Client Services Framework',
        'model': 'Cisco Unified Client Services Framework',
        'class': 'Phone',
        'protocol': 'SIP',
        'protocolSide': 'User',
        'devicePoolName': 'Default',
        'commonPhoneConfigName': 'Standard Common Phone Profile',
        'locationName': 'Hub_None',
        'useTrustedRelayPoint': 'Default',
        'builtInBridgeStatus': 'Default',
        'packetCaptureMode': 'None',
        'certificateOperation': 'No Pending Operation',
        'deviceMobilityMode': 'Default'
}

# Execute the addPhone request
try:
    resp = service.addPhone( phone )
except Exception as err:
    print( f'\nZeep error: addPhone: { err }' )
    sys.exit( 1 )
```

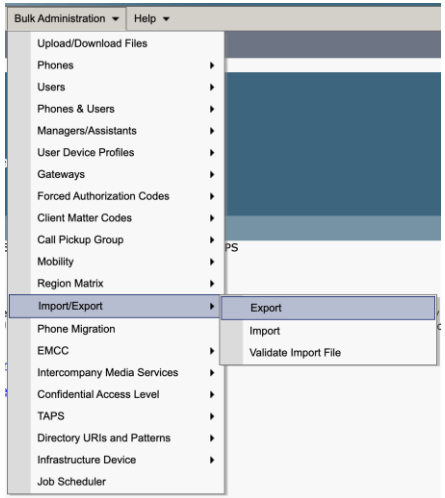
Examples: https://github.com/CiscoDevNet/axl-python-zeep-samples

Bulk Administration Tool (BAT)

- Main focus: simplify Unified CM provisioning
- .. but for migrations we are actually looking for the reverse
- Limited export capabilities: Users, Devices, User Device Profiles

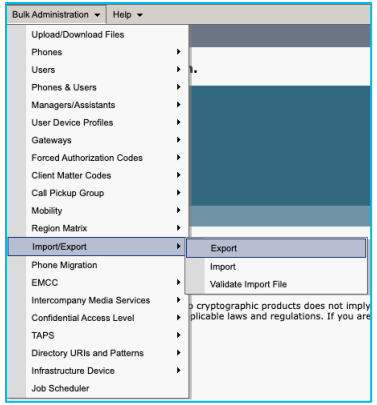
Config Export

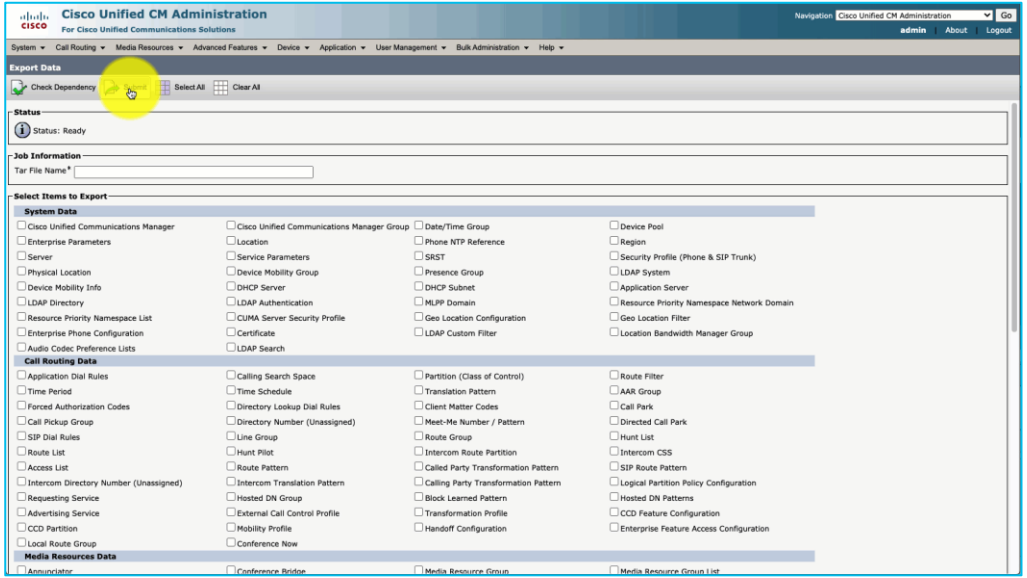
- Specific BAT option
- Allows full or partial Unified CM config export
- Result: TAR file with 172 files
  - One CSV for each config object type
- Examples
  - callpark.csv
  - callpickupgroup.csv
  - directedcallpark.csv
  - enduser.csv
  - huntlist.csv
  - huntgroup.csv
  - …

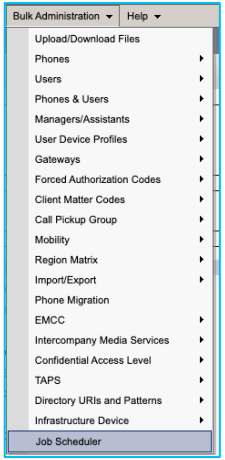


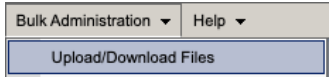
Unified CM Data Export

- Initiate Unified CM config export
- Wait for job to complete
- Download files









Working with Config Export Tar and CSV Files

