



VOSS



**VOSS Automate
Best Practices Guide**

Release 24.2-PB2

March 19, 2025

Legal Information

- Copyright © 2025 VisionOSS Limited.
All rights reserved.
- This information is confidential. If received in error, it must be returned to VisionOSS ("VOSS"). Copyright in all documents originated by VOSS rests in VOSS. No portion may be reproduced by any process without prior written permission. VOSS does not guarantee that this document is technically correct or complete. VOSS accepts no liability for any loss (however caused) sustained as a result of any error or omission in the document.

DOCUMENT ID: 20250319064026

Contents

- 1 Deployment 1**
 - 1.1 VOSS Automate deployment topologies 1

- 2 Deployment Models and Web Weight Settings 18**
 - 2.1 Overview 18
 - 2.2 Active-Active Web Weights 18
 - 2.3 Active-StandBy Web Weights 19

- 3 Overload Controls 20**
 - 3.1 Session Limits 20
 - 3.2 Throttle Limits 20
 - 3.3 Configurable Number of Queue Processes 21

- 4 Onboarding Customers and Users 23**
 - 4.1 Guidance on Planning for Onboarding and Ongoing Operations 23

- 5 Data Sync 24**
 - 5.1 General Sync Principles and Best Practices 24
 - 5.2 Cisco Unified CM 29
 - 5.3 Cisco Unity Connection 33
 - 5.4 LDAP 34
 - 5.5 Cisco Webex App (Spark) 34
 - 5.6 MSGraph and MS-Teams Sync 35

- 6 Data Collection 45**
 - 6.1 Recommended RIS API Data Collector Interval 45

- 7 API Performance 46**
 - 7.1 API Resource Listing Best Practice 46
 - 7.2 Long Running API Requests 47

- 8 System Maintenance 49**
 - 8.1 Transaction Archiving 49
 - 8.2 Automated Database Cache Cleanup 49

- 9 Administration Portal Setup 51**
 - 9.1 Navigation - Menu and Dashboards 51

- Index 55**

1. Deployment

1.1. VOSS Automate deployment topologies

1.1.1. Overview

VOSS Automate offers two main deployment topologies:

- *Unified Node Cluster topology*
- *Modular Node Cluster deployment topology*

In addition, the following deployment options are available:

- Cloud deployments
- VOSS Automate Cloudv1

1.1.2. Node types

The Automate deployment topologies are comprised of a configuration of the following types of nodes:

- Web proxy node
- Unified/single node
- Application node
- Database node

Each node type performs specific functions within the topologies, and comprises one or more of the following components (software subsystems):

Component	Description
Operating system	Ubuntu, stripped down / hardened
Platform	Docker, isolated components
Web server	nginx, receives and forwards HTTP requests <ul style="list-style-type: none"> • Hosts static files: CSS, JS and images • Load balance between unified nodes (UNs): round robin, configurable, e.g. two data centres • Detects inactive UN: removes from round robin
Database	MongoDB (scalable, distributed), PostgreSQL (scalable)
Application	JavaScript, Python, REST API, device drivers, workflow engine, transactions/queue engine, RBAC, search, bulk loader, and more ...

The matrix outlined in the table describes the set of components in each node type:

Component	Node types			
	Web proxy	Uni-fied/single node	Applica-tion	Database
Operating system	X	X	X	X
Platform	X	X	X	X
Web server	X	X		
Database		X		X
Application		X	X	

1.1.3. Unified Node Cluster topology

Automate's **Unified Node Cluster** topology provides the following options:

- Single-node cluster (cluster-of-one/standalone)
- Single-node cluster (cluster-of-one/standalone) with VMWare HA
- 2 Node with Web proxies
- 4 Node with Web proxies
- 6 Node with Web proxies

Important: Choose between a Unified Node deployment or a Modular Architecture deployment.

In a **Unified Node Cluster** deployment, VOSS Automate is deployed either as a **Single Unified Node Cluster**, 2 unified nodes, or a cluster of multiple nodes with High Availability (HA) and Disaster Recovery (DR) qualities.

Each node can be assigned one or more of the following functional roles:

- WebProxy - load balances incoming HTTP requests across unified nodes.
- Single Unified Node - combines the Application and Database roles for use in a non-multi-clustered test environment.
- Unified - similar to the **Single Unified Node** role Application and Database roles, but clustered with other nodes to provide HA and DR capabilities

The nginx web server is installed on the WebProxy, **Single Unified Node**, and **Unified Node Cluster**, but is configured differently for each role.

In a clustered environment containing multiple **Unified Node Clusters**, a load balancing function is required to offer HA (High Availability providing failover between redundant roles).

VOSS Automate supports deployment of either the WebProxy node or a DNS load balancer. Here are some considerations in choosing a WebProxy node vs. DNS:

- The Proxy takes load off the **Unified Node Cluster** to deliver static content (HTML/JAVA scripts). When using DNS or a third-party load balancer, the **Unified Node Cluster** has to process this information.
- DNS does not know the state of the **Unified Node Cluster**.
- The WebProxy detects if a **Unified Node Cluster** is down or corrupt. In this case, the WebProxy will select the next **Unified Node Cluster** in a round robin scheme.

We recommend that you run no more than two **Unified Node Clusters** and one WebProxy node on a physical server (VMware server). Also recommended is that the disk subsystems be unique for each **Unified Node Cluster**.

The following deployment topologies are defined:

- Test: a standalone, **Single Unified Node** with Application and Database roles combined. No High Availability/Disaster Recovery (HA/DR) is available.

Important: This deployment should be used for test purposes only.

- Production with Unified Nodes: in a clustered system, comprising:
 - 2, 3, 4 or 6 Unified nodes (each with combined Application and Database roles)
 - 0 to 4 (maximum 2 if 2 Unified nodes) WebProxy nodes offering load balancing. The WebProxy nodes can be omitted if an external load balancer is available.

Single-node cluster (cluster-of-one/standalone)

A single-node cluster (cluster-of-one/standalone) deployment should be used for test purposes only.



The table describes the advantages and disadvantages of a single-node cluster (cluster-of-one/standalone) deployment topology:

Advantages	Disadvantages
<ul style="list-style-type: none"> • Smallest hardware footprint 	<ul style="list-style-type: none"> • No high availability or disaster recovery • Less throughput than clusters

Single-node cluster (cluster-of-one/standalone) with VMWare HA

The table describes the advantages and disadvantages of a single-node cluster (cluster-of-one/standalone) with VMWare HA deployment topology:

Advantages	Disadvantages
<ul style="list-style-type: none"> • Smallest hardware footprint • Disaster recovery available 	<ul style="list-style-type: none"> • Less throughput than clusters

Multinode cluster with unified nodes

To achieve Geo-Redundancy using the Unified nodes, consider the following:

- Either four or six Unified nodes - each node combining Application and Database roles - are clustered and split over two geographically disparate locations.
- Two Web Proxy nodes to provide High Availability that ensure an Application role failure is gracefully handled. More may be added if Web Proxy nodes are required in a DMZ.

It is strongly recommended *not* to allow customer end-users the same level of administrator access as the restricted groups of provider- and customer administrators. This is why Self-service web proxies as well as Administrator web proxies should be used.

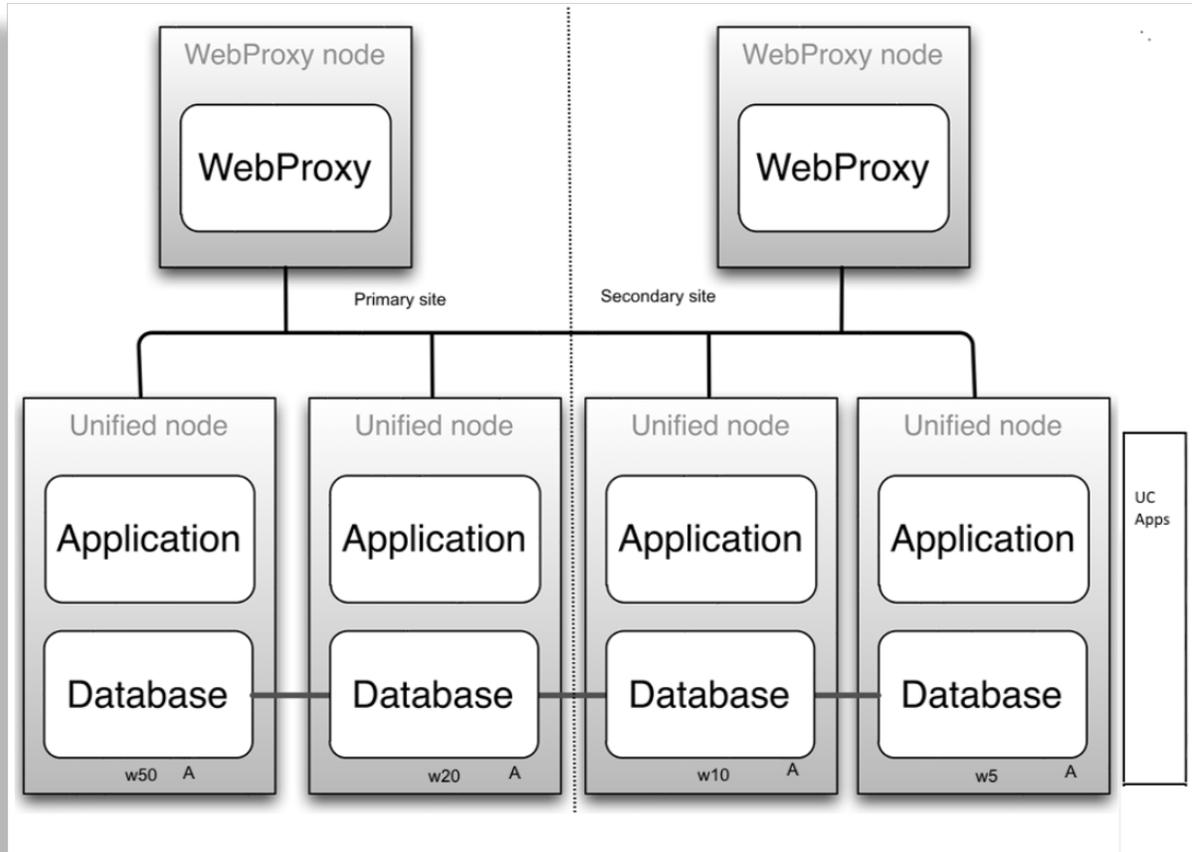
Systems with Self-service only web proxies are *only* recommended where the system is customer facing, but where the customer does not administer the system themselves.

- Web Proxy and Unified nodes can be contained in separate firewalled networks.
- Database synchronization takes places between all Database roles, thereby offering Disaster Recovery and High Availability.
- For 6 unified nodes, all nodes in the cluster are active. For an 8 node cluster (with latency between data centers greater than 10ms) , the 2 nodes in the DR node are passive, in other words, the **voSS workers 0** command has been run on the DR nodes.

Primary and fall-back Secondary Database servers can be configured manually. Refer to the Platform Guide for further details.