- Challenge: large column count in some CSVs
  - Flat DB schema
  - phone.csv has the same 122 columns for each line on the phone
  - … and if there is a single phone with 11 lines then this alone leads to 1342 columns
- Manual analysis can/will be painful
- Automation
  - Excel: import, hide columns, filter operations, …
  - programmatic

Working with CSV Files using Python

- Csv: Python standard module
  - DictReader: parse CSV file and read into Python dictionaries
- But then, how to process the data?
  - List of dicts directly
  - Pandas: Python data analysis tool (think of it as programmatic Excel)
  - Parse CSVs into Python objects and “unflatten” CSVs (e.g. create line objects from data in phone.csv)

Example: Working with CSV Files using Python

- GitHub repository: https://github.com/jeokrohn/ucmmigration
- `ucmexport.Proxy` implements ”pythonic” way to access object within a tar file.
  - Not the most memory efficient ☺
  - .. but plays nicely with Python IDEs (auto completion, interactive debugger, …)
- Proxy also has logic to determine phone ownership, map DN/partition to user, …

User Migration Batches

Migration Batches

Which users have to be migrated together?

- Dependencies between Users
  - Monitoring each other on BLFs
  - Used in the same hunt pilot
  - Shared lines
  - Call pick-up
  - Using the same call-park numbers
  - Intercom
  - Shared DN on phones owned by different users
  - …
- Need to make sure that users w/ dependencies are migrated together
- This information is available in the Unified CM config export
- .. But somewhat hard to extract

Example: User Relations based on Hunt Pilots

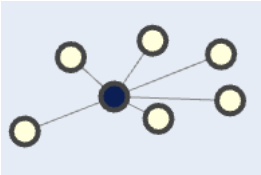
- Start with `huntpilot.csv`
- Look for ”HUNT LIST 1” column; this is a reference to “NAME” in `huntlist.csv` table
- In `huntlist.csv` looks at “LINE GROUP x” columns; reference to “NAME” in linegroup.csv
- Collect DNPs from “DN OR PATTERN x” and “ROUTE PARTITION x” columns in `linegroup.csv`
- Find phones with these DNPs in `phones.csv`; look at “Directory Number X” and “Route Partition X” columns
- Look at “Owner User ID” and “User ID x” columns to collect user IDs
- → Definitely needs to be done programmatically

Example: User Relations based on Hunt Pilots

- Start with `huntpilot.csv`
- Look for ”HUNT LIST 1” column; this is a reference to “NAME” in `huntlist.csv` table
- In `huntlist.csv` looks at “LINE GROUP x” columns; reference to “NAME” in linegroup.csv
- Collect DNPs from “DN OR PATTERN x” and “ROUTE PARTITION x” columns in `linegroup.csv`
- F and “P
- L
- → Definitely needs to be done programmatically

Alternative:
- Parse DN and partition from “PRIMARY EXTENSION” in `enduser.csv`
- And match these with DNPs collected from `linegroup.csv`
- .. If primary extension is populated for all users

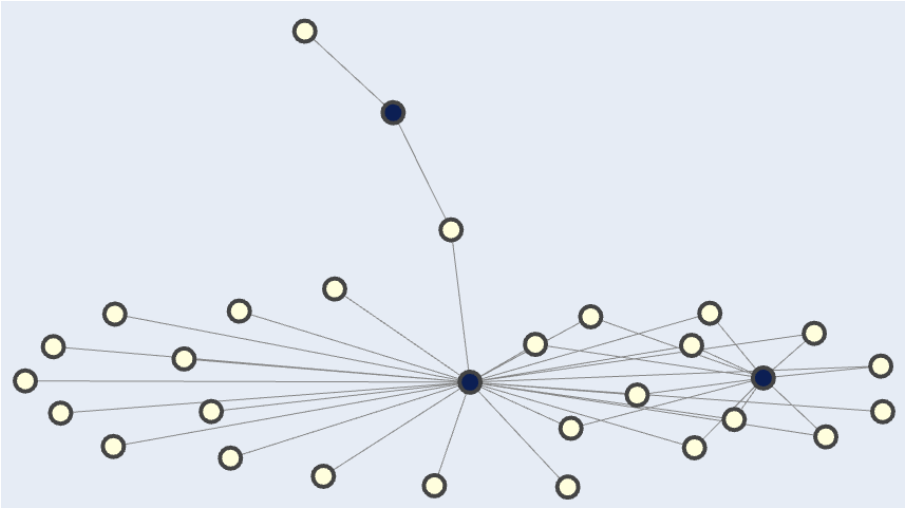
Real life examples: Hunt Pilots



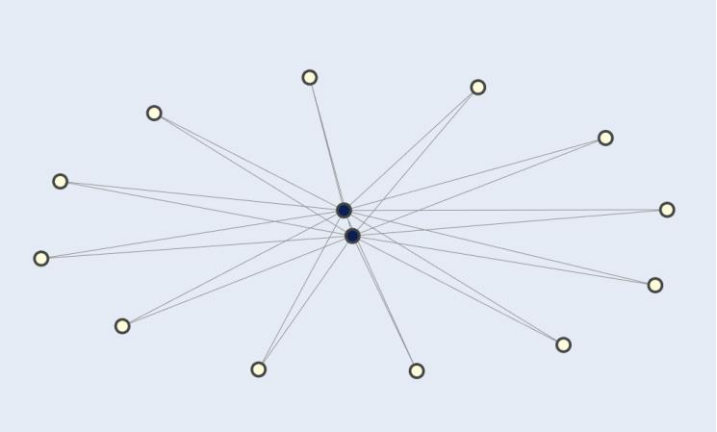
Simple

User

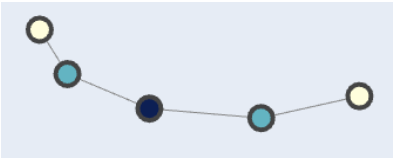
Hunt Pilot



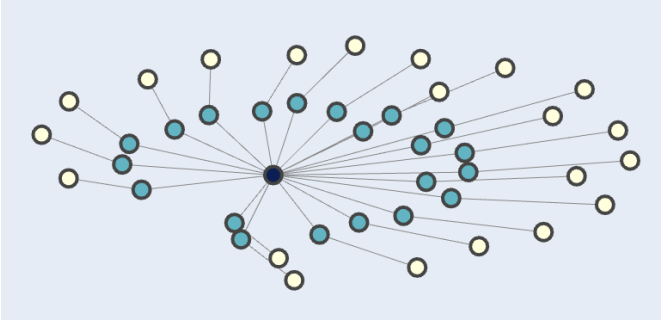
Complex



Real life examples: Shared Lines



DN shared on two phones owned by two users

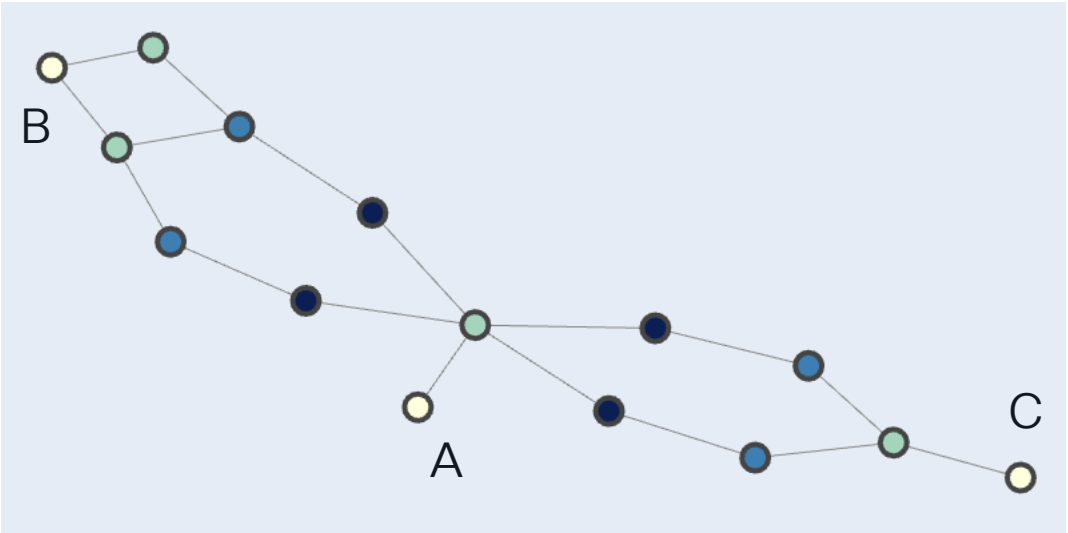


Single DN shared on multiple phones

- User
- DN
- Phone

Both cases are problematic for a migration to WxC