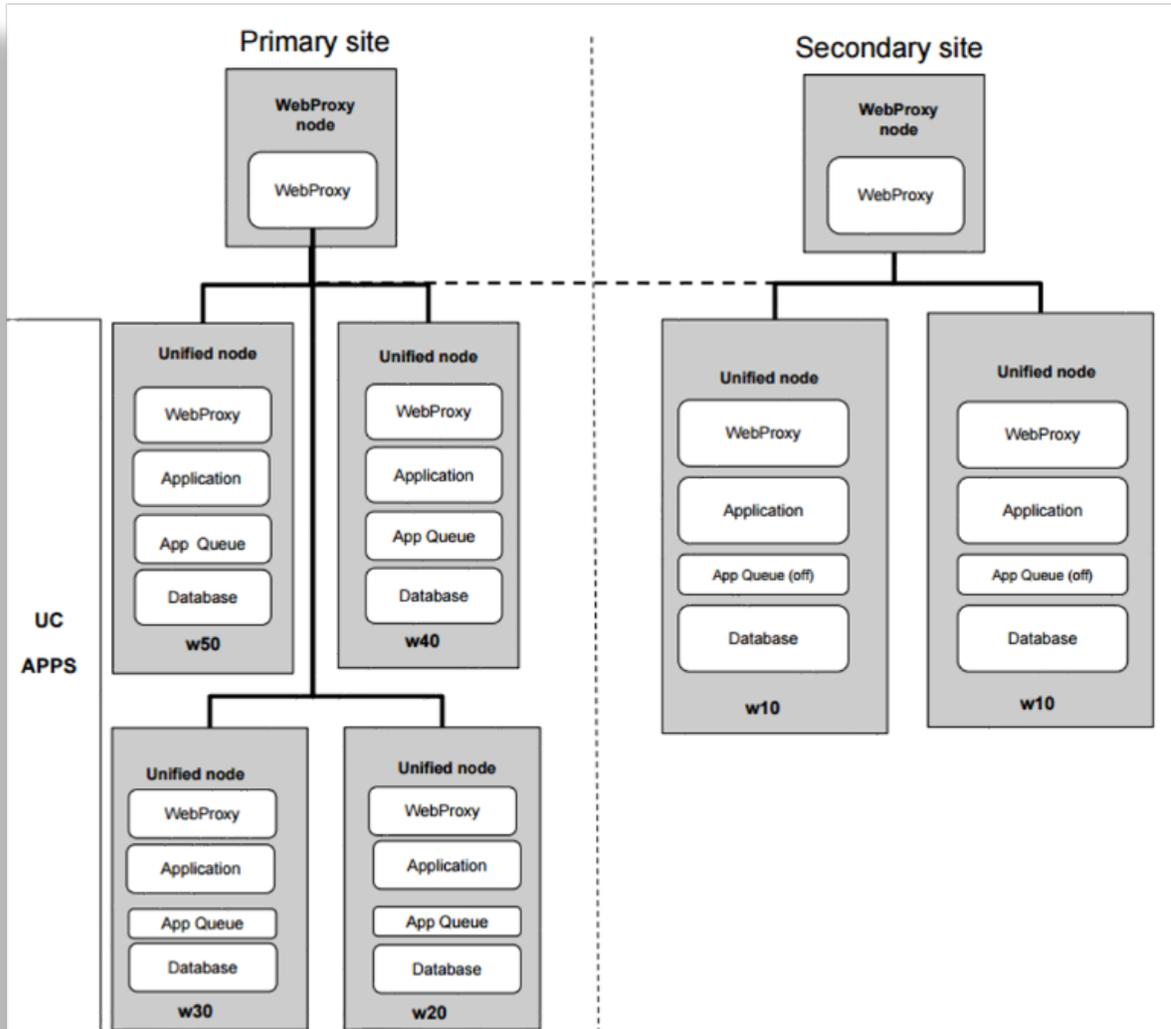
Example: 6-Node Cluster

The diagram illustrates an example of a 6-Node Cluster:



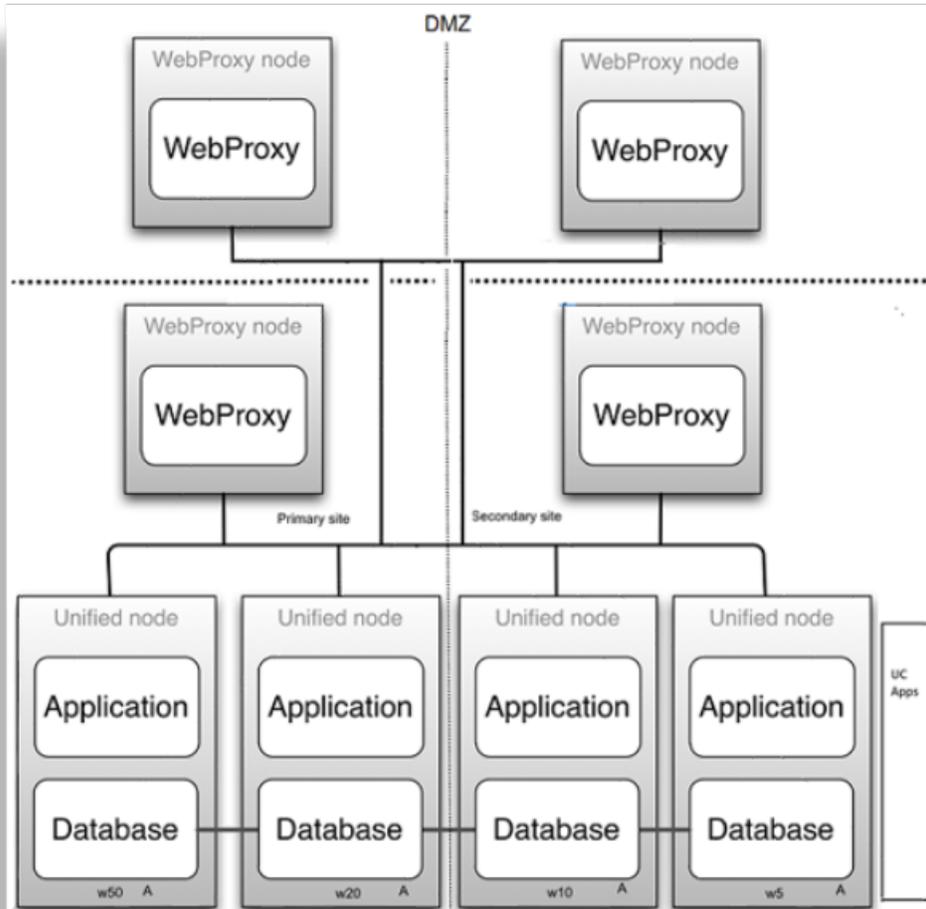
Example: 8 Node Cluster

The diagram illustrates an example of an 8-Node Cluster:



Example: 2 Web Proxy Nodes in a DMZ

The diagram illustrates an example of 2 Web proxy nodes in a DMZ:



Example: 4 Web Proxy Nodes in a DMZ (2 admin, 2 Self-service)

The diagram illustrates an example of 4 Web proxy nodes (2 admin, and 2 Self-service) in a DMZ:

- If the secondary unified node has *more than 10ms latency* with the primary unified node, it must be configured to be in the *same* geographical location.

Important:

With only two Unified nodes, with or without Web proxies, there is no High Availability. The database on the primary node is read/write, while the database on the secondary is read only.

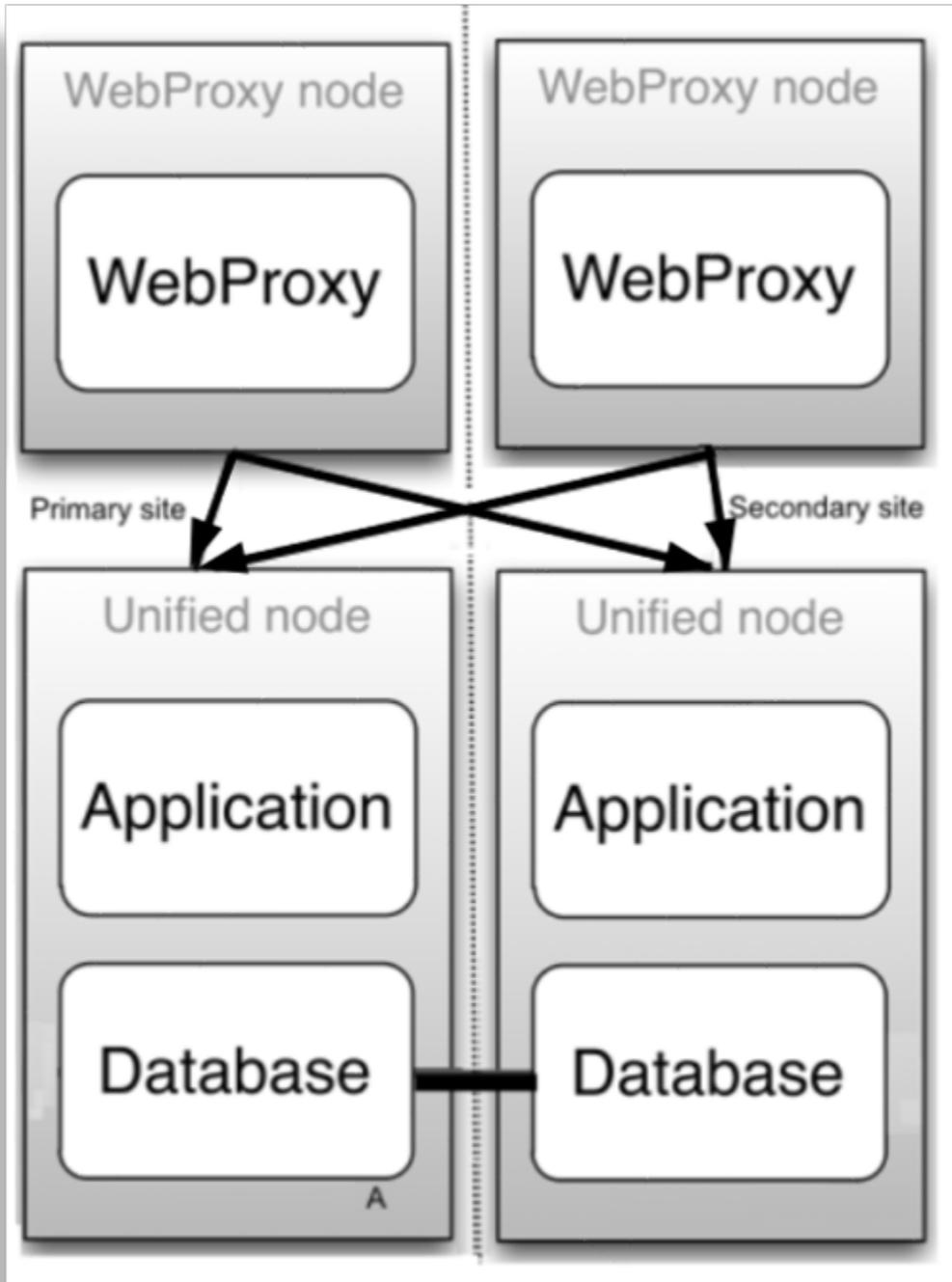
Only redundancy is available.

- If the primary node fails, a manual delete of the primary node on the secondary and a cluster provision will be needed.
- If the secondary node fails, it needs to be replaced.

Refer to the topic on DR Failover and Recovery in a 2 Node Cluster in the Platform Guide.

Example: 2 Node Cluster

The diagram illustrates a 2 Node Cluster:



4 Node with web proxies

The table describes the advantages and disadvantages of a 4 Node with Web Proxies deployment topology:

Advantages	Disadvantages
<ul style="list-style-type: none"> • More disaster recovery scenarios supported • More throughput than 3 Node 	<ul style="list-style-type: none"> • More hardware than 3 Node

6 Node with web proxies

A 6 Node with Web proxies deployment topology:

- Is typically deployed for multi-data center deployments
- Supports Active/Standby

1.1.4. Modular Node Cluster deployment topology

Overview

A **Modular Node Cluster** topology has separate Application and Database nodes:

- 3 Database nodes
- 1 - 8 Application Nodes
- Web Proxies

A **Modular Node Cluster** topology has the following advantages:

- Increased processing capacity
- Horizontal scaling by adding more Application nodes
- Improved database resilience with dedicated nodes and isolation from application
- Improved database performance by removing application load from the primary database

Important: Choose between a **Unified Node Cluster** deployment or a **Modular Node Cluster** deployment.

VOSS Automate is deployed as a **Modular Node Cluster** of multiple nodes with High Availability (HA) and Disaster Recovery (DR) qualities.