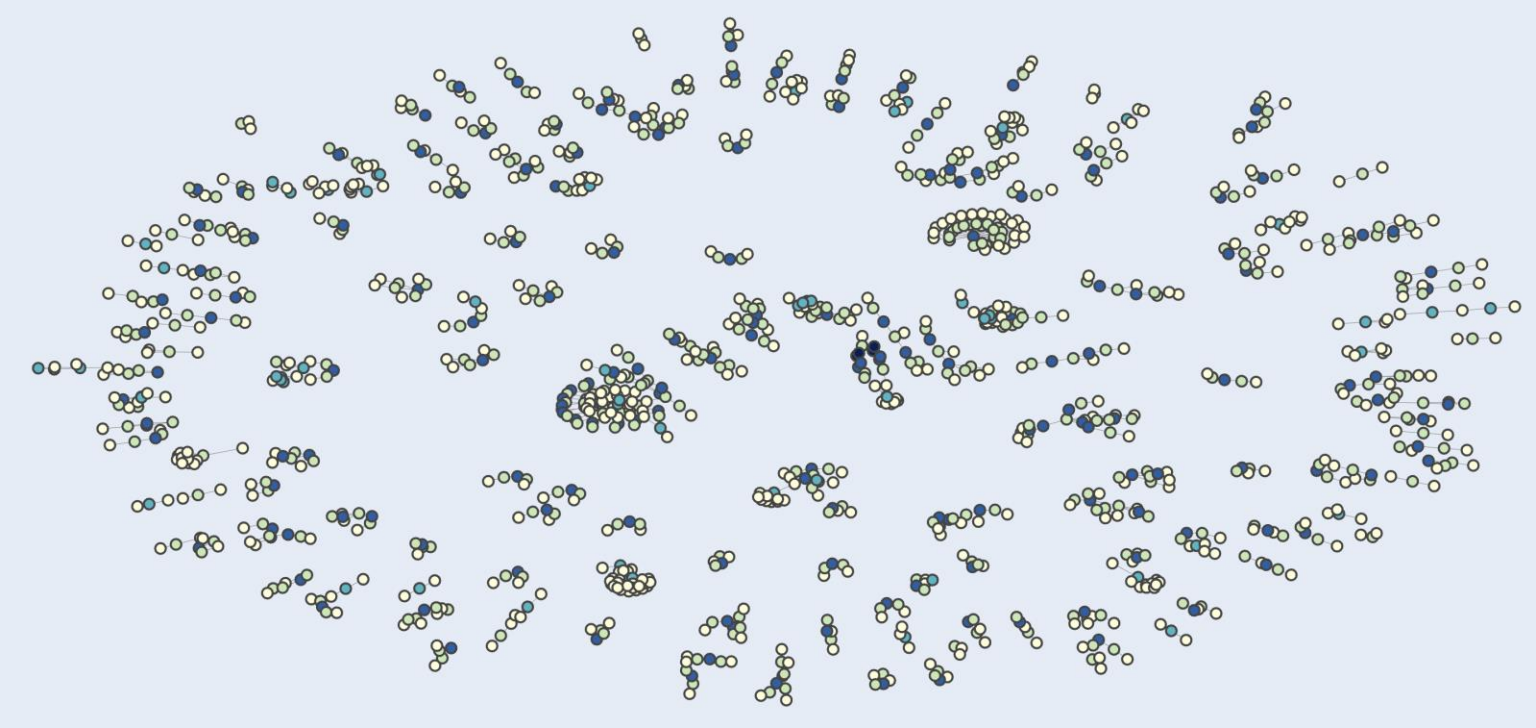
Real life examples: BLF



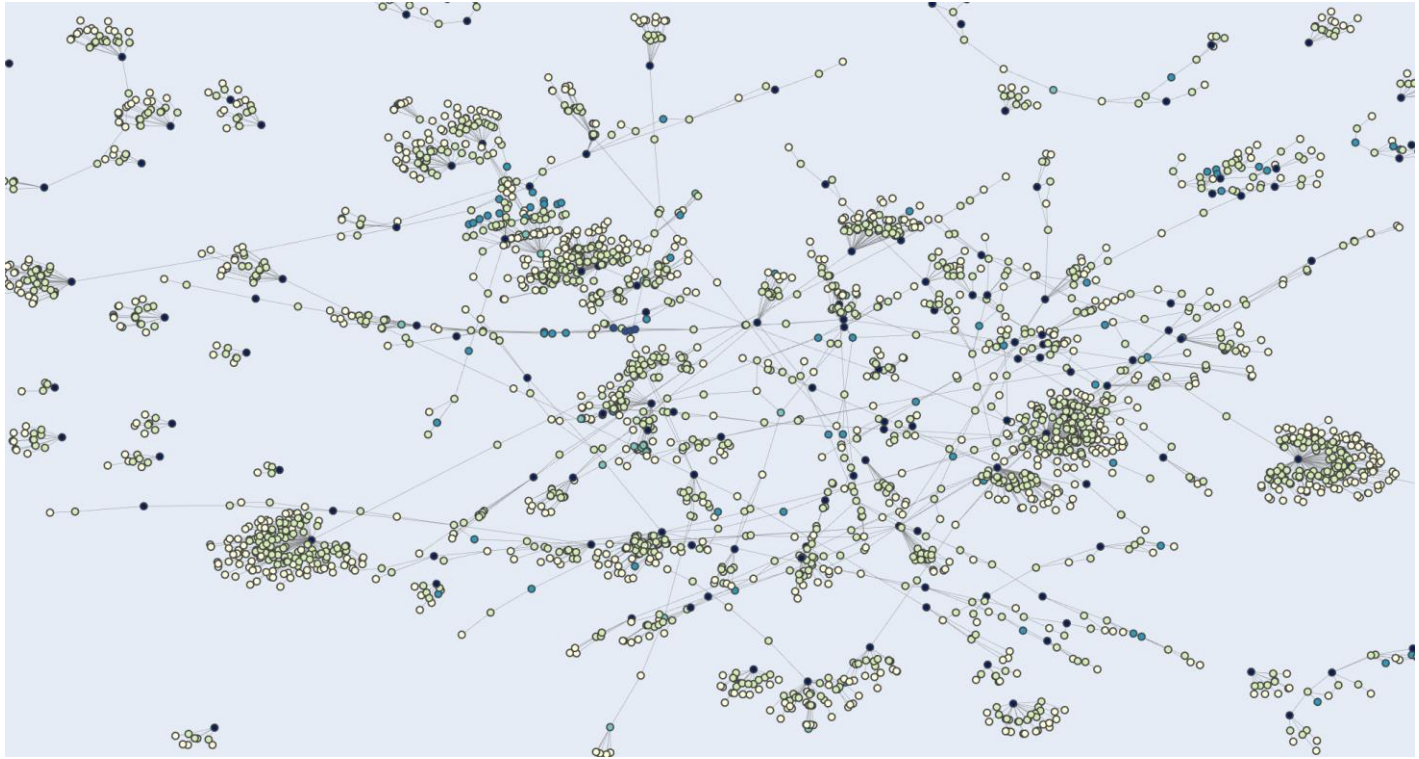
User
Phone
DN
BLF

User A on one phone has four BLFs monitoring DNs on phones of two users B and C

Real life example: combined

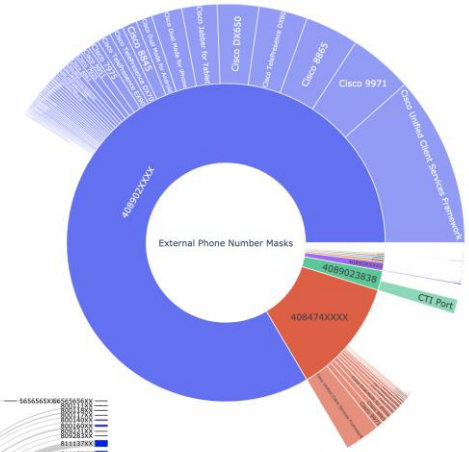


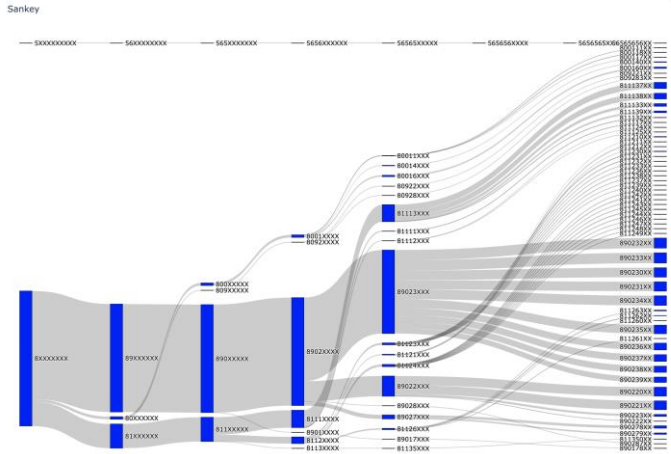
.. Or something … “interesting”



Migration Batches … and other analysis

- https://github.com/jeokrohn/ucmmigration
- Uses Python 3.9
- Read and analyse Unified CM data exports
- Work in progress…
- Use as is





Demo

Design

Challenge: DN vs User Centric Model

- Webex Calling
  - users with extension or TN (or both) have devices
  - Concept of shared lines fundamentally different to Unified CM
  - Users ”live” in a calling location
- What about Unified CM?

Challenge: DNP vs User Centric Model

- Unified CM
  - DNPs (DN, partition) associated with devices, potentially owned by users, which might have a primary extension, ..
  - DNs can exist on an arbitrary number of devices in varying order
  - What is a user's extension?
  - What is a location?
    - Common extension range
    - common +E.164 prefix (how long, what about extension only)
    - same device pool
    - shared CoS (how to you find users sharing equivalent CoS?)
    - ...

Challenge: DN vs User Centric Model

- Control Hub based migration tool has heuristics to try to address this challenge
- Admin can still override the tool
- Using the tool during the design phase (w/o actually executing a migration!) can assist in identifying characteristics of Unified CM setup

Feature Mapping

- Calling is not about features, but about business requirements
  - Different call control solutions have different sets of features
  - 1:1 feature mapping not necessarily the best option
- Different ways to address the same set of business requirements

Example: Hunt Group

- Unified CM
  - Hunt Pilot: DNP, Alerting Name
    - Call treatment: FwdNoAnswer, FwdHuntBusy, Queuing (MoH, overflow, max wait time, no agent available)
  - Hunt List: list of hunt groups
  - Line Group: distribution algorithm, RNA, hunt options (no answer, busy, unavailable), members (DNP)

- Webex Calling
  - Basics: location, name, number/extension, caller ID
  - Routing: circular, longest idle, weighted, simultaneous
  - Routing settings: advance when busy, forward after set number of rings, divert when unreachable
  - Agents

- DNP vs user based
  - Unified CM has DNPs in line groups
  - Webex Calling: user or workspaces as agents
- Gap: HG login/logout -> Webex Calling agents can set DND though
- OTOH: Webex Calling has way more options for selective call forwarding
- Gap: queueing .. But then there are Webex Calling call queues
  - Also allow agents to set their state to available/unavailable (for all queues though)

Dialing Habits

Unified CM (typical, best practice)

- Extensions (2-6 digits)
- Abbreviated inter-site (ESN)
- +E.164
- Country specific (PSTN) dialing habits
- .. and potentially countless variations based Unified CM “magic toolbox” → all bets are off

Webex Calling

- Extensions (2-6 digits)
- Abbreviated inter-site (ESN)
- +E.164
- Country specific (PSTN) dialing habits

- Unified CM dial plan flexibility: curse and blessing