Each node can be assigned one or more of the following functional roles:

- WebProxy - load balances incoming HTTP requests across nodes.
- Application role node, clustered with other nodes to provide HA and DR capabilities
- Database role node, clustered with other nodes to provide HA and DR capabilities

The nginx web server is installed on the WebProxy and application role node, but is configured differently for each role.

Related Topics

Modular Architecture Multinode Installation in the Install Guide.

Migrate a **Unified Node Cluster** to a **Modular Node Cluster** in the Platform Guide.

A load balancing function is required to offer HA (High Availability providing failover between redundant roles).

VOSS Automate supports deployment of either the WebProxy node or a DNS load balancer. Consider the following when choosing a WebProxy node vs. DNS:

- The Proxy takes load off the application role node to deliver static content (HTML/JAVA scripts). When using DNS or a third-party load balancer, the application role node has to process this information.
- DNS does not know the state of the application role node.
- The WebProxy detects if an application role node is down or corrupt. In this case, the WebProxy will select the next application role node in a round robin scheme.

We recommend that you run no more than one application role node and one database role node and one WebProxy node on a physical server (VMware server). When selecting disk infrastructure, high volume data access by database role replica sets must be considered where different disk subsystems may be required depending on the performance of the disk infrastructure.

The following **Modular Node Cluster** topology is recommended (minimum):

Important: Single Unified Node topologies are not available for **Modular Node Cluster** deployments.

- Production with nodes: in a clustered system of 2 data centers:
 - DC1 = primary data center containing primary database node (highest database weight)
 - DC2 = data recovery data center

The system comprises of the following nodes:

- 3 nodes with application roles (2 in DC1; 1 in DC2)
- 3 nodes with database roles (2 in DC1; 1 in DC2)
- Maximum 2 WebProxy nodes if 2 data centers; offering load balancing. The WebProxy nodes can be omitted if an external load balancer is available.

Multinode Modular Node Cluster with application and database nodes

To achieve Geo-Redundancy using Application and Database nodes, consider the following:

- Six Application and Database nodes - 3 nodes with an application role and 3 nodes with a database role - are clustered and split over two geographically disparate locations.
- Two Web Proxy nodes to provide High Availability that ensure an Application role failure is gracefully handled. More may be added if Web Proxy nodes are required in a DMZ.

It is strongly recommended *not* to allow customer end-users the same level of administrator access as the restricted groups of provider- and customer administrators. This is why Self-service web proxies as well as Administrator web proxies should be used.

Systems with Self-service only web proxies are *only* recommended where the system is customer facing, but where the customer does not administer the system themselves.

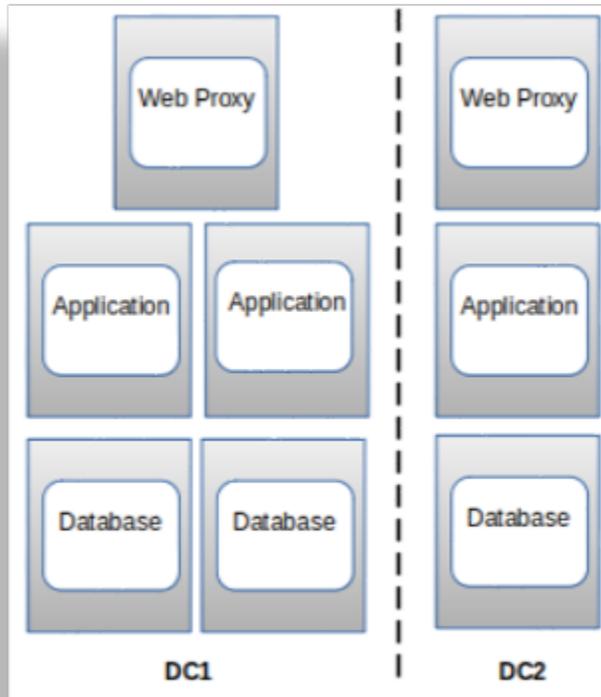
- Web Proxy, application and database nodes can be contained in separate firewalled networks.

- Database synchronization takes place between all database role nodes, thereby offering Disaster Recovery and High Availability.
- All nodes in the cluster are active.

Primary and fall-back Secondary Database servers can be configured manually. Refer to the Platform Guide for further details.

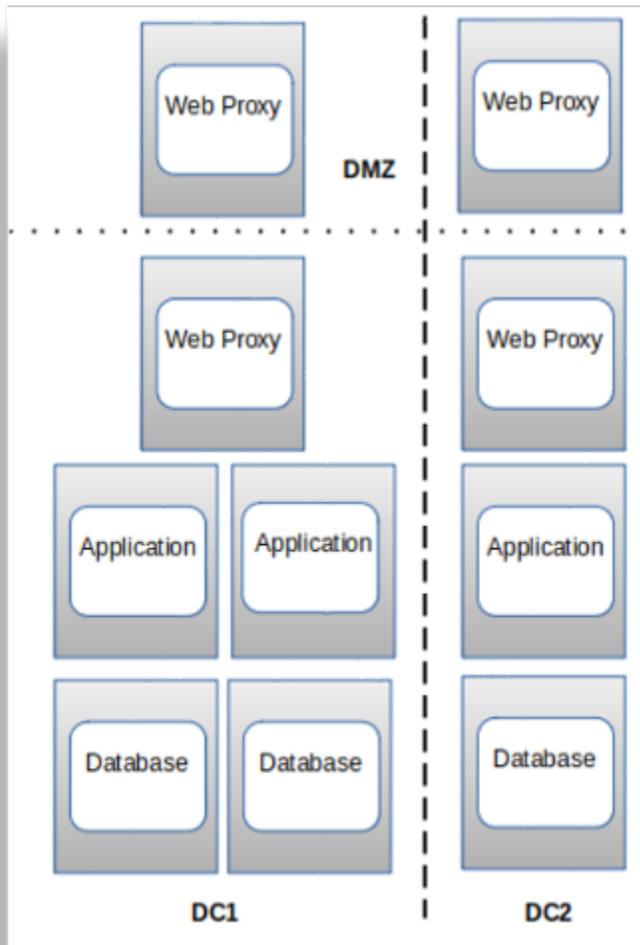
Example: 6 Node Cluster

The diagram illustrates an example of a 6 Node Cluster:

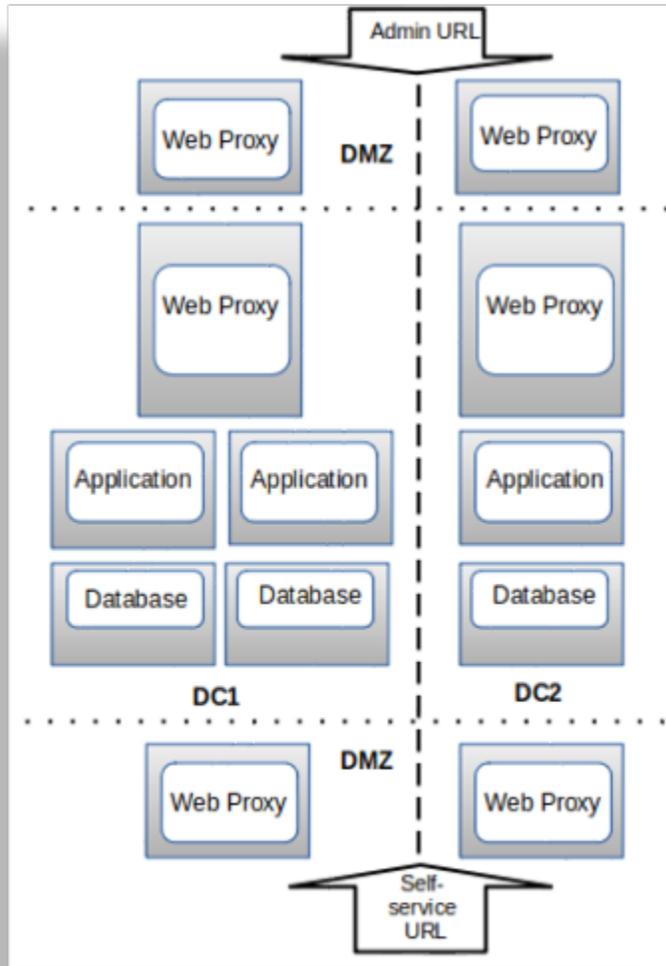


Example: 2 Web Proxy Nodes in a DMZ

The diagram illustrates an example of 2 Web Proxy Nodes in a DMZ:

**Example: 4 Web Proxy Nodes in a DMZ**

The diagram illustrates an example of 4 Web Proxy Nodes in a DMZ (2 admin, 2 Self-service):



1.1.5. Cloud deployments

VOSS Automate supports the following Cloud deployments:

- Microsoft Azure
- Amazon Web Services (AWS)

Although Google Cloud Platform (GCP) is not officially supported, contact us to discuss your requirements.

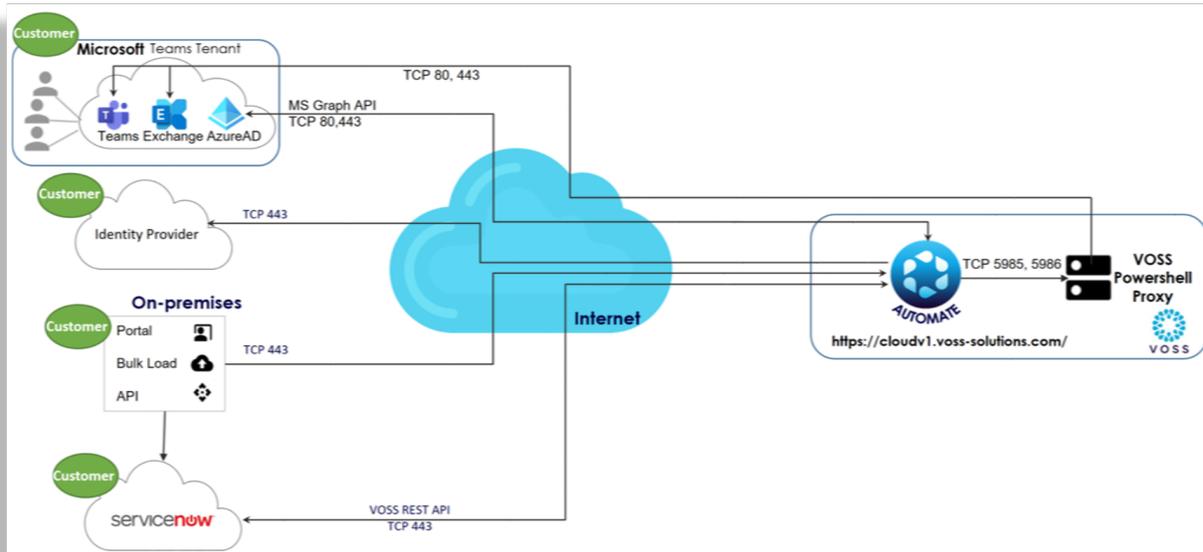
The advantages of a Cloud deployment topology:

- Leverage cloud tooling, such as proxies (which can be used instead of VOSS Web Proxy)

1.1.6. VOSS Automate Cloudv1 (SaaS)

VOSS Automate Cloudv1 is a Software-as-a-Service (SaaS) offering hosted on a shared VOSS Automate instance within Microsoft Azure.

VOSS manages this instance, which seamlessly integrates with a customer's Unified Communications (UC) platform, Microsoft Exchange, Microsoft Active Directory, and third-party applications like ServiceNow and Identity Providers (IdPs) for Single Sign-On (SSO) authentication.



2. Deployment Models and Web Weight Settings

2.1. Overview

- The supported deployment models are described in the Install Guide under Chapter 2 Deployment Topologies.
- Web weights are explained in the Install Guide under Multi Data Center Deployments and Multinode Installation. The web weight specifies the routing and relative counts of the initial HTTP request from the Web Proxy to a Unified Node. The initial request could be a request such as a transaction, or for example a GET request.
- Transactions can be processed on *any* Active Unified Node - regardless of which Unified Node processed the initial HTTP request. The transaction log provides the detailed information in the fields shown below:
 - `submitter_host_name`: the hostname of the application node that scheduled the transaction.
 - `processor_host_name`: the hostname of the application node that processed the transaction (this value is set once the transaction is processed).
- Note that a sub-transaction can be processed on a different Unified Node than its immediate precedent in the hierarchy.
- We recommend that you use both Web Proxies. However, the use of only 1 Web Proxy is supported (and Web Proxy use is optional).
- To display the configured web weights, run the command **web weight list** at the CLI of each Web Proxy Node .
- The recommended web weight settings for the various deployment models are shown in the following sections.

2.2. Active-Active Web Weights

- There are 4 Active Unified Nodes, 2 in each Data Center. The maximum supported Round Trip Time (RTT) is 10ms.
- WP1: 1 1 0 0
- WP2: 0 0 1 1

This scheme is designed to route the initial HTTP request to the Unified Nodes local to the Web Proxy Node that forwards the request. If only one Web Proxy (WP) is used, then use the following setting for WP1:

WP1: 1 1 1 1

This results in some of the initial HTTP requests crossing to the Secondary Data Center, however this has a minimal impact on system performance.

2.3. Active-StandBy Web Weights

- There are 4 Active Unified Nodes in the Primary Data Center and 2 StandBy Unified Nodes in the secondary Data Center. The maximum supported RTT is 400ms.
- WP1: 1 1 1 1 0 0
- WP2: 1 1 1 1 0 0

If only Web Proxy 1 (WP1) is used, the default weights provided by the system are sufficient. If Web Proxy 2 or both Web Proxy Nodes are used, change the web weights at Web Proxy 2 to the values noted above.

The logic behind the web weight settings for the Active-StandBy model is that some non-transaction work may generate a significant number of queries back to the Primary DB. Therefore, processing such work in the secondary Data Center may result in unacceptably long processing times.

3. Overload Controls

3.1. Session Limits

The numbers below represent the default and maximum values.

- global administration: 200 (includes non-customer admins as well as service provider and reseller admins). This limit also includes API clients configured as Admin Users.
- global self-service: 20,000 - This is the total number of self-service users logged into the system - both active and inactive.
- per customer administration: 10 (this should be set to a lower value in some cases). The Partner must first “reserve” a number of non-customer admin sessions from the global limit of 200 noted above. The remaining admins can be allocated to customers. Based on the expected number of customers, the Partner can then set the per customer admin limit.

For example, if the Partner wishes to “reserve” 20 admin sessions for non-customer use, that would leave 180 available for customer use. If a total of 40 customers is expected, the Partner should set the per customer admin limit to no more than $180/40 = 4$ (rounded down from 4.5). In this example, a maximum of $40 \times 4 = 160$ admin sessions can be allocated to customer-level admins. This would effectively reserve $200 - 160 = 40$ admins for the Partner to use.

- per customer self-service: 1000

3.2. Throttle Limits

- Admin (by default, this is disabled). We recommend that the Admin throttle is enabled and set to 450 API requests/min. The setting is per Unified Node.
 - Service Inventory (SI) Report: Relies on the per-user throttle setting to ensure adequate throughput.
 - * For the Active-StandBy deployment model, we highly recommended that the SI report is configured to run on a specific non-Primary Unified Node (preferably in the Secondary Data Center as those nodes are likely to have a lower load). This results in faster performance, but there is not any protection against a single node failure in the middle of an SI report run (not very likely). The use of a Web Proxy is *not* recommended here as 25% of the initial requests are routed to the Primary Unified Node based on the recommended web weight settings at either Web Proxy.
 - * For the Active-Active deployment model, you can use a Web Proxy instead. The SI report would run slower, but this configuration provides protection against a single Unified Node failure during the SI report run. If a Web Proxy is used, then:

- Use Web Proxy 2. This prevents routing to the Primary Unified Node based on the recommended web weight settings.
- The Web Proxy knows the health of the UN and can route requests accordingly.
- Per User throttle for API Clients:
 - Default setting is 20 API req/sec per Active Unified Node. 4 Active Unified Nodes x 20 = 80 API req/sec (system wide) or 4800 API req/min (system wide). We recommend that you keep this setting.
 - This limit applies to all admin users, but in practice serves to limit API clients. Human admin users are not likely to create a traffic rate of 80 API req/sec.
- Self-service throttle. The default setting is 300 API req/min (per Unified Node). APIs for logins and actions would count against this throttle.

3.3. Configurable Number of Queue Processes

Important:

- It is strongly advised that VOSS Support is consulted before making changes to the number of queue processes.
- The number of queues cannot be set to a value larger than the number of cores in the VM. A message `Validate: <num> is not a valid less_than_cores_number` will show in this case.

Available commands:

- **voss queues <number>** - Set the number of queue processes

This command restarts the voss-queue services.

voss queues - Get the number of queue processes

When using these commands, a CLI warning is shown to refer to this documentation:

```
Warning, updating this setting, without proper consideration of
Best Practices or consultation with VOSS support, can lead to system
instability.
Do you wish to continue?
```

The number of queue processes is configurable in order to increase transaction throughput and will improve workload distribution across the cluster, but can only be made after considering other configuration changes or performance areas. These include:

- The maximum number of queues cannot exceed the number of cores in the VM
- Node memory configuration
- Impact on API and indirectly GUI responsiveness
- Number of workers for queue processes on different unified nodes
- Overall load on the primary node (node with primary database responsible for all database updates)

3.3.1. Node Memory

When increasing queue processes, too little memory headroom can lead to out of memory errors on the unified node, which can cause services to be restarted and in rare situations, can also lead to database services being stopped.

The suggested required headroom per queue process should be considered as 4GB.

3.3.2. Impact on API and GUI responsiveness

A balance has to be created between the number of queue processes and API/GUI responsiveness. Increasing the number of queue processes on all nodes will increase the load on the primary node during high transaction load and the increased load on the database can lead to degraded API and GUI responsiveness if the number of queue processes are set too high.

3.3.3. Number of workers per queue process

Note: This consideration applies to the standard topology with unified nodes.

In order to alleviate load on the primary node, it is recommended to set the number of workers to zero. This will prevent any transactions from being processed on the primary node. This will allow the primary node to better service

- the higher query load from secondary nodes due to higher transaction load
- API requests requiring database interaction

A special consideration exists with setting the workers to zero on the designated primary node. When the primary node fails over to a secondary due to some event, the newly elected primary node will not have the number of queue workers set to zero, which could lead to an increased load on the newly elected primary that will process transactions, service API requests and service database queries.

Manual intervention will be required to set the number of workers to zero on the newly elected primary or restore the configured primary node to primary state.

It is recommended that monitoring be set up to automatically provide notifications in case of primary failover.

4. Onboarding Customers and Users

4.1. Guidance on Planning for Onboarding and Ongoing Operations

This is a high-level view:

- Number of Parallel operations, for example BL and QAS, for best performance:
 - BL: 4x500 (4 Bulk Load sheets in parallel with a maximum of 500 rows per sheet).
 - QAS: 5x200 (5 QAS Bulk Load sheets in parallel with a maximum of 200 rows per sheet).
- Recommendations for sync operations:
 - We recommend that you schedule sync operations during off-peak hours.
 - During business hours, sync operations are slower due to the presence of other work on the system.
- Scheduler Template settings (20%, 50%, 80%):
 - For periods of high self-service and administrative work (including API clients), we recommend that the template is set to 20%.
 - For periods of moderate self-service and administrative work (including API clients), we recommend that the template is set to 50%. This is the value in the system prior to SU-1.
 - For periods of low self-service and administrative work (including API clients), set the template to 80%.
 - The current implementation only allows you to set 2 of the 3 values, that is a peak and an off-peak setting.
- In some cases, there are situations where VOSS Automate is used for changes. Ad Hoc syncs are best in this case.
- AS may choose to change CUCM and sync back to VOSS Automate. This is where you must schedule daily off-peak sync operations.

5. Data Sync

5.1. General Sync Principles and Best Practices

5.1.1. Sync Overview

Overview

VOSS Automate provides several features for keeping the system in sync with underlying UC applications. This allows for the configuration and management of the UC apps outside of VOSS Automate, when required.

Sync feature	Description
Cache control policy	This VOSS Automate mechanism provides the ability to pull in the latest live data from the UC application(s) for the entity you're viewing, or at the time that it's needed - for example, before executing a change on that entity, to prevent any overwrite or setting conflict. Find out more about cache control policy behavior and configuration in the Data Sync chapter in the Core Feature Guide.
Data Sync	This workflow pulls the latest data from the UC apps and updates the VOSS Automate cache when ran adhoc, or via a schedule. This is typically used for processes such as overbuild, to pull in the existing configuration from the UC applications or to pull in other changes made in the UC apps outside of VOSS Automate. For more information on the sync behavior and configuration see the Data Sync chapter in the Core Feature Guide.

Note: With the cache control policy in place, the need to setup and schedule sync regular syncs should aim to address any gaps that the cache control policy won't handle.

When to use Data Syncs vs Cache Control Policy

This section describes some prime use cases and guidelines for determining when a regular or scheduled sync might be required for an entity, instead of using the cache control policy.

Note: These scenarios assume some level of regular configuration being done to the UC apps directly, outside of VOSS Automate.

Adding/removing entities in the UC applications directly

If you're adding or removing entities, such as users or phones, in the UC applications directly, then a sync is required to pull in new entities or to remove existing entities.

Modifying key values that appear in the list views in Automate via the UC applications directly

If you're modifying key values that appear in the list views in VOSS Automate via the applications directly, such as changing a user's name, then an update sync may be required.

List view data is driven only from the VOSS Automate cache; thus, any updates made in the UC applications will not display in VOSS Automate until the entity is viewed in VOSS Automate (for example, opening that subscriber), or when an update sync is run.

Extracts run from VOSS Automate

Any type of extract that might be run from VOSS Automate, such as file dump, VOSS Automate Analytics, or billing feed, are based on cached data in VOSS Automate. Thus, a sync may be required if those capabilities are in use and any of the critical settings in those extracts are being managed outside of VOSS Automate.

External clients accessing VOSS Automate via the API

External clients accessing VOSS Automate via the API have the `cached` flag available to request VOSS Automate cached data (`cached = true`), or to have VOSS Automate retrieve the latest data from the UC applications before responding (`cached = false`). Thus, the presence of this external client does not require a regular sync to be run as it can (and likely should) request the latest data in any case depending on the use case.

Any other mods made on the entity

Any other mods made on an entity, such as call forward on a line via the CFwdALL softkey, will be pulled in when the record is viewed, and so won't necessarily require a sync. For example, if the only concern is that when executing an update to an entity, that the latest current settings are shown, then the cache control policy handles this without the need for a regularly scheduled update sync.

Consider the Sync Hierarchy

When running a sync, manually or via schedule, the hierarchy (customer or site) at which you're running the sync determines where the items are pulled into. For example:

- A sync at the customer level pulls data in at customer level
- A sync at a site pulls data in at the site

For this reason, when setting up the sync, consider its purpose. If the items being pulled in need to be in a site, it may be more efficient to set the sync up at the site level, and run the sync at the site rather than syncing in at the customer level, and then having to move the various elements. This can even be done as a once-off sync, and using model type lists and model instance filters to grab the data relevant for the site.

5.1.2. Data Sync Types

VOSS Automate provides the following data sync types:

Data sync type	Description
Pull from Device	Available to all device types. <ul style="list-style-type: none"> • Pull all data from the device • Pull only the schema from the device (used for LDAP) • Pull data from the Change Notification Feature local data collection
Purge Local Resources	Available to all device types. <ul style="list-style-type: none"> • Purge data from the cache
Push to Device	Available only to Cisco Unified CM devices <ul style="list-style-type: none"> • Push data in the cache to the device
Change Notification Sync	Available only to Cisco Unified CM devices

Note: A quick import option is available to fetch only summary data that is contained in a list operation response and not the data for all instances/fields. See Data Sync Overview in the Core Guide for details.