  - Often used as workaround to address specific requirements
- Set of dialing habits in Webex Calling is fixed
- Case by case conversation
- Changing dialing habits → changing UX

Class of Service

Unified CM

- Based on CSSes, partitions and patterns
- Common: on-net, national, international
- Potentially: internal calling restrictions, Chinese walls, C-level fences
- Unified CM “magic toolbox” at its best→ all bets are off

Webex Calling

- outgoing permissions (internal, local, LD international…) - based on tags in national dial plan
- Executive/Executive Assistant
- Per user: selectively accept/reject/forward calls

- Different concepts
- Keep in mind: using Unified CM “dial plan magic” to address certain requirements often has been seen as a “workaround”
- If all you have is a hammer (CSS) then everything looks like a nail (set of patterns)
- Different toolbox → different solutions

Shared Lines

Unified CM

- Phone can have multiple DNPs
- Not necessarily tied to user
- DNPs can be in different sites
- Shared line appearances ring at the same time

Webex Calling

- User's lines can be shared (35 devices max)
- Limited to a single location
- For inbound calling: hunt groups and call queues might be the better option
- Single number reach allows to ring multiple destinations
- Executive assistant feature might address some use cases (exec and assistant can be in different locations)

- Different concepts
- Same location limitation seems to be the biggest challenge*

*Roadmap item

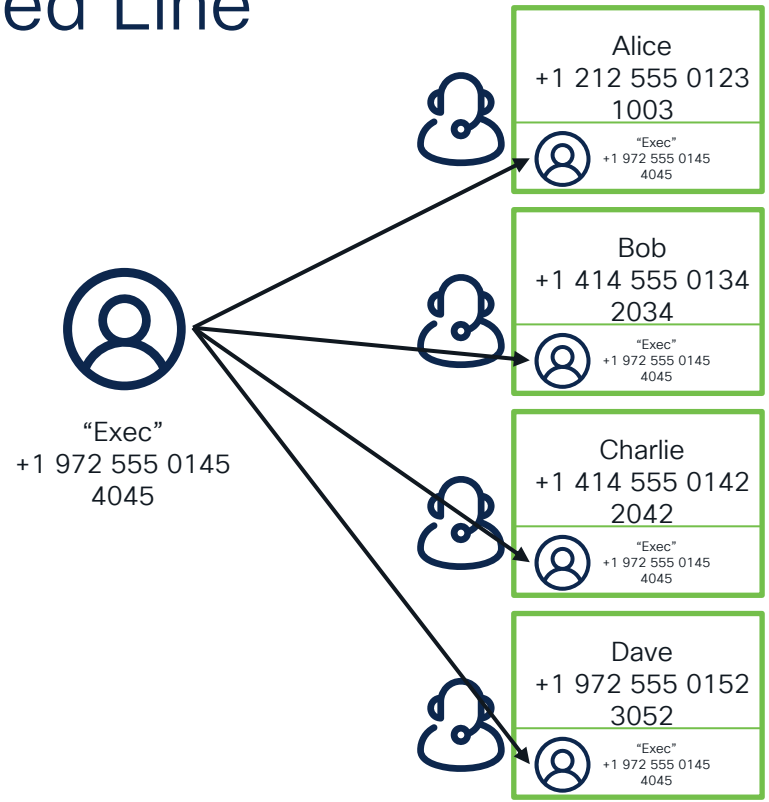
”Executive” as Flexible Shared Line

- Create ”dummy” exec

“Exec”
+1 972 555 0145