Generally, for all sync types, VOSS Automate builds up the lists of entities from both VOSS Automate and the device, and compares them, using the key for the device entity. The key is typically the unique identifier (ID) for the record in the device we're syncing with. For example, for Unified CM, the ID is the *pkid*, which is the internal Unified CM database ID.

For subscribers, a sync builds up the list of `device/cucm/Users` in VOSS Automate and then requests from the Unified CM the lists of users it currently has for the comparison. Differences in the lists are handled according to each sync type.

Related Topics

- Data Sync Overview in the Core Feature Guide
- Change Notification Feature Overview in the Core Feature Guide

Pull from Device

For sync type *Pull from Device*, the VOSS Automate resource is updated where the same key is present in both lists. In this case, the device data is the master and the VOSS Automate system model data is updated with the device data.

For example, let's say new data is added to the Unified CM, so that the VOSS Automate system data state for a Unified CM `device/cucm/User` does not show instances that are shown on the Unified CM.

In this case, a *pull* data sync synchronizes the system data with the Unified CM data. For example, a user's Department may be updated on the Unified CM, but the update only shows on the system after a *Pull from Device* sync. If a user resource is created in Unified CM but not in VOSS Automate, this adds the `device/cucm/User` instance into VOSS Automate at the level the *pull* sync was run from, for example, at the customer level.

When deleting a VOSS Automate resource from the device, so that the key is in the VOSS Automate list but not in the device list, a *pull* sync removes the resource in VOSS Automate. For example, if the resource is a user in VOSS Automate but not in Unified CM, the *pull* sync removes the `device/cucm/User` record in VOSS Automate.

To restrict the number of records removed in VOSS Automate, ensure you have the following named macro at the hierarchy where the sync takes place:

```
PULL_SYNC_DELETE_THRESHOLD_<device_type>
```

For details, see Pull Sync Delete Threshold topic in the Advanced Configuration Guide.

When pulling device data, for example LDAP users from an LDAP device, the results returned to VOSS Automate depend on the LDAP server configuration. For example, if the returned results exceed the LDAP server configured maximum, and if the server does not support paging, an appropriate error message is returned.

For Microsoft 365 syncs, a **Max page size** (default 1000) setting can be adjusted if the error "Template output exceeded memory limit" is shown.

For details, see the Configure Microsoft Tenant Connection Parameters topic in the Core Guide.

Push to Device

Sync type *Push to Device* is available only to Cisco Unified CM device types.

In a *Push to Device* sync type, devices are synchronized with the VOSS Automate system data state, which is the primary data state.

- When deleting device data from VOSS Automate so that the key is in the *device* list but not in the VOSS Automate list (for example, delete user in VOSS Automate), the user is removed from Unified CM. The user will not exist on the device or on VOSS Automate.
- When adding new device data to VOSS Automate so that the resource shows instances that are not shown on the device, a *push* data sync synchronizes the device data with the VOSS Automate data. For example, adding a `device/cucm/User` instance to VOSS Automate and running a *Push to Device* sync adds the user record to Unified CM.

Keys found in both lists are ignored. Existing records are not updated in either direction.

In the `device/cucm/User` example, if the same user exists on both VOSS Automate and on Unified CM, no update occurs in either direction. Detailed settings may still not match after a *Push to Device* sync.

Important: When performing a *push* sync, it is important to consider data dependencies between different models.

For example, data dependencies may exist between users and phones in the Cisco Unified CM. In this case, if a user is associated to a phone (via the associated devices on the user), you can't add the user if the phone does not yet exist in Cisco Unified CM.

On the other hand, for ownerID on the phone, pushing the phone first will fail since the user isn't in place.

This might mean running the *push* sync multiple times so it loads in the required order, or you may need to modify data (such as removing device association) to allow the *push* sync to succeed.

Note: The keys list sync logic described in this topic implies that in case of a reversion of the Unified CM to restores/inactive partitions, the end-state of the relevant pkids may differ to their state the last time VOSS Automate was in sync with Unified CM (before a restore), particularly if testing occurred in between. This means you may, for example, have a user with the same username in both VOSS Automate and Unified CM, but if that user's pkid in Unified CM now differs to the one in VOSS Automate from previous syncs or interactions, they will be seen as different users even though they have the same usernames.

Change Notification Sync

Sync type *Change Notification Sync* is available only to Cisco Unified CM device types.

A *Change Notification Sync* is a pull sync of changes stored in the local collection that is updated by the Change Notification Collector service.

For more details on Change Notification Sync, see the related topics in Data Sync section of the Core Feature Guide.

Purge Local Resources

In a *Purge Local Resources* sync type, all resources or instances of device information that exists in the system are deleted. Entities in the device are not deleted.

Note: The default *purge* syncs created when adding a CUCM, CUC, LDAP or CCX server type are disabled by default. To use the *purge* sync, the "Remove" check box must first be cleared on the "Disabled Operations" tab of the relevant sync.

This sync type is typically used when cleaning up the system. The system displays a warning before executing an enabled *purge* sync.

See the following sample device type syncs:

- `HcsPurge-{{CUCMHostname}}-{{CUCMClusterName}}-DS`
- `HcsUserPurgeDS-{{CUCMHostname}}-{{CUCMClusterName}}`
- `HcsPhonePurgeDS-{{CUCMHostname}}-{{CUCMClusterName}}`

- HcsPurge-{{CUCXHostname}}-{{CUCXClusterName}}-DS
- PurgeUccx-{{UCCXHostName}}
- HcsLdapUserPurge-{{UniqueID}}
- PurgeSpark{{CustomerName}}

5.1.3. Scheduling Syncs

- When scheduling syncs, avoid too many overlapping syncs at a given time. VOSS Automate already blocks multiple syncs against a given device.
- The best practice is to not have more than 5 syncs running at a given time.
- To avoid load and issues with concurrency, schedule syncs carefully and at intervals when they are really required. For example, do not run nightly syncs unless it is necessary. Since syncs generally cover the case where information is changed on the UC apps outside VOSS Automate, the level of third party integration or direct configuration tasks should play a role in the decision to schedule. For details, refer to the topics on Cisco Unified CM, CUC and LDAP below.
- Since it is possible to limit the number of records processed with a given sync, more predictability can be obtained with scheduling.

5.2. Cisco Unified CM

5.2.1. Cisco Unified CM Sync

Cisco Unified CM supports two types of sync:

- Regular API sync - utilizing the regular use of LIST and GET API calls to retrieve data; like any other device sync.
- Change Notification Sync - utilizes a service on the Cisco Unified CM side to pull information about records that have changed in a given period. Note: Model Instance Filters cannot be used in conjunction with a Change Notification Sync.

The Change Notification Sync type is generally the most efficient sync type to use, as it minimizes the amount of data that needs to be retrieved from the Cisco Unified CM (especially for updates).

The Change Notification Sync process works as follows:

VOSS Automate retrieves the change records from the Cisco Unified CM on a regular interval (configurable). For example, this could be every 10 minutes. At the time of a scheduled sync is setup, VOSS Automate processes the change records collected (for example, nightly). VOSS Automate then processes the records accordingly:

- Add - will do a GET API call to retrieve the full record and add it to VOSS Automate.
- Update - will do a GET API call to retrieve the full record and update the record in VOSS Automate.
- Del - will remove the record from VOSS Automate.

So the efficiency on update syncs is because we do not need to do a GET API call for every single record in the system - only those that changed. In large UC application installations, this can make a big difference in Update sync times.

5.2.2. Update Sync Operations

Partners can use the Change Notification Feature of the CUCM to process update sync operations faster. The feature is OFF by default, but can be turned on by the partner.

The information here provides guidelines for setting up a sync schedule and lists the associated performance implications. Details on the operation of this feature are provided in other documents such as the Core Feature Guide. The changes mentioned here are not transactions. As a result, information is not displayed in the translation log, but rather in the special logs created for this feature.

The guidelines presented here are derived from concepts related to total processing capacity. The total number of updates processed in a time period is the sum of all of the updates across the customers selected for update in that time period. In our case, the time period is one hour. In this example, we assume that each customer has 1000 CUCM-related changes in that hour. The recommendation noted in the table that follows indicates that 5 customers can run in parallel (concurrently), and therefore a total of 5,000 changes processed in total.

If the partner exceeds the recommendation of 5 concurrent customers, a performance degradation may be observed, and the full set of required changes may not complete within that hour. Alternatively, if the number of changes for any customer is significantly higher than the 1,000 or if the total number of changes is significantly greater than 5,000, then the concurrency number supported may be less than 5.

If some of the planned changes do not complete within the hour noted in the table below, then those changes are completed the next time that particular customer is scheduled for a sync. If the number of changes for any customer is so large that the changes continually exceed those that can be processed in one hour, it will eventually result in a full sync. For such customers, we advise to schedule within an hour where less than 5 customers execute concurrently.

Configuration	Recommendation
Maximum number of concurrent CNF synd	5
Maximum number of changes processed per CNF sync	1,000
CNF sync schedule frequency	Once per hour per customer - This is subject to the staggering of CNF sync across customers.
Staggering of CNF syncs across customers	Factor of maximum changes processed and maximum number of concurrent CNF syncs.
CNF collector frequency	Initial recommendation is 15 minutes.
When is Full sync required?	Weekends only or when there are CNF alerts prompting for full sync.

If you experience a significant performance issue, you can turn the feature OFF again. Contact your support representative if you have any performance concerns.

5.2.3. Staggering of CNF Syncs Across Customers

Below follows an example and considerations:

If a Partner has 20 customers who want to use CNF sync, only schedule a maximum of 5 CNF sync to run concurrently. This means that syncs would run as follows:

- 1st hour, for example 12:00
Customer 1 to Customer 5
- 2nd hour, for example 13:00
Customer 6 to Customer 10
- 3rd hour, for example 14:00
Customer 11 to Customer 15
- 4th hour, for example 15:00
Customer 16 to Customer 20
- 5th hour, for example 16:00 (Begin repeating customers)
Customer 1 - Customer 5
- and so on.

The preceding example means that the CNF sync schedule per customer must run at 4 hour intervals. Therefore, there are 6 CNF syncs per customer within a 24 hour window. With each CNF sync processing up to 1k changes, there are:

- A total of 6k changes processed per customer in a 24 hour window
- A total of 120k changes processed across all 20 customers in a 24 hour window

5.2.4. Recommended CUCM Sync Setups

Cisco Unified CM (CUCM) sync recommendations are covered here.

Bottom Up User Sync

If using bottom up sync into CUCM, the users are added to CUCM via LDAP. In this scenario they do not appear in VOSS Automate in order to be managed until they are synced in.

Note: If you use this sync in a multi-cluster environment, additional guidance on the user sync setup is provided in the Multi-Cluster Deployments Technical Guide.

- Recommended setup
 - Model Type List - device/cucm/User
 - Actions - Add/Update/Del all enabled.
 - When to use - scheduled. The most frequent this should run is in line with the LDAP->CUCM sync time (typically once every 24hrs but minimum of every 6hrs or so). The length of this sync should consider the maximum allowable time for an end user to be in the system in typical business practices. Edge cases can always be handled in between scheduled syncs by running the sync

manually if required - that is often better than having a very frequent sync that is not typically needed.

- Events - the different actions (add/update/del) have different post execution events for the `device/cucm/User` model type that need to occur. These handle various aspects of the user setup. See below for a screenshot of the setup for an example:
 - * Add Operation workflow = `UserCucmSyncAdd`
 - * Update Operation workflow = `UserCucmSyncUpdate`
 - * Delete Operation workflow = `UserCucmSyncRemove`
- Change notification should be used for this sync to manage load (except if using a model instance filter).

Workflows fields of the event setup on the CUCM User sync:

- **Model Type:** `device/cucm/User`
Operation: Add
Phase: Post Execution
Workflow: `UserCucmSyncAdd`
- **Model Type:** `device/cucm/User`
Operation: Update
Phase: Post Execution
Workflow: `UserCucmSyncUpdate`
- **Model Type:** `device/cucm/User`
Operation: Delete
Phase: Post Execution
Workflow: `UserCucmSyncRemove`

Phone Types and Related Entities

This will force VOSS Automate to retrieve the latest phone type data from the CUCM and related entities like phone button templates, and so on. This is not possible via the change notification in CUCM today.

- Recommended setup:
 - Model type list including: `device/cucm/PhoneType`, `device/cucm/PhoneTemplate`, `device/cucm/securityProfiles`
 - Actions - Add/Update/Del all enabled
 - When to use - Not scheduled - run ad hoc as needed. This includes post CUCM upgrades, installation of a new device COP file in CUCM, managing phone button templates, managing device security profiles. If you are not seeing a phone type of the button template in VOSS Automate that you are expecting, running this sync will likely resolve it.

Other Syncs

Beyond the syncs above, others can be setup to suit specific needs based on the implementation.

Important: In setting up processes that sync any new entities into VOSS Automate, these will add the entities to the hierarchy level of the sync. So this will require the use of overbuild or ad hoc move processes to get the entities into the right site, for example, if needed (such as users, phones, lines, and so on).

5.3. Cisco Unity Connection

5.3.1. Cisco Unity Connection Sync

User related services such as Unified Messaging, Alternate Extension, etc. are only imported when the User is added or if there is a modification done to the User, e.g. First Name, Last Name, Email address, etc.

- If services are added directly to the User on Cisco Unity Connection, e.g. when adding Unified Messaging or user related services like Alternate Extension, this service will not be imported when running the next full import from CUC. To import these services a Model Type List must be applied to a dedicated Data Sync to target the required Model Types. A default Model Type list **CUCXN Overbuild Resources** exists for this purpose, which includes the following model types:
 - device/cuc/User
 - device/cuc/UserPassword
 - device/cuc/UserPin
 - device/cuc/AlternateExtension
 - device/cuc/SmtpDevice
 - device/cuc/SmsDevice
 - device/cuc/PagerDevice
 - device/cuc/PhoneDevice
 - device/cuc/HtmlDevice
 - device/cuc/Callhandler
 - device/cuc/CallhandlerMenuEntry
 - device/cuc/CallhandlerTransferOption
 - device/cuc/Greeting
 - device/cuc/MessageHandler
 - device/cuc/ExternalService
 - device/cuc/ExternalServiceAccount
- If making changes on the CUC directly to the schedules, then it is recommended that a dedicated sync be created which will pull in the all the Schedule related models (4 models) using the MTL **CUCXN Schedules**.

- It is recommended to use the `CUCXN Exclude ImportUser` MTL on Cisco Unity Connection Data Syncs in order to avoid unneeded data and slowing the sync time.

5.4. LDAP

5.4.1. LDAP Sync

The LDAP sync process currently only supports regular syncs.

5.5. Cisco Webex App (Spark)

5.5.1. Cisco Webex App Sync

If Cisco Webex App (Spark) is part of the solution and being managed, there are a number of considerations around sync with Cisco Webex App.

The typical setup is that the Cisco Webex App users are fully managed by the VOSS Automate system so there is no need for user sync. In this setup the only sync required is to pull in basic system data from Webex App for VOSS Automate to utilize in user configuration. A sync for this is added into VOSS Automate when a Cisco Webex App Service is added to the system and is executed automatically after Service creation or can be initiated by an admin as needed:

- `SyncSparkRolesLicenses<customername>` - Sync of basic data - e.g. licenses and roles, etc.

In an alternate scenario where some element of user management is occurring outside of VOSS Automate (for example, LDAP Connector), then a user sync will be required to pull that data into VOSS Automate for further management. Once the users are synced into VOSS Automate, they need to be moved to the appropriate site with the rest of the end user's services to be further configured and managed. This move can be done via the Webex App menu item by selecting the users and then using the **Action > Move** option to move them.

This sync can be initiated by an administrator as needed or if required, a schedule can be setup to run the sync on a regular interval.

- `SyncSpark<customername>` - Full sync of Webex App (Spark) including user data.

When VOSS Automate is integrated with a customer's user directory, the normal Subscriber management approach applies, in other words:

- Users are synced into VOSS Automate at the Customer hierarchy level
- Users must be moved to the relevant Site hierarchy level
- Once at the correct Site level, Quick Add Subscriber or Advanced Subscriber can be used to enable services (Webex App in this case) for the users

5.6. MSGraph and MS-Teams Sync

5.6.1. Microsoft Sync Overview

Overview

Automate interfaces with and syncs with a range of Microsoft applications, depending on the specific setup and use in your environment. The table describes how these applications sync in to Automate:

Application	Synced in via	Description
Microsoft Entra ID (Office365)	Graph API	Syncs in Msol users. Used for managing licensing.
Microsoft Teams	PowerShell	Syncs in Csol users that have a matching Msol entry. Used for managing voice, voicemail, and collaboration.
Microsoft Exchange Online	PowerShell	

Related Topics

- Introduction to Data Sync in the Core Feature Guide
- Controlling a Data Sync with a Model Type List in the Core Feature Guide
- Create a Targeted Model Type List in the Core Feature Guide
- Model Instance Filter in the Core Feature Guide
- Microsoft Quick Start Guide in the Core Feature Guide
- [Microsoft Entra ID and Office365 MSOLUser Filter Examples](#)

Sync from Microsoft Entra ID / Office365

Syncs from Microsoft Entra ID (Azure Cloud) sync in Msol users and are used to manage licensing.

This sync fetches users and licenses from the Microsoft Entra ID instance for the tenant. This can either be the source AD or it could be synced with an on-prem AD which is the real source. A combination of sources is also possible depending on the setup.

The licensing for cloud services is controlled on this user object in Microsoft Entra ID.

Primary purpose of syncs with Microsoft Entra ID:

- Provide the user objects and details that are in the Microsoft Entra ID environment, including existing settings and licenses.
- Fetch license details for the Microsoft tenant - which licenses are valid for the tenant, current available/used statistics, as well as the service plans for each License SKU.

The setup and timing of the sync can depend on how much of this data is required for your environment and how often you're viewing and using the data. For example, if you're using Automate to assign licenses to users as part of user onboarding, then syncing in users and licenses is important and needs to be timely in

order to pick up new users in the system that need licenses. However, if licensing is being handled outside of Automate, this sync is less critical in terms of frequency.

As this is a sync with the corporate AD, it is recommended that you apply a filter to control what's being synced in to the system. Filters can limit the number of users synced in to save load on the system, to reduce the time it takes for a sync to execute, and to enhance data security by ensuring that only relevant data (required for Automate's management purpose) is synced in.

These types of filters are set up via model instance filters (MIFs) in Automate.

The table describes common use cases for filtering in syncs:

Use case	Description
Geography	Filter users by geography - certain cities and/or countries - potentially for a limited rollout of voice services in some geographies/sites, but not others.
Licensing	Filter users based on O365 licensing. You can have a simple filter that only pulls in licensed users (with any license), or you can have a more specific filter to identify licenses you want to match. The "IsLicensed" field is a filterable field for MsolUser and is the easiest filter (any license). Building a list of license IDs and permutations is more complex, but can be useful to narrow the sync down to a specific service (for example, those with licenses related to voice services such as E5, Phone System, or Calling plan). When contemplating this condition, consider whether you're managing users without a Phone System - for example, MS Exchange, or MS Teams users without Phone system, and so on - in which case a specific license filter may not be appropriate.
Combination	You can filter also use a combination of filters, that is, users from a certain geography, with an appropriate license.

Automate includes example model instance filters for the scenarios outlined above. You can clone and use these model instance filters, and adjust as needed - including your City name(s), country name(s) or License SKU ids, for instance.

Plan Your First Sync

If your environment has more than a few thousand users, it is recommended that you plan your sync strategy and not just do a full sync.

Use these guidelines to plan an approach based on the number of users you wish to pull in, and adjust the sync before you start pulling any users in - for example:

1. Start with a sync of licenses only (without users) to determine which licenses are in the environment, and the corresponding IDs, to set up in your model instance filters (if filtering by license).
2. Consider if your deployment will only include certain geographic regions and you can set up filters for those regions (maybe even starting with a single region initially to test and finalize setup).
3. Once you have your filters in place, run the sync, then adjust as needed.

Sync from Microsoft Teams

Syncs from Microsoft Teams pulls in a range of elements from the MS Teams tenant to allow Automate to manage the tenant. Fetched elements include, for example, users, voice, voicemail, collaboration, resource accounts, numbers, policies, and teams.

Note: When syncing in data from `msteamsonline/CsOnlineUser`, Automate sets the default maximum number of records fetched at 130 000. Paging is supported, and records are sorted on the `UserPrincipalName` (UPN). You'll need to contact VOSS support to reduce this value since a lower value may impact performance.

Users are the main consideration for setting up this sync since creating and updating users is related to the licensing of users in Microsoft Entra ID. A user won't appear in MS Teams unless they have a relevant license assigned in Microsoft Entra ID/O365. Once a license is assigned, Microsoft automatically creates the users as a base user into MS Teams. Base users must be synced in to Automate before they can be managed, for example, to assign policies and enable services for these users.

Default Data Syncs

Automate syncs in all O365 and Teams data via the default data syncs that are created when a tenant is added. No Model Type List (MTL) is applied to the default syncs, which means that Automate syncs in all supported data for the tenants, including all `CsOnlineUsers`, that is, both enabled and disabled users (`accountEnabled=true` and `accountEnabled=false`).

Since user accounts remain in MS Teams even when licensing to MS Teams is removed from the user or the user is removed from Microsoft Entra ID/O365, syncing in all users in the default sync allows Automate to identify any numbers currently assigned to disabled subscribers. This ensures that the number inventory is accurate, and prevents assignment of duplicate numbers. In addition, the Microsoft Entra ID (M365) data syncs have *Quick Import* enabled, which improves the performance of the sync.

To limit the sync to enabled users only (`accountEnabled=true`), you can apply a model instance filter to future syncs; that is, set up a sync with a model instance filter by using `accountEnabled equals true` as a condition. You may also consider menu filters and other configuration to separate disabled accounts from enabled accounts.

As with the Microsoft Entra ID sync, you may consider filtering for setting up the MS Teams sync in your environment. As the MS Teams integration currently uses PowerShell, there are additional considerations for filtering and setup to ensure the syncs are optimized to be efficient, and performance is acceptable.

PowerShell filtering guidelines for CsOnlineUser and MsolUser syncs

PowerShell supports filtering with the **equals** condition on some fields. It is recommended that you plan your filters and conditions based on these fields due to the filtering that happens on the Microsoft end. This will significantly improve the PowerShell response time and limits the amount of data being passed between the systems for processing.

Syncs using these conditions are faster and consume less bandwidth.

The table describes the fields that PowerShell supports for filtering, for `CsOnlineUser` and `MsolUser`:

CsOnlineUser filtering

- City
- Country
- UsageLocation
- Department
- FeatureTypes
- AccountEnabled
- UserPrincipalName
- SamAccountName

Mso1User filtering

- IsLicensed
- City
- Country
- Licenses.Skuld
- UserPrincipalName

Note: There are an additional 15 read-only extension attributes to filter on for Microsoft subscribers (when MS Exchange is installed). The values for these fields are customizable via your Active Directory server only (not in Automate), and are synced in with MS Entra ID via the MS Graph API.