4045

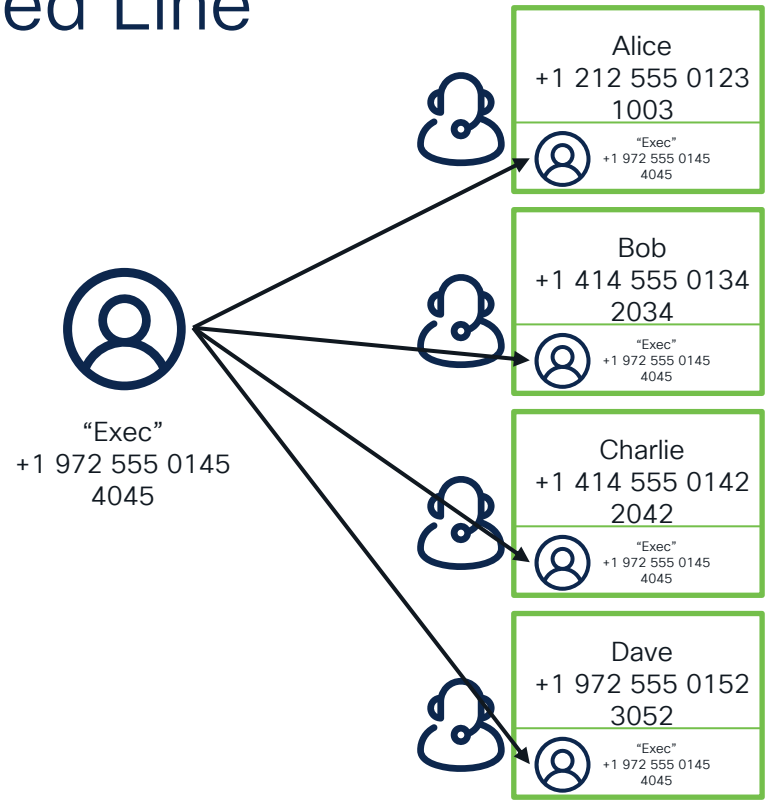
”Executive” as Flexible Shared Line

- Create ”dummy” exec
- .. and assign a bunch of assistants



”Executive” as Flexible Shared Line

- Create ”dummy” exec
- .. and assign a bunch of assistants
- Assistants can place calls on behalf of Executive
- Assistants get notification for incoming calls to Executive
  → can answer calls
- Exec and assistants don’t need to be in same location



Closing

Summary

- Covered in this session
  - Migration Process
  - Discovery
  - Design
- Foundation for deployment and migration
  → covered in BRKCOL-2481b

Key Takeaways

- Programmatic approach to extract/analyze data
- User batched based on dependencies between users
- Unified CM and Webex Calling are different ☺
- Focus on business requirements instead of 1:1 feature mapping

References

- Analyze Unified CM config exports: https://github.com/jeokrohn/ucmmigration
- API supported migration from Unified CM to Webex Calling and GDPR export: https://github.com/jeokrohn/migrationapi
- Python SDK for Webex Calling provisioning: https://pypi.org/project/wxc-sdk/

Technical Session Surveys

- Attendees who fill out a minimum of four session surveys and the overall event survey will get Cisco Live branded socks!
- Attendees will also earn 100 points in the Cisco Live Game for every survey completed.
- These points help you get on the leaderboard and increase your chances of winning daily and grand prizes.

Cisco Learning and Certifications

From technology training and team development to Cisco certifications and learning plans, let us help you empower your business and career. www.cisco.com/go/certs