The values for these attributes can be modified in Automate via the following device models, provided the MS Entra ID Active Directory server is not synced with an On Premise Active Directory server: `device/msexchangeonline/UserMailbox` and `device/msexchangeonline/SharedMailbox`

You can find more information about these On Premises Extension Attributes in the following topic:

- [View and Update a Microsoft Subscriber in the Core Feature Guide](#)

If your requirements can't be met via the fields above, you can filter on other fields on the `CsOnlineUser` and `Mso1User` objects. This filtering is applied to the results provided by PowerShell back to the system using the PowerShell filter. This results in the additional non-PowerShell criteria being treated as an AND criteria with the PowerShell filter.

The filter is configured via the same model instance filter mechanism used for all other sync filtering. Automate takes the model instance filter and breaks it down to use it in the following way:

- The model instance filter is checked for any criteria that uses the fields supported for PowerShell filtering. It builds the PowerShell command and issues it using those fields as a filter.
- The condition is `equals` regardless of the condition set in the model instance filter for that field (PowerShell only supports the `equals` condition).
- The result is compared against the Automate cache for users that have been added, updated, or deleted.
- With this resulting list of users, any further criteria from the model instance filter is applied to the returned data.
 - Conditions for these fields are applied as configured in the model instance filter
 - This results in an AND condition between the PowerShell filter criteria and the “regular” model instance filter criteria.

Although combining criteria in a model instance filter allows for less complexity and more predictable results, it is recommended that you avoid doing this, if possible. If you need to mix criteria, it is recommended that you test this carefully for expected results. For example, consider a filter with:

- Filter on country (can happen in PowerShell filtering) and telephoneNumber (not in PowerShell filtering):
- The first list of users from the device would be filtered based on country (PowerShell Filter)
- Code does the compare to see the resulting set of users that are added/updated/deleted
- telephoneNumber filter is then applied.
- The result of the model instance filter is users where country and telephone number match

Note: It is possible to filter by `UserPrincipalName` (UPN) in the model instance filter. Due to size limitations, this is implemented slightly differently in terms of capturing the list of `UserPrincipalNames` to filter.

Some common user cases and scenarios taken individually or in combination. The criteria needs to be equals, and multiple values can be provided - either as AND or as OR, depending on the setup in the model instance filter. These cases can all be handled via PowerShell filtering so are the preferred use cases:

Filter users by geography - Country	Pull in users from a subset of countries in the environment
Filter users by geography - UsageLocation	Similar to country, to pull in users in certain UsageLocations
Filter users by geography - City	Possible but less efficient as still requires pulling all the users into Automate, then filtering by city. Thus, best if used in conjunction with another PowerShell filter field.
Filter users by department	Pull in users assigned to certain departments

The system includes a few sample MIFs for these use cases. The samples can be cloned and adjusted to meet specific needs.

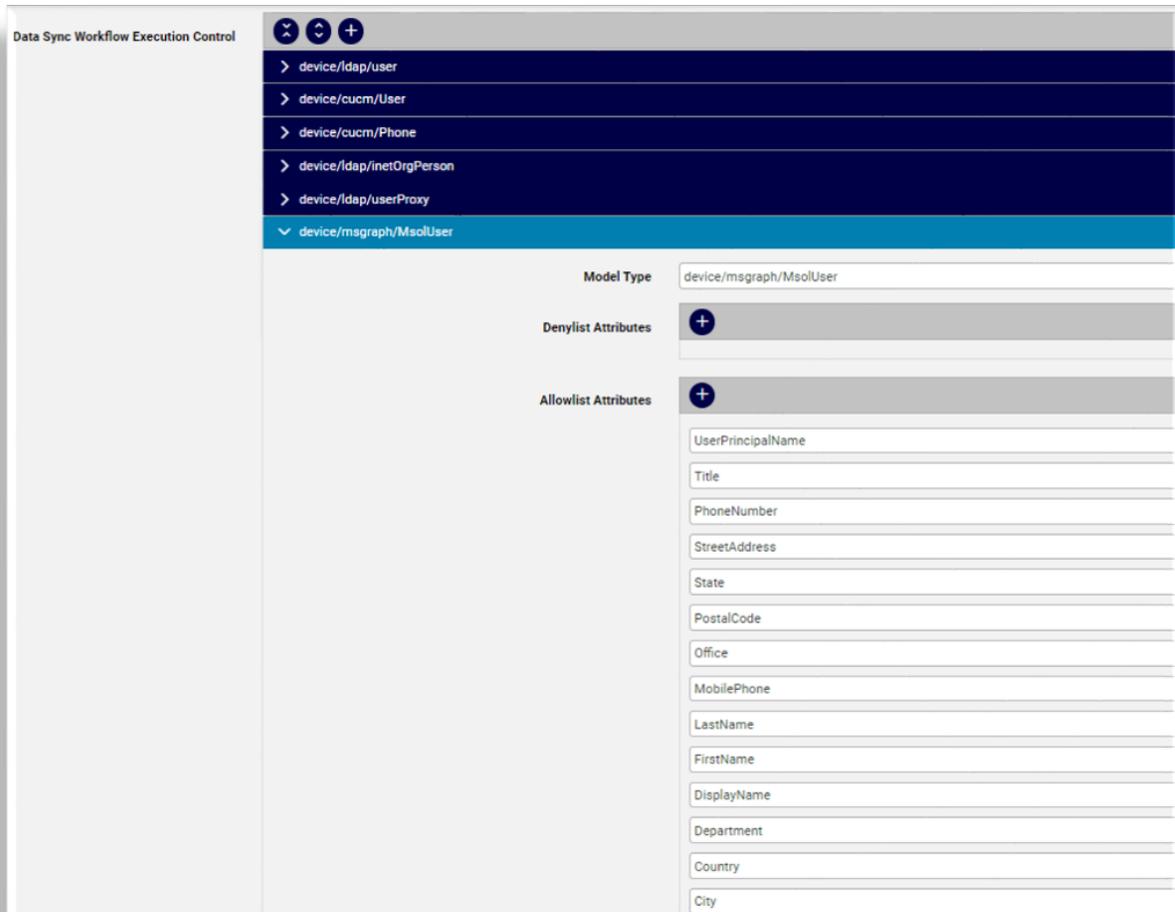
Example model instance filters included in Automate:

MSTeamsOnline-CsOnlineUser-	A simple model instance filter that provides an example of how to filter on users matching <i>United Kingdom</i> OR <i>United States</i>
MSTeamsOnline-CsOnlineUser-	A model instance filter that includes an example of how to filter on users matching a country and multiple cities in the country.

Allowlist and Denylist Considerations

Automate ships with allowlist and denylist entries for Microsoft Entra ID and MS Teams users to help optimize syncs for these applications.

High level admins (`sysadmin`) with access to the `data/Settings` model can view and modify the default allowlists and denylists for their scenario and environment, for example, to meet their flow through provisioning requirements. The predefined allowlist and denylist entries that ship with the system are typically suitable for most cases, and will account for the fact that sync workflows for a user may not be triggered even though a change for the user was pulled in during the sync. VOSS may update these default lists from time to time during upgrades if future features/functionality require it.



Related Topics

- Allowlists and Denylists in the Core Feature Guide.

See Settings (Data Sync Workflow Execution Control) in the Advanced Configuration Guide.

Workflow Execution Phase Considerations

For any Hybrid syncs, ensure that the inclusion of the following workflows have the **Phase** set (or moved) to Post Execution:

- UserMS365SyncRemove
- UserMSTeamsSyncRemove

Delete Protection for Syncs

While there are a number of considerations for the setup of Microsoft syncs with Automate, it is also important to consider delete protection. In this case, the relevant macros for Microsoft Entra ID and MS Teams syncs are the following:

- PULL_SYNC_DELETE_THRESHOLD_MSTeamsOnline
- PULL_SYNC_DELETE_THRESHOLD_MSGraph

These macros help to prevent mass deletes above a configured threshold defined in the macro (default 10). You can adjust the values for these macros initially, if required (depends on churn in your environment) or adjust them later as needed, based on the sync messages.

Note: For more information on sync delete thresholds, see:
the Pull Sync Delete Threshold topic in the Advanced Configuration Guide.

Syncing Teams and Groups on Large Microsoft Tenants

Microsoft Teams users often create ad-hoc Teams with channels, settings, and members. Microsoft tenants in large-scale deployments may thus contain hundreds of thousands of Groups as a Group is created in Microsoft Entra ID each time a Team is created. Additionally, Teams may contain many channels and members.

Previously, syncing in all data for a large Microsoft tenant resulted in inefficient and time-consuming syncs that included a large volume of unnecessary details, including Team members and channels. To reduce the time it takes to sync in Teams and Groups, Automate syncs in only high-level details for the list view for Teams and Groups on the tenant, via the `device/msgraph/Group` model. Additional details for the Team or Group is fetched as a live update on demand (when an admin clicks on a Team or Group to view its details).

Note: Separate models that existed for Teams (`relation/MicrosoftTeams`) and Groups (`device/msgraph/Groups`) are merged into one model, `device/msgraph/Group`.

Microsoft doesn't differentiate between Teams and Groups. In Automate, the **Teams** list view contains an *Is Teams* flag (green check) to indicate that the entity you're viewing is a Team and not a Group.

<input type="checkbox"/>	Name ↕	Group Type ↕	Mail ↕	Is Team ↕	Located At ↕	Device ↕
<input type="checkbox"/>	All Company	Microsoft 365	allcompany@MODERNCOMMS534550.onmicrosoft.com		Synergy (Customer)	Connection parameters for Microsoft Graph Synerg
<input type="checkbox"/>	All Employees	Security	Employees@MODERNCOMMS534550.OnMicrosoft.com		Synergy (Customer)	Connection parameters for Microsoft Graph Synerg
<input type="checkbox"/>	All Users	Security			Synergy (Customer)	Connection parameters for Microsoft Graph Synerg
<input type="checkbox"/>	Ask HR	Microsoft 365	askhr@MODERNCOMMS534550.onmicrosoft.com		Synergy (Customer)	Connection parameters for Microsoft Graph Synerg
<input type="checkbox"/>	CEO Connection	Microsoft 365	ceoconnection@MODERNCOMMS534550.onmicrosoft.com		Synergy (Customer)	Connection parameters for Microsoft Graph Synerg
<input type="checkbox"/>	Contoso Life	Microsoft 365	contosolife@MODERNCOMMS534550.onmicrosoft.com		Synergy (Customer)	Connection parameters for Microsoft Graph Synerg

Related Topics

- Teams in the Core Feature Guide
- Groups in the Core Feature Guide

5.6.2. Microsoft Entra ID and Office365 MSOLUser Filter Examples

Note: For general information on Data Sync, Model Type Lists (MTL) and Model Instance Filters (MIF), refer to the chapter on Data Sync in the Core feature Guide.

Overview

Automate ships with a number of model instance filters to get you started with examples you can clone and work from. You can find these are the model instance filters that have names starting with MSGraph-MSOL and include many examples with different SKUIDs, country filters, etc.

Below are a few examples explained around Microsoft Entra ID/Office365 MSOLUser records.

Note: Microsoft changed the name of Azure Active Directory to Microsoft Entra ID.

Example 1 - Filter to only sync users assigned any License

This is one of the out of the box model instance filters (MIFs): MSGraph-MSOL-MIF_wLicence. This example shows the filter setup for Microsoft Entra ID/Office365 Users (device/msgraph/MSOLUser record) to use the field IsLicensed. This is a simple filter and instead of adding a comprehensive lists of relevant SKUs to the filter, it makes it easy to search for users that have a license assigned or not. This should be the most common filter and likely should handle most situations where the licensing of users is handled outside of the Automate system.