Pay for Learning with Cisco Learning Credits

(CLCs) are prepaid training vouchers redeemed directly with Cisco.

Learn

Cisco U.
IT learning hub that guides teams and learners toward their goals

Cisco Digital Learning
Subscription-based product, technology, and certification training

Cisco Modeling Labs
Network simulation platform for design, testing, and troubleshooting

Cisco Learning Network
Resource community portal for certifications and learning

Train

Cisco Training Bootcamps
Intensive team & individual automation and technology training programs

Cisco Learning Partner Program
Authorized training partners supporting Cisco technology and career certifications

Cisco Instructor-led and Virtual Instructor-led training
Accelerated curriculum of product, technology, and certification courses

Certify

Cisco Certifications and Specialist Certifications
Award-winning certification program empowers students and IT Professionals to advance their technical careers

Cisco Guided Study Groups
180-day certification prep program with learning and support

Cisco Continuing Education Program
Recertification training options for Cisco certified individuals

Here at the event? Visit us at **The Learning and Certifications lounge at the World of Solutions**

Continue your education

- Visit the Cisco Showcase for related demos
- Book your one-on-one Meet the Engineer meeting
- Attend the interactive education with DevNet, Capture the Flag, and Walk-in Labs
- Visit the On-Demand Library for more sessions at www.CiscoLive.com/on-demand

Thank you

cisco Live!

ALL IN

#CiscoLive