The image shows a configuration interface for filters. It is organized into a tree-like structure:

- Model Parent:** device/msgraph
- Filter Type*:** Include Matching Instances
- Model Filters:** A container with a '+' icon and a sub-filter:
 - Model Type*:** device/msgraph/MsolUser
 - Attribute Filters:** A container with '+' and '-' icons and a specific filter:
 - Attribute Name*:** IsLicensed
 - Condition*:** Equals
 - Value:** True

Example 2 - Filter to only sync users assigned specific licenses

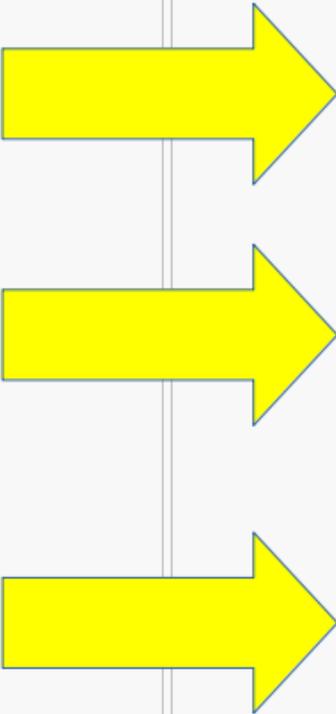
This example shows the filter setup for Microsoft Entra ID/Office365 Users (device/msgraph/MSOLUser record) to use the field SKUID. This setup will sync in users that have any of the 3 SKUs identified in the filter. As with the above example, we have numerous example model instance filters out of the box (names start with MSGraph-MSOL-MIF_wLicence) that include the various SKUIDs for most of the common licenses to save some effort in looking those up.

Model Parent:

Filter Type*:

Model Filters

-
-
- Model Type*:
- Attribute Filters
 -
 -
 - Attribute Name*:
 - Condition*:
 - Value:
-
- Attribute Name*:
- Condition*:
- Value:

-
- Attribute Name*:
- Condition*:
- Value:


6. Data Collection

6.1. Recommended RIS API Data Collector Interval

As a guideline to determine the interval that the (RIS) data collector service should poll the Unified CM, consider that:

- it takes about 14 minutes to collect information for around 200K phones on a cluster

The default value of the **RIS API data collector interval**: 43200 seconds (12 hours) can be adjusted accordingly.

Note: Collection processes run in parallel for each Unified CM on VOSS Automate.

To adjust the value, refer to the System Monitoring Configuration section in the Advanced Configuration Guide.

7. API Performance

7.1. API Resource Listing Best Practice

This section provides best practices when using API GET requests when listing resources. The best practices for the use of a number of API request parameters and parameter values are examined.

For further details on API parameters, refer to the API Guide.

The list of API request parameters for resource listing are:

Parameter	Description	Value	Default
skip	The list resource offset as a number.		0
limit	The maximum number of resources returned. The maximum value is 2000. If the Range request header is used, it will override this parameter.	1-2000	50
count	Specify if the number of resources should be counted. If false, the pagination object in the response shows the total as 0, so no total is calculated and the API performance is improved.	true, false	true
order_by	The summary attribute field to sort on.		First summary attribute
direction	The direction of the summary attribute field sort (asc:ascending, desc: descending).	asc, desc	asc
summary	Only summary data is returned in the data object.	true, false	true
policy_name	Return a model form schema where the Field Display Policy with name [FDP name] is applied to it. Use policy with the parameters schema and format=json.	[FDP name]	
cached	System will respond with resource information where the data was obtained from cache. (Functionally only applicable to device models and data models).	true, false	true

Consider the following comments and best practices for the parameters below:

- **count**: The value of `count=true` is very expensive in terms of performance, and more so as the size of the resource grows. The first count query of for example a 36 000 Data Number Inventory resource can take as long as a minute to return a response. However, subsequent calls should decrease in execution time.

The value `count=true` should only be used if it is unavoidable. An alternative is to iterate over pages (`limit=200`) until the request returns less than 200 instances, or to simply paginate until no more resources are returned.

- `order_by`: no performance change if another summary attribute is specified.
- `direction`: no performance change if either values `asc` or `desc` are used.
- `policy_name`: the parameter is used by the GUI for display purposes. Timing data shows that the initial call with this parameters takes longer than subsequent ones, possibly because of cache priming after a restart. Subsequent calls shows the execution time is on par with requests that do not include the parameter.
- `summary`: depending on the data required by the request, time can be saved if the value `summary=true`, so that only the summary data is returned.
- `limit`: execution time and memory consumption is impacted if the `limit` value is large.

To summarize, the recommended parameter values for an optimal API list request (GET) are:

- `cached=true`
- `summary=true`
- `count=false`
- `policy_name` not used

Example results with various parameter values (36 000 Data Number Inventory resource):

```
count:true, skip:0, policy_name:, limit:200, summary:false in 6.51744103432 s
count:true, skip:0, policy_name:, limit:200, summary:true in 5.6118888855 s
count:false, skip:0, policy_name:, limit:200, summary:false in 1.55350899696 s
count:false, skip:0, policy_name:policy_name=HcsDNInventoryDatFDP, limit:200,
summary:true in 5.17663216591 s
count:false, skip:0, policy_name:, limit:200, summary:true in 1.09510588646 s
```

7.2. Long Running API Requests

To optimise memory utilization and performance, the system has been configured so that the API server will manage workers with the following defaults:

- after receiving a restart signal, workers have 100 minutes to finish serving requests
- a random restart interval of between 0 and 600 requests per worker (4 workers per node, 4 nodes in a cluster)

API best practices is to schedule and then poll transactions, since long running requests can affect recycling. In other words, preferably short requests and then poll.

7.2.1. Polling Example

To retrieve the status of a given transaction:

```
GET /api/tool/Transaction/[pkid]/poll/?format=json
```

The response contains essential status of the transaction, for example:

```
{
  [pkid]: {
    status: "Success",
    href: "/api/tool/Transaction/[pkid]",
    description: "Name:RDP-auser1857 Description:RD for auser1857"
  }
}
```

Refer to the topics *Poll Transactions* and *Example of an Asynchronous Mutator Transaction with nowait=true* in the API Guide.

8. System Maintenance

8.1. Transaction Archiving

The following are considerations when determining the frequency of the transaction archiving schedule to set up on the system. If a schedule is not set up for transaction archiving, system Alerts will be raised as well as a warning on the platform CLI login:

TRANSACTION DATABASE MAINTENANCE NOT SCHEDULED

- Run **voss transaction count <days>** on your system to inspect the number of transactions during a given period to determine your usage metrics.

Refer to the *Database Commands for Transaction Management* topic in the Platform Guide for details on transaction archive command use and scheduling:

- **voss transaction delete <days>**
- **voss transaction export <days>**
- **voss transaction archive <days>**
- Business policies - company policies may drive your choices: the immediate access to transaction logs for a period of time, security policy on data/audit retention, and so on.

Note: The transaction archive process does mean the logs are not lost, just that they are not immediately accessible in the administrator graphical interface for searching.

- You can also set up system monitoring thresholds so that you receive alerts via the GUI and SNMP if the threshold is exceeded - which might indicate you need to review the archive schedule to increase how frequently it runs.

See the *SNMP* and *VOSS Automate System Monitoring Traps* topics in the Platform Guide.

8.2. Automated Database Cache Cleanup

From VOSS Automate release 19.3.2 onwards, it is now longer necessary to schedule or manually manage the database cache optimization using the **voss trim-cache** command.

From release 20.1.1, this command is no longer available. A resource history is now maintained as a series of resource differences and is automatically optimized.

Note: A minimum retention period of 7 days is applied to resource differences in the resource history.

9. Administration Portal Setup

9.1. Navigation - Menu and Dashboards

9.1.1. Overview

VOSS Automate provides several tools for customizing the Portal experience to your requirements.

The advanced Admin Portal utilizes two key ways to provide users with the means to navigate around the system to key features:

- Configurable navigation menus (on the left of the screen)
- Configurable Home page - this is the page you see when logging on or when clicking the Home button

Related Topics

- Dashboards in the Core Feature Guide
- Menu Layouts Core Feature Guide

9.1.2. Navigation Configuration Options

The table describes configuration options to enhance navigation:

Configuration	Description
Naming conventions for menus and dashboard	Use naming conventions relevant to your users and organization. For example, use business-relevant names for admins, and technical terms only for advanced users.
Linking from menus	Links from menus are typically set up to the form/view or a list or to other system models that users need access to for various tasks.
Display policy	The display policy for views, or for when users choose an entity from a list view, which determines the form layout displayed to a user.
Configuration template (CFT)	If you've chosen a CFT, it is applied when a view is accessed from the menu item, or if the user clicks Add from a list view. The CFT can also define default fields (or read-only default values for read-only fields), as well as default values for fields that the CFT hides from view.
Fixed filters	Configured for a menu layout, and applied by default in the back-end and thus not visible to a user.
Configurable filters	May be fully or partially defined for menu layouts pointing to lists. This filter provides an interim step between clicking a menu item or link, and opening a list view. A configurable filter launches a pop-up allowing a user to enter filter criteria relevant to the menu item. When the user chooses the criteria the list view displays based on the criteria. Users can view, modify, or delete configurable filters.

Note: See the Core Feature Guide for more information on navigation configuration options.

9.1.3. Navigation Strategies

It's important that you provide efficient methods for users to navigate through the system and to access required functionality.

Ensure frequently used functionality is easy to reach for different types of users. Create menus and the dashboard based on the requirements of different user roles, and review these requirements regularly to ensure the greatest efficiency and user experience.

The table outlines key efficiency outcomes for enhancing navigation and the user experience:

Goal	Description
Quick Access to tasks and searches	<p>Populate dashboards with frequently used tasks and searches to provide access with one click, and the ability to easily return to quick access functionality via a Home icon.</p> <p>EXAMPLES:</p> <ul style="list-style-type: none">• Access to top MACDs on dashboards, with appropriate FDPs and CFTs. See Feature Experience.• Add saved searches with predefined filter criteria as links. For example, a saved search for the list of unregistered phones, linking to phones with criteria set to <i>status starts with unregistered</i>. Note that you'll only have access to the Saved Searches functionality if your access profile permissions allow it (read permissions on data/UserSavedSearch).• Links with configurable search criteria, for example, to find a phone by user. This can be done with a link for Phone, and filter criteria set to ownerID. In this case, the user is prompted to provide the user name to find a phone. <p>If you don't have enough space on a dashboard for all the frequently accessed tasks you wish to add, add these items to the menus.</p>

Goal	Description
Feature experience	<p>Instead of creating one link to a feature that has many scenarios, you may want to include multiple menus or dashboard entries to the same feature, but use different display policies, CFTs, and filter options for specific use cases.</p> <p>This can be a very simple way to create an experience of a feature for specific scenarios, and prevent relying on users to follow a procedure or enter specific information.</p> <p>EXAMPLES:</p> <ul style="list-style-type: none">• Create a link to add SIP trunks as part of a regularly added 3rd party application integration. This can link to the SIP trunk device model, and use a display policy that shows only settings requiring user input (such as IP address and the port for the remote system). The CFT could predefine all other technical settings based on the scenario (for example, CSS, call presentation details, digit manipulation, and so on).• Create lines for different scenarios. Use display policies to show only those fields that must have user input, while CFTs pre-populate settings based on the scenario. You can add menu items for line type A, or line type B, for example, to cover the various scenarios, and combine this with filters in the menus and landing pages to differentiate between line types in the list views.• UCM feature management: UCM can be managed in VOSS Automate by accessing the device models directly. The API definition from Cisco drives the default device model layout, and may include field names and ordering that doesn't align with the VOSS Automate Admin Portal experience. You can improve this display policies to impose your preferred order and field labels. Combine this with CFTs to predefine default values or to simplify user input requirements to reduce possible setup errors.

Index

V

voss

- voss queues, 21
- voss transaction archive, 49
- voss transaction count, 49
- voss transaction delete, 49
- voss transaction export, 49
- voss workers, 5, 13

W

web

- web service, 5, 